UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

MANEJO DE LOS PUERTOS VGA Y PS/2 CON VHDL PARA LA COMUNICACIÓN DE LA TARJETA BASYS2 CON UN MONITOR Y UN TECLADO EN EL LABORATORIO DE LÓGICA DIGITAL DE LA UNIVERSIDAD DE CARABOBO

Tutor Académico:
Prof. Demetrio Rey Lago

Autores:
Fung, Frank C.I. 20.082.306
Moyano, José C.I. 20.969.995

Valencia, Febrero de 2015

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA
DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

MANEJO DE LOS PUERTOS VGA Y PS/2 CON VHDL PARA LA COMUNICACIÓN DE LA TARJETA BASYS2 CON UN MONITOR Y UN TECLADO EN EL LABORATORIO DE LÓGICA DIGITAL DE LA UNIVERSIDAD DE CARABOBO

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TITULO DE INGENIERO ELECTRICISTA

Tutor Académico:

Prof. Demetrio Rey Lago

Autores:

Fung, Frank C.I. 20.082.306

Moyano, José C.I. 20.969.995

Valencia, Febrero de 2015

UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA ELÉCTRICA

DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

CERTIFICADO DE APROBACIÓN

Los abajo firmantes miembros del jurado asignado para evaluar el trabajo especial degrado titulado “Manejo de los puertos VGA y PS/2 con VHDL para la comunicación de la tarjeta BASYS2 con un monitor y un teclado en el Laboratorio de Lógica Digital de la Universidad de Carabobo”, realizado por los bachilleres Frank Fung, C.I.:20.082.306, y Jose Moyano, C.I.:20.969.995, hacemos constar que hemos revisado y aprobado dicho trabajo.

_____

Prof. Demetrio Rey Lago
TUTOR

_____

Prof. Wilmer Sanz
JURADO

_____

Prof. Luís Llave
JURADO

AGRADECIMIENTOS

A Dios en primer lugar, por brindarnos de vida, salud y permitirnos culminar esta meta tan importante para nuestras vidas.

A nuestros padres, Jesús Moyano, Elsi Mariela de Moyano, Fung Tai Lam y Lin Wen Hui por servirnos de bastón de apoyo en los momentos de flaqueza y necesidad y por enseñarnos gran parte de lo que somos ahora.

A Madeleine, por su constante apoyo en los momentos en que más necesitaba de una mano de apoyo y ser para mí la persona más especial, que me impulsa a ser mejor cada día.

A nuestro tutor Demetrio Rey Lago, por siempre tratarnos con empatía y siempre atendernos incluso en días complicados, por impulsarnos a seguir adelante.

A Verónica Hernández por facilitarnos sus prácticas de la materia Lógica digital.

A Jherica Pulido por recomendarnos la herramienta Snagit, que tanto ayudo en la edición de imágenes de este Proyecto.

José G. Moyano C.

Frank N. Fung L.

DEDICATORIA

A Dios ante todo, por ser la luz que nunca se apaga.

A mis padres por su entrega total a lo largo de mi vida, y dar siempre todo sin esperar nada a cambio, dando ejemplo en todo momento de lo que esperan de mi.

A mis hermanos, que me han ayudado en los momentos difíciles y servirme de inspiración y motivación.

José G. Moyano C.

A mis padres y a mi hermano por siempre estar allí en todo momento, por el constante apoyo que me han dado durante toda la vida y todo el aprecio que me han dado.

A mis amigos de la Facultad de Ingeniería, por compartir esos buenos momentos de risa, de simpatía y de apoyo.

Al profesor Demetrio Rey Lago, por ser un excelente tutor, siempre apoyándonos e impulsándonos por seguir nuestras metas con este Trabajo de Grado.

Frank N. Fung Lin.

INDICE GENERAL

LISTA DE TABLAS

LISTA DE FIGURAS

MANEJO DE LOS PUERTOS VGA Y PS/2 CON VHDL PARA LA COMUNICACIÓN DE LA TARJETA BASYS2 CON UN MONITOR Y UN TECLADO EN EL LABORATORIO DE LÓGICA DIGITAL DE LA UNIVERSIDAD DE CARABOBO

Autores: Frank N. Fung L.

Jose G. Moyano C.

Tutor Académico: Demetrio Rey Lago.

RESUMEN

El objetivo principal de este proyecto es el manejo de los puertos PS/2 y VGA, que posee la tarjeta lógica programable Basys2 para conectarlo con un teclado con puerto PS/2 y un monitor con puerto VGA, a través de la herramienta Active-HDL del entorno de Aldec. Ésta programación se realizó en el lenguaje para descripción de hardware VHDL. Se analizó el funcionamiento de los puertos PS/2 y VGA, para realizar la programación en lenguaje para descripción del hardware VHDL y así lograr la comunicación de la tarjeta con el teclado y con el monitor. Se desarrolló una interfaz gráfica, en donde se ilustran los pulsadores, interruptores y leds de la tarjeta Basys2 en el monitor, además se demostró el uso de la interfaz con una aplicación combinacional y otra secuencial, se desarrolló un manual de usuario en donde se explicara paso a paso como usar la interfaz así como la descripción del bloque de programación con sus subprogramas, para el fácil entendimiento del usuario.

Palabras claves: VHDL, PS2, VGA, BASYS2, Lógica Digital

INTRODUCCIÓN

El Presente Trabajo Especial de Grado se orienta hacia la manipulación de los puertos VGA y PS/2 de la tarjeta Basys2, para el desarrollo de una interfaz gráfica que beneficie al Laboratorio de Lógica Digital de la Universidad de Carabobo.

El presente Proyecto se encuentra organizado en 4 capítulos, los cuales se describen a continuación:

Capítulo I FORMULACION DEL PROBLEMA, En este capítulo se describe el planteamiento del problema, la justificación del proyecto, el alcance y los objetivos a cumplir en este Trabajo Especial de Grado.

Capítulo II ANTECEDENTES Y BASE TEORICA, se explica la composición de la de entradas y salidas de la tarjeta lógico programable Basys2 y el software en el cual se desarrollara la programación, y se explica además, la composición de los puertos a emplear en este Trabajo Especial de Grado para lograr la comunicación entre la tarjeta Basys2 y estos puertos.

Capítulo III MARCO METODOLOGICO, en esta sección se describe el procedimiento a seguir para llevar a cabo cada una correcta realización de este Proyecto de Grado, como lo es identificar en primer lugar las limitaciones de la tarjeta para así describir la estructura de diseño a emplear y los bloques necesarios.

Capítulo VI MARCO METODOLOGICO, Se describieron todos los bloques diseñados, tanto para la comunicación con los puertos VGA y PS/2 como para la realización de la interfaz gráfica en el monitor, así como los protocolos de comunicación entre los puertos y se realizó un manual de usuario.

CAPÍTULO I: EL PROBLEMA

1.1 PLANTEAMIENTO DEL PROBLEMA

Con los constantes avances en la tecnología, el entorno empresarial se encuentra en continua innovación y por ello el perfil del ingeniero electricista en sus diferentes ramas, demanda a las casas encargadas de su formación, adaptarse continuamente a las necesidades impuestas por estos avances tecnológicos, para así brindar a los estudiantes las herramientas necesarias para el desarrollo, utilización y mantenimiento de dichas tecnologías. De acuerdo al artículo Ingeniería eléctrica: Hacia una propuesta de currículo por competencias de la revista de la Facultad de Ingeniería Universidad Central de Venezuela, citando:

> "La sociedad actual demanda de las universidades la formación de profesionales idóneos que puedan adaptarse a la naturaleza acelerada de la evolución científica y tecnológica, para constituirse en agentes de cambios sociales positivos y conducentes al bien común" [1].

Es por ello que en este caso la Universidad de Carabobo, insiste en profundizar el desarrollo de nuevos temas en las asignaturas, para enriquecer la formación y capacidad de sus egresados y así contribuir con el desarrollo del país.

Una de las tendencias tecnológicas que ha surgido en la última década, han sido los FPGAs (Arreglo de Puertas Programable en campo - Field Programmable Gate Arrays), gracias a que permite realizar un circuito integrado a medida, sin los riesgos económicos asociados a las otras opciones tecnológicas y a su bajo costo. Hoy las FPGAs están presentes en campos tan diversos como la automoción, la electrónica de consumo, o la investigación espacial. La tecnología FPGA tiene una aplicación horizontal en todas las industrias que requieren computación a alta velocidad.

En semestres anteriores se ha implementado el lenguaje para descripción de hardware VHDL en el Laboratorio de Lógica Digital de la Universidad de Carabobo para configurar la tarjeta lógico programable Basys2, el cual anteriormente se impartía de manera informativa. Las primeras prácticas del Laboratorio mencionado se enfocaban en el desarrollo y montaje de circuitos lógicos en el protoboard, y como herramienta de verificación se emplean software de simulación; posteriormente se hace uso del lenguaje para descripción de hardware VHDL o en su defecto el editor esquemático para la tarjeta.

La tarjeta lógica programable Basys2 usada en el Laboratorio de Lógica Digital de la Escuela de Eléctrica de la Universidad de Carabobo, cuenta con 250.000 compuertas además de 8 LEDS, 8 interruptores, 4 pulsadores, 4 displays de 7 segmentos, un conector PS/2, y una salida VGA. En el laboratorio solo están usando los pulsadores, los interruptores, los leds y los display de 7 segmentos, lo cual representa una limitante con respecto a la cantidad de variables de entradas y de salidas, además que no se aprovechan los puertos PS/2 y VGA.

Al no aprovechar los puertos PS/2 y VGA, se tienen una limitante con respecto a la cantidad de variables de entradas y de salidas. Un sin fin de prácticas que se pueden desarrollar al tener conocimiento de la programación de un teclado con puerto PS/2 y un monitor con puerto VGA, como desarrollar alguna aplicación, un juego virtual, entre otros, adicionalmente impartir conocimiento del funcionamiento de los protocolos como el PS/2 y VGA.

1.2 JUSTIFICACIÓN

Este proyecto se encuentra enmarcado en la línea de investigación Sistemas Digitales del Departamento de Sistema y Automática en la Escuela de Eléctrica de la Facultad de Ingeniería de la Universidad de Carabobo. Surge ante la necesidad que se presenta en el laboratorio de lógica digital de la Universidad de Carabobo, de aprovechan los puertos

disponibles en la tarjeta lógica programable Basys2 para optimizar su uso en el laboratorio que cuenta con un instrumento de alto nivel tecnológico como lo es la tarjeta lógica programable Basys2 y no se aprovechan sus puertos VGA y PS/2. La implementación de estos puertos, tiene beneficios como:

- Un mayor número de entradas y salidas adicionales, de la tarjeta lógico programable Basys2 para el desarrollo de las practicas.
- Optimizar el uso del tiempo en la realización de las prácticas en el laboratorio de lógica.
- Una manera amigable de comunicarse y hacer uso de los puertos PS2 y VGA, adaptado a los requerimientos del estudiante del curso.

El objetivo de la cátedra es brindarle al estudiante la capacidad de desarrollar circuitos lógicos, por medio del uso y comprensión de la lógica de Booleana, así como la aritmética involucrada en el diseño y manejo de dichos circuitos, para ello se requiere de herramientas que contribuyan a la enseñanza del funcionamiento de los componentes fundamentales de los circuitos lógicos, sin necesidad de enseñar un nuevo lenguaje de programación, que no aporta beneficios significativos a los requerimientos de la cátedra; sino que por el contrario retarda el desarrollo de las practicas realizadas, es por ello la fortaleza de este proyecto de investigación, la cual es facilitarle las herramientas al estudiante mediante un diseño adaptado a las necesidades del estudiante y de la cátedra, usando incluso puertos que estaban en desuso y así estimular su aprendizaje.

1.3 OBJETIVOS DE LA INVESTIGACIÓN

1.3.1 OBJETIVO GENERAL

Manejar los puertos VGA y PS/2 con VHDL para la comunicación de la tarjeta Basys2 con un monitor y un teclado, en el Laboratorio de Lógica Digital de la Universidad de Carabobo.

1.3.2 OBJETIVOS ESPECÍFICOS

- Identificar y establecer limitaciones de la tarjeta lógico programable Basys2.
- Utilizar el lenguaje VHDL para describir el sistema.
- Establecer y describir el código de programación para lograr una comunicación entre el teclado y la tarjeta lógico programable Basys2.
- Establecer y describir el código de programación para lograr una comunicación entre el monitor y la tarjeta lógico programable Basys2.
- Establecer el código de programación de una interfaz gráfica que se visualice en el monitor con puerto VGA en donde las variables de entradas serán tanto de un teclado con puerto PS/2 como de los componentes de la tarjeta Basys2.
- Implementar una aplicación combinacional y otra secuencial tomada de las prácticas de laboratorio utilizando la interfaz gráfica desarrollada.
- Realizar un manual de usuario para el uso de la interfaz modelo.

1.4 ALCANCE

Para el desarrollo del presente trabajo de grado se tomaron en cuenta las siguientes consideraciones:

- El código está ejecutable solo con Active HDL en forma de un bloque que tiene contemplados subprogramas para su uso en las prácticas, de igual manera se garantiza compatibilidad únicamente con la tarjeta lógica programable BASYS2 CP132 - 250.

- Se realizó un manual de usuario, en donde se describió cada entrada y salida del bloque principal de la interfaz gráfica, además se describen los subprogramas contenidos en el bloque principal.

- El diseño facilita la incorporación de dos números hexadecimales de dos dígitos a partir del teclado.

- Se realizaron bloques adicionales para adaptar la señal del teclado con el programa del usuario.

CAPÍTULO II: FUNDAMENTOS TEÓRICOS

2.1 ANTECEDENTES

En la actualidad, en diferentes Universidades alrededor del mundo ya están empleando el uso y estudio en la programación de los FPGA en los Laboratorios ya que estos tienen destacados beneficios, como el rendimiento, ya que excede la potencia de cómputo de los procesadores digitales de señales rompiendo paradigma de ejecución secuencial y logrando más en cada ciclo de reloj; el precio, ya que el diseño personalizado ASIC excede considerablemente al de las soluciones de hardware basadas en FPGA; la fiabilidad, Los FPGA no requieren de un sistema operativo, es por ello que minimizan los retos de fiabilidad con ejecución paralela y hardware preciso dedicado a cada tarea y el mantenimiento a largo plazo. Entre los trabajos que se pueden destacar en relación a los FPGA y/o programación en VHDL tenemos:

El trabajo de Ascenso de los Ingenieros SIMONE, A y PEREZ, A (2006) sobre la "Incorporación del lenguaje VHDL para la síntesis, descripción y simulación de circuitos lógicos digitales bajo el estándar IEEE 1076-2002" [2].

Con estos trabajos se le abrió la puerta al lenguaje de descripción de hardware (VHDL) en la asignatura de Lógica Digital, de la Universidad de Carabobo, tanto en la teoría como en el laboratorio de esta materia.

Guzmán A, [2009] "MANUAL DE REFERENCIA DE LA TARJETA BASYS2". Universidad Politécnica de Madrid. En este trabajo nos describe los componentes adjuntos en la tarjeta Basys2, el funcionamiento de cada una de ellas y además nos muestra con ejemplos, el funcionamiento de sus componentes en lenguaje de programación VHDL [3].

Con el manual del Basys2, se identificaron las entradas, salidas de la tarjeta y su correcta configuración, además de analizar correctamente los tiempos de simulación y comunicación.

Merino M, [2012] "VIABILIDAD DE LA TARJETA BASYS2 PARA SU IMPLEMENTACIÓN EN EL CONTROL DE UN PROCESO". Instituto Tecnológico de Tehuacán. En este trabajo el autor realiza el estudio de la viabilidad de la tarjeta Basys2 en controles de procesos, concluye que esta tarjeta cuenta con entradas y salidas limitadas y el proceso a implementar debe estar pensado en las capacidades de la tarjeta, el tiempo de diseño de un prototipo con la Tarjeta Basys2 es reducido y ofrece ventajas tales como la reprogramación y reasignación de pines para hacer pruebas que permitan llegar al producto final [4].

Aportes: Con este trabajo de investigación, se logró determinar el procedimiento para la identificación de la señal de entrada y como configurarla. Se diferencia de esa investigación puesto que además se prevé lograr configurar los puertos PS/2 y VGA.

López M, Anzueto A. [2013] "CONTROL Y DESPLIEGUE DE IMÁGENES EN MONITOR (VGA) MEDIANTE EL USO DE LENGUAJE VHDL Y UNA PLACA NEXYS 2" Instituto Politécnico Nacional. En este trabajo, tiene como objetivo principal el desarrollo del control y despliegue de imágenes en un monitor mediante una placa NEXYS 2, mediante la programación en VHDL, el método de configurar y programar los tiempos de pulsos, el conexionado de los puertos de comunicación, entre otros [5].

Aportes: Con este proyecto de investigación, se lograba el adecuado despliegue de programación del puerto VGA y sus consideraciones técnicas más resaltantes.

Haskell & Hanna (2009) Digital Design Using Digilent FPGA Boards. En este libro se realizan diferentes ejemplos usando el software Active-HDL, desde encender un LED, hasta realizar la comunicación con un monitor con puerto VGA, incluyendo también la

comunicación con un teclado con puerto PS/2. Se van desarrollando los programas ejemplos de manera progresiva e intuitiva y complementada con su respectiva documentación teórica, Con estos ejemplos se realizaron diferentes ejemplos a fin de familiarizarse con el lenguaje, y los dos últimos capítulos de este libro contribuyeron como base en el código para realizar la comunicación entre el monitor y el teclado, con puertos VGA y PS/2 [6].

2.2 MARCO CONCEPTUAL

Como en todo trabajo de grado se realizó una recopilación de información necesaria para el desarrollo del proyecto, a fin de identificar cada una de sus limitaciones en cuanto a hardware se refiere, así como también sus extensiones como el teclado y monitor, para así obtener una completa comprensión de sus características y modo de comunicación entre estos equipos.

2.2.1 TARJETA BASYS2

La tarjeta Basys2 es una plataforma para el diseño e implementación de circuitos digitales. La tarjeta está construida en base a un FPGA Spartan-3E de Xilinx y un controlador USB Atmel AT90USB2. La tarjeta Basys2 provee el hardware necesario listo para usarse capaz de soportar circuitos que van desde el rango de lo básico hasta el control complejo de procesos. Una amplia gama de dispositivos de E/S y todas las conexiones del FPGA son incluidas, por lo que pueden ser creados incontables diseños sin la necesidad de componentes adicionales.

Características de la tarjeta BASYS2

Antes de realizar cualquier programación, se debe conocer el dispositivo en el que se programará.

En primer lugar, la fuente de poder de la tarjeta Basys2, procede de su cable USB, por el cual se realiza también la comunicación JTAG. Pero ésta tarjeta también posee una conexión para agregarle una fuente de poder externa en caso de ser necesario.

Para usar la fuente de poder del puerto USB, se debe conectar el cable en el puerto mini-USB para obtener energía de allí., Necesario tomar en consideración que para usar la fuente de poder externa, ésta debe estar en el rango de los 3.5V a 5.5V. Voltajes mayores a los 5.5V pueden generar daños permanentes en la tarjeta. La tarjeta posee un regulador interno encargado de convertir la tensión de entrada en los 3.3V de suministro principal y en las tensiones de 2.5V y 1.2V de la FPGA.

La tarjeta posee además, ocho interruptores de entrada, que dependiendo de la posición de los interruptores, puede variar su señal de entrada a un valor alto o bajo, permaneciendo constante, mientras se mantenga la misma posición. En la Figura 2.1 se presenta el diagrama circuital de la conexión interna de la Basys2, con sus respectivas resistencias para prevenir cortocircuitos.

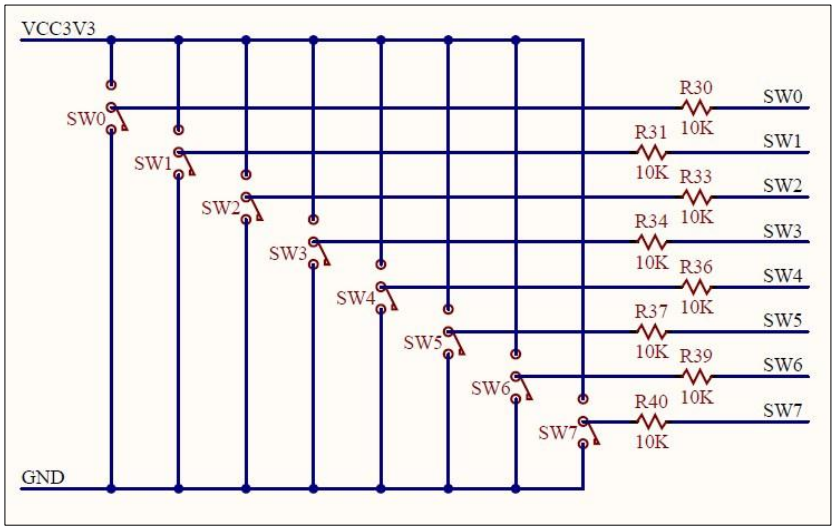


Figura 2.1 Circuito de conexión de los interruptores de entrada
Fuente: Manual Esquemático de conexiones de la tarjeta Basys2 [7].

Es significativo destacar que en las figuras mostradas se encuentran referidas las identificaciones que se le da a cada pin de la FPGA para hallarle en la tarjeta en físico y de esta manera poder ubicarlos con facilidad.

Otra característica importante de la tarjeta son los cuatro pulsadores de entrada que están normalmente en su valor bajo y luego de ser presionados pasan a su valor alto (uno lógico); con la configuración mostrada en la Figura 2.2.

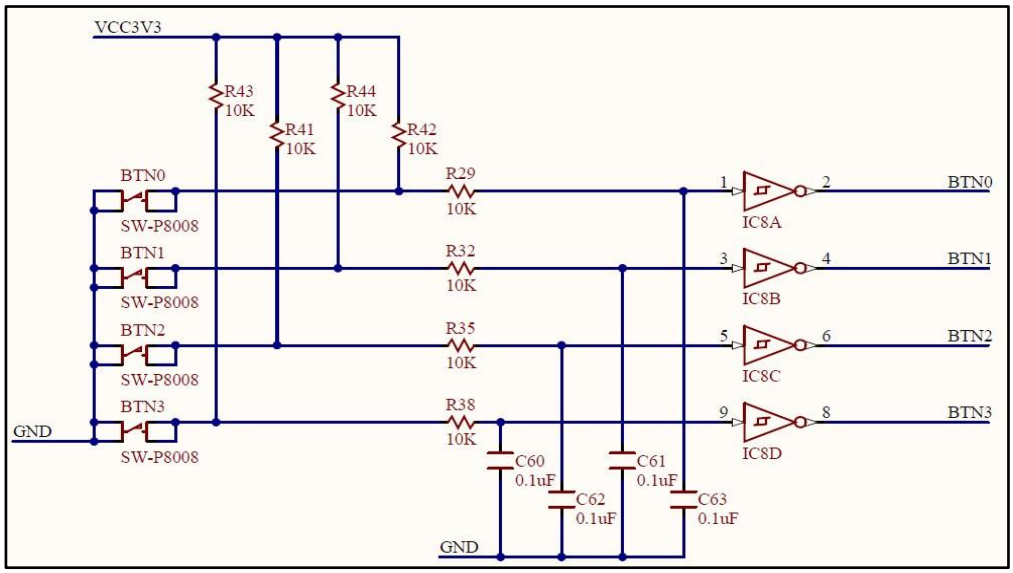


Figura 2.2 Circuito de conexión de los pulsadores de entrada

Fuente: Manual Esquemático de conexiones de la tarjeta Basys2 [7].

Adicionalmente cuenta con ocho indicadores luminosos Leds, como dispositivos de salida, que cuentan con sus resistencias limitadoras de corriente. Los ánodos de los Leds están conectados a resistencias limitadoras de corriente, de tal manera que solo se encenderán cuando un "uno lógico" sea enviado como salida por ese pin de la FPGA (Figura 2.3).

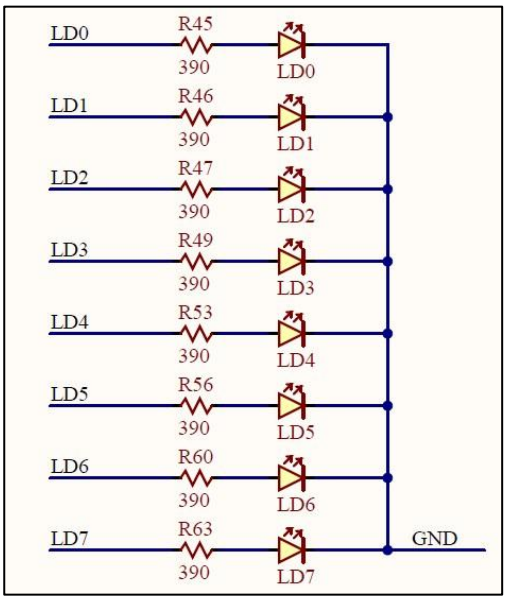


Figura 2.3 Circuito de conexión de los leds de entrada.

Fuente: Manual Esquemático de conexiones de la tarjeta Basys2 [7].

Cuatro displays de 7-Segmentos de ánodo común, disponibles como dispositivos de salida, están compuestos por siete Leds identificados individualmente, cada uno con una letra y distribuidos en un patrón como se muestra en la Figura 2.4.

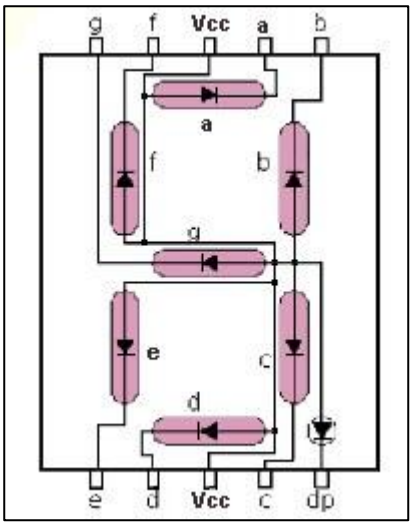


Figura 2.4 Disposición de los Les del display de 7-Segmentos.

Presenta un transistor de colector abierto que trasmite la señal AN a la base del transistor PNP que conecta a la fuente Vcc a los dislpays. Cada display se ilumina con un ‘0’ en la base de su transistor AN(i), para habilitar cada uno de ellos, conectados como se muestra en la Figura 2.5. Hay un bus de dato por cada señal de los Leds del siete segmentos, este bus simple identificado desde CA, hasta CG incluyendo DP sirve para transmitir la señal de activo bajo a los Leds de todos los displays 7-segmentos.

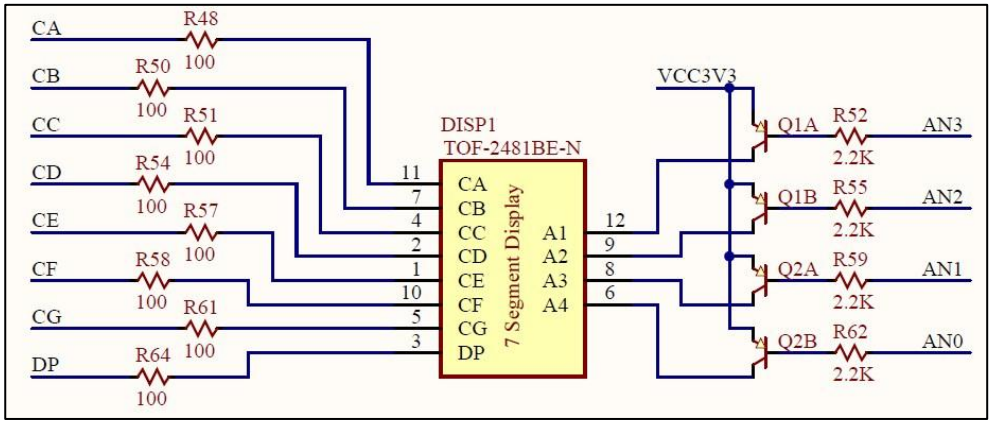


Figura 2.5 Circuito de conexión de los pulsadores de entrada.
Fuente: Manual Esquemático de conexiones de la tarjeta Basys2 [7].

Que sean de ánodo común, significa que los ánodos de los sietes Leds están conectados a un punto en común, mientras que los cátodos quedan separados. Los cátodos de todos los segmentos similares de los 4 displays, también están conectados juntos, identificados con las señales desde CA hasta CG e incluyendo DP.

Del AN0 al AN3 se controlan los ánodos individuales de los displays 7-segmentos, proporcionando una señal de potencia Vcc para activar el transistor. De CA al CG, se controlan los segmentos de los cuatro display 7-Segmentos. DP controla el punto decimal.

La tarjeta Basys2 no realiza por si sola la comunicación entre sus puertos, pero si identifica cada una de las variables relevantes para lograr la comunicación. Presenta conexiones como las mostradas en la Figura 2.6 en las que se muestra las señales

identificadas como RED, GRN y BLU para emitir una señal de video del color deseado, HSYNC, VSYNC para reconocer los flancos de sincronización correspondientes.

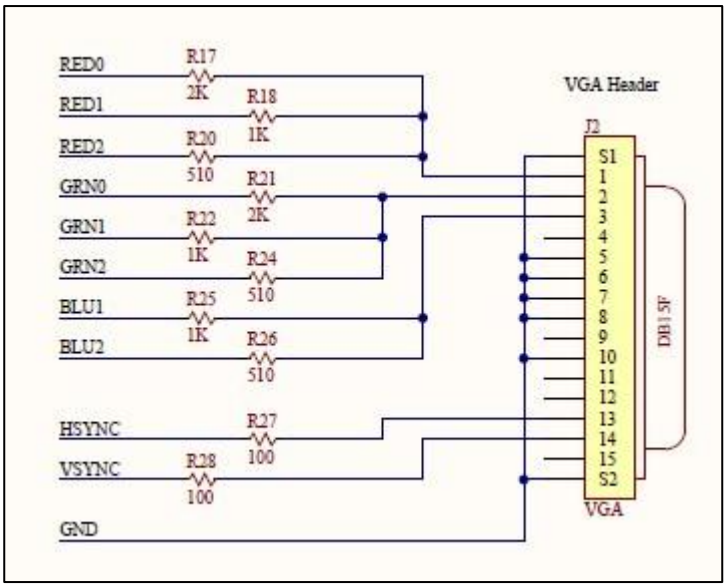


Figura 2.6 Circuito de conexión del puerto VGA.

Fuente: Manual Esquemático de conexiones de la tarjeta Basys2 [7].

Con el puerto PS/2 presenta conexiones como las mostradas en la Figura 2.7 en las que se muestra las señales identificadas como PS2D, PS2C para emitir señales de buses de datos y de reloj del teclado respectivamente.

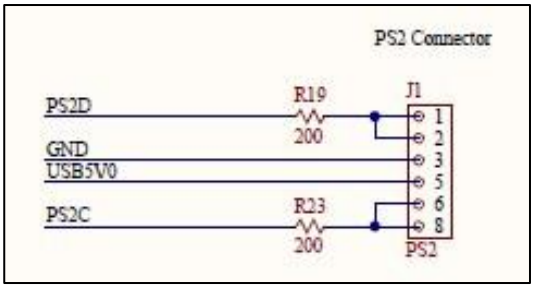


Figura 2.7 Conexión interna del puerto PS/2.

Fuente: Manual Esquemático de conexiones de la tarjeta Basys2 [7].

Con este puerto se puede conectar tanto un mouse como un teclado, ya que trabajan con los mismos buses de datos.

Los tres pines que se muestran como JP4 en la Figura 2.8 son los que se encuentran en físico de la tarjeta BASYS2, para ajustar la frecuencia del oscilador interno entre 25, 50 y 100 MHz, dividiendo ésta y transmitiéndola mediante la señal DIV, Aunque este jumper no está instalado inicialmente y requiere ser soldado, se encuentra disponible de ser necesario.

En la misma figura se muestra el oscilador LTC6905 identificado como IC5 en la tarjeta lógico programable. Posee capacitores para filtrar la señal y un circuito retroalimentado (síncrono). Enviando la señal de salida del reloj interno por la señal MCLK. Disponible en el manual esquemático de conexiones de la tarjeta BASYS2 [7].

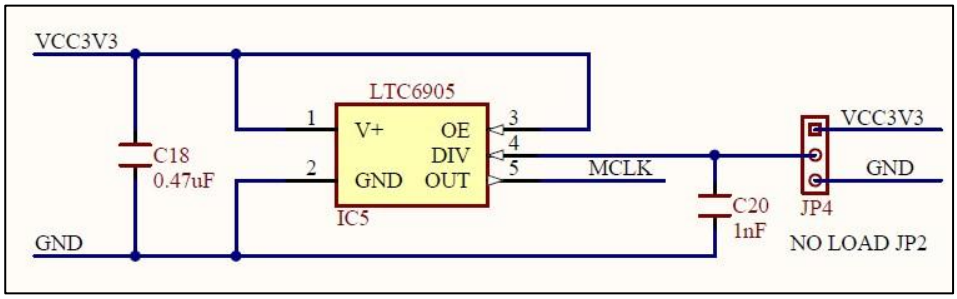


Figura 2.8 Conexión interna del Oscilador interno.
Fuente: Manual Esquemático de conexiones de la tarjeta Basys2 [7].

Los pines del SG8002 identificado en la tarjeta BASYS2 como IC6, como se muestra en la Figura 2.9, sirven de adaptadores de comunicación entre un posible oscilador externo y la FPGA, cuya salida se emite mediante la señal UCLK. Estos pines están diseñados para soportar cualquier oscilador externo CMOS de 3.3V.

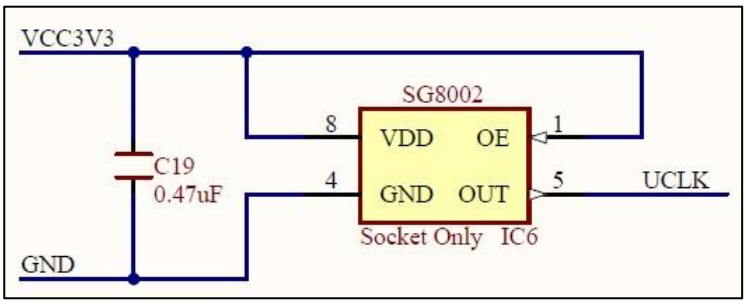


Figura 2.9 Conexión interna del Socket IC6.
Fuente: Manual Esquemático de conexiones de la tarjeta Basys2 [7].

En aplicaciones con uso del puerto VGA de la FPGA Basys2 usando su oscilador interno, la señal en el monitor resulta inestable, debido a que este oscilador no presenta la misma calidad en su señal que un oscilador de cristal, como lo explica el manual de usuario.

Está dotado de dos Leds adicionales, que sirven de indicadores, el primero notifica que la FPGA tiene energía, ya sea desde la conexión USB o de una fuente externa, el otro indicador, denominado LD-D en la FPGA, sirve para informar al usuario cuando se está programando en la tarjeta, que queda parpadeando y luego se mantiene encendido en indicación de que la FPGA fue programada y está lista para usarse. En la Tabla 2.1 se muestra la identificación completa de los pines de la Basys2

Basys2 Spartan-3E pin definitions											
Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
C12	JD1	P11	SW0	N14	CC	B2	JA1	P8	MODE0	M7	GND
A13	JD2	M2	USB-DB1	N13	DP	C2	USB-WRITE	N7	MODE1	P5	GND
A12	NC	N2	USB-DB0	M13	AN2	C3	PS2D	N6	MODE2	P10	GND
B12	NC	M9	NC	M12	CG	D1	NC	N12	CCLK	P14	GND
B11	NC	N9	NC	L14	CA	D2	USB-WAIT	P13	DONE	A6	VDDO-3
C11	BTN1	M10	NC	L13	CF	L2	USB-DB4	A1	PROG	B10	VDDO-3
C6	JB1	N10	NC	F13	RED2	L1	USB-DB3	N8	DIN	E13	VDDO-3
B6	JB2	M11	LD1	F14	GRN0	M1	USB-DB2	N1	INIT	M14	VDDO-3
C5	JB3	N11	CD	D12	JD4	L3	SW1	P1	NC	P3	VDDO-3
B5	JA4	P12	CE	D13	RED1	E2	SW6	B3	GND	M8	VDDO-3
C4	NC	N3	SW7	C13	JD3	F3	SW5	A4	GND	E1	VDDO-3
B4	SW3	M6	UCLK	C14	RED0	F2	USB-ASTB	A8	GND	J2	VDDO-3
A3	JA2	P6	LD3	G12	BTN0	F1	USB-DSTB	C1	GND	A5	VDDO-2
A10	JC3	P7	LD2	K14	AN2	G1	LD7	C7	GND	E12	VDDO-2
C9	JC4	M4	BTN2	J12	AN1	G3	SW4	C10	GND	K1	VDDO-2
B9	JC2	N4	LD5	J13	BLU2	H1	USB-DB6	E3	GND	P9	VDDO-2
A9	JC1	M5	LD0	J14	HSYNC	H2	USB-DB5	E14	GND	A11	VDDO-1
B8	MCLK	N5	LD4	H13	BLU1	H3	USB-DB7	G2	GND	D3	VDDO-1
C8	RCCLK	G14	GRN2	H12	CB	B14	TMS	H14	GND	D14	VDDO-1
A7	BTN3	G13	GRN1	J3	JA3	B13	TCK-FPGA	J1	GND	K2	VDDO-1
B7	JB4	F12	AN0	K3	SW2	A2	TDO-USB	K12	GND	L12	VDDO-1
P4	LD6	K13	VSYNC	B1	PS2C	A14	TDO-S3	M3	GND	P2	VDDO-1

Tabla 2.1 Identificación de los pines de la Basys2.
Fuente: Manual de Referencia de la tarjeta Basys2 [3].

2.2.2 LENGUAJE VHDL

En la actualidad, el lenguaje de descripción en hardware más utilizado a nivel industrial es VHDL (Hardware Description Language), la cual apareció en la década de los ochenta como un lenguaje estándar, capaz de soportar el proceso de diseño de sistemas electrónicos complejos, con propiedades para reducir el tiempo de diseño y los recursos tecnológicos requeridos. El departamento de Defensa de Estados Unidos creó el lenguaje VHDL como parte del programa "Very High Speed Integrated Circuits" (VHSIC), a partir del cual se detectó la necesidad de contar con un medio estándar de comunicación y la documentación para analizar la gran cantidad de datos asociados para el diseño de dispositivos de escala y complejidad deseados [8].

El IEEE (Instituto de Ingenieros Eléctricos y electrónicos) publicó en diciembre de 1987 el estándar IEEE std 1076-1987. Un año más tarde, surgió la necesidad de describir en

VHDL todos los ASIC creados por el Departamento de Defensa, por lo que en 1993 se adoptó el estándar adicional de VHDL IEEE 1164.

Actualmente VHDL es considerado como un estándar para la descripción, modelado y síntesis de circuitos digitales y sistemas complejos. Además presenta diversas características que lo hacen uno de los HDL más utilizados. A continuación se presentan en vista general las Ventajas del uso de VHDL para la descripción de hardware [9].

a. VHDL permite diseñar, modelar, y comprobar un sistema desde un alto nivel de abstracción bajando hasta el nivel de definición estructural de compuertas.
b. Los módulos creados en VHDL pueden utilizarse en diferentes diseños, lo que permite la reutilización del código. Además, la misma descripción puede utilizarse de diferentes tecnologías sin tener que rediseñar todo el circuito.
c. Por estar basados en un estándar (IEEE STD 1076-1987) los ingenieros de toda la industria de diseño pueden usar este lenguaje para minimizar errores de comunicación y problemas de compatibilidad.
d. VHDL admite diseño Top-Down, lo que consiste en describir (modelado) el comportamiento de los bloques de alto nivel, analizándolos (simulación), y refinar la funcional de alto nivel requerida antes de llegar a niveles más bajos de abstracción de la implementación de diseño.
e. Modularidad: VHDL permite dividir o descomponer un diseño hardware y su descripción VHDL en unidades más pequeñas.

VHDL es un lenguaje portable y reusable ya que es independiente de la tecnología o fabricante (Xilinx, Altera, Actel, QuickLogic, etc.). Sus sentencias a diferencia de un programa de software se ejecutan inherentemente en forma “concurrente” salvo aquellas que se incluyan dentro de un procedimiento (Procedure), Proceso (Process) o Función (Function) donde se ejecutaran en forma secuencial.

Estructura de un programa VHDL

Las secciones fundamentales que forman el código VHDL son: Librería (LIBRARY), entidad (ENTITY) y arquitectura (ARCHITECTURE) como se muestra en la Tabla 2.2.

Elemento	Descripción
LIBRARY	Contiene un conjunto de librerías que contienen la definición de datos, estructuras, etc.
ENTITY	Especifica las E/S del circuito.
ARCHITECTURE	Existen dos formas de describir un circuito, estructural y por su comportamiento.

Tabla 2.2 Secciones fundamentales del VHDL.

Una entidad está compuesta por tres elementos (Tabla 2.3).

Elemento	Características
Puertos de entradas y salida.	Cada una de las señales de entrada y salida en una entidad, son llamadas puertos. Los puertos deben poseer nombre modo y tipo de dato.
Modos	Modo in. Modo out. Modo inout (puerto bidireccional con retroalimentación dentro o fuera de la entidad).
Tipos de datos	Std_logic. Boolean. Std_logic_vector (vectores de bits). Integer.

Tabla 2.3 Elementos de una entidad en VHDL.

Ejemplo Full-Adder

En este ejemplo se incluye un sumador completo (Figura 2.10), en el que se muestra su entidad, seguido de los puertos de entrada o salida y el tipo de variable. Luego de esto, en arquitectura se establece la operación a realizar en este caso un Or especial de "x" con "y" y "Cin" todos de tipo std_logic quiere decir un solo bit. Asignando el valor de la suma a "s" y el valor del acarreo en Cout.

```
use LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY fulladd IS
    PORT (
    Cin, x, y     : IN     STD_LOGIC ;
    s, Cout            : OUT     STD_LOGIC ) ;
END fulladd ;

ARCHITECTURE suma OF fulladd IS
BEGIN
    s <= x XOR y XOR Cin ;
    Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y) ;
END suma ;
```

Figura 2.10 Ejemplo Full Adder.

Resumen de aspectos más importantes del lenguaje

Estructural: Cada módulo se divide a su vez en sub-módulos que se conectan entre sí, por medio de señales. Enumera los componentes y las interconexiones entre ellos.

Comportamental: se describe cómo se comporta el sistema. Describe la funcionalidad del sistema.

Un proyecto de VHDL puede contener muchos ficheros. El código VHDL usualmente se encuentra en los ficheros con extensión *.vhd. La estructura de uno de estos ficheros es:

- Llamadas a librerías.
- Entidad.
- Arquitectura(s).

Elementos Sintácticos del VHDL

El lenguaje VHDL tiene sus elementos sintácticos, sus tipos de datos y sus estructuras como la mayoría de los lenguajes de programación o descripción existentes. El hecho de que sirva para la descripción de hardware lo hace ya diferente a los lenguajes convencionales, pues permite ejecutar instrucciones a la vez de forma concurrente. Algunos de estos elementos sintácticos son (Tabla 2.4).

Comentarios	**Cualquier línea que empieza con dos guiones cortos "- -", es un comentario.**
Identificadores	Son todos aquellos caracteres que pueden servir para identificar variables, señales, nombres de rutina, entre otros. Pueden ser cualquier conjunto de caracteres formados por letras, incluyendo el guion bajo, Las mayúsculas y minúsculas son consideradas iguales. No puede haber identificador que coincida con alguna de las palabras clave del VHDL.
Números	Cualquier número se considera que se encuentra en base 10. Se admite la notación científica convencional para números en coma flotante. Es posible poner números en otras bases utilizando el símbolo del

	sostenido "#" ejemplo: 2#11000100# y 16#C4# representan el entero 196.
Caracteres	Es cualquier letra o carácter entre comillas simples 'l',3','t'.
Cadenas	Es cualquier letra o carácter entre comillas dobles: "Esto es una cadena".
Cadena de bits	Los tipos bit y bit vector son en realidad de tipo carácter y matriz de caracteres respectivamente. En VHDL se tiene una forma elegante de definir números con estos tipos y es mediante la cadena de bits. Dependiendo de la base en que se especifique el número se puede poner un prefijo B (binario), O (octal), o X (hexadecimal). Ejemplo B"100101101".

Tabla 2.4 Elementos sintácticos del VHDL.

Operadores y expresiones

En VHDL, se mantiene el mismo formato para las expresiones como en otros lenguajes de programación o descripción existentes, para mantener la simplicidad y evitar que el usuario deba iniciar de cero. Por ello se explicaran únicamente los más resaltantes (Tabla 2.5).

Operadores	
Operadores varios	& (Concatenación) Concatena matrices de manera que la dimensión de la matriz resultante es la suma de las dimensiones de las matrices sobre las que se opera.
Operadores aritméticos	** (Exponencial) sirve para elevar un número a una potencia 3**2 equivale a 3^2. El operador de la izquierda puede ser entero o real, pero el de la derecha únicamente puede ser entero.
	ABS() (Valor absoluto). Devuelve el valor absoluto de su argumento que puede ser de cualquier tipo numérico.
	* (Multiplicación) Sirve para multiplicar dos número de cualquier tipo (los tipos bit o vectores de bits no son numéricos).
	/ (División) También funciona con cualquier dato de tipo numérico.
	- (Resta y signo negativo) Cuando va entre dos operandos se realiza la operación de sustracción y si va delante de una expresión le cambia el signo. Los operandos pueden ser numéricos de cualquier tipo.
	+ (Suma y signo positivo) Este operador sirve para indicar suma, si va entre dos operandos, o signo, si va al principio de una expresión. La precedencia es diferente en cada caso. Opera sobre valores numéricos de cualquier tipo.
Operadores relacionales	Se encargan de devolver un valor de tipo booleano (TRUE o FALSE). Los tipos con los que pueden operar dependen de la operación.

	=, /= (igualdad) El primero devuelve TRUE si los operandos son iguales y FALSE en caso contrario. El segundo indica desigualdad y funciona al revés de la igualdad. Los operandos pueden ser de cualquier tipo, pero ambos operandos deben ser del mismo.
Operadores lógicos	Entre los operadores lógicos se tienen, NOT, AND, NAND, OR, NOR y XOR. El funcionamiento es el habitual para este tipo de operadores aunque los tipo bit, vectores de bit y boolean. En el caso de realizarse estas operaciones sobre un vector, la operación se realiza bit a bit, incluyendo la operación NOT.

Tabla 2.5 Operadores del VHDL

Para profundizar en la lectura sobre el lenguaje de descripción de hardware se recomienda la lectura de Pardo F, Boluda [9].

2.2.3 ACTIVE HDL STUDENT EDITION

En este caso se cuenta con la edición estudiantil, gratuita que ofrece Aldec. que contiene tanto la plataforma del lenguaje de diseño, como la herramienta de simulación. Esta herramienta combina el Schematics, el lenguaje de descripción de hardware Verilog y Simulación en un paquete, permitiendo diseñar en diferentes niveles de abstracción e interfaces a una variedad de herramientas de implementación para FPGAs y ASICs

En cuanto a la licencia, es gratuita, por lo que el estudiante solo debe registrarse y descargar el software, el cual tendrá una licencia que será válida por aproximadamente un año. Las partes principales de Active-HDL son:

- El browser de diseño
- El explorador de diseño
- Editor de VHDL

- Editor de máquinas de estado
- Editor de forma de onda
- Consola
- Administrador de librería
- Flujo de datos
- Listado
- Pila de llamadas
- Procesos
- Señales de observación

2.2.4 ADEPT DIGILENT

Adept es un programa creado por DIGILENT Inc. que se utiliza para transferir al FPGA el archivo .bit almacenado y creado por el software de programación. Adept utiliza un cable USB o JTAG para transferir archivos desde la PC a la tarjeta que contiene el FPGA (a través del puerto del FPGA JTAG de programación). Adept puede además programar un archivo en la ROM no volátil llamada plataforma Flash de la tarjeta. El archivo de configuración se enviara al FPGA o plataforma Flash y el software le indicaran si la programación se ha realizado correctamente. El mensaje de configuración "LED" (LD_D) se ilumina después de que el FPGA se ha configurado correctamente [10].

2.2.5 ARQUITECTURA SPARTAN 3-E FPGA

Cuatro conectores de expansión estándar permiten a la tarjeta Basys2 (Figura 2.11) crecer utilizando circuitos diseñados por el usuario o PMods. (Los PMods son módulos de E/S analógicos y digitales de bajo costo que ofrecen conversión A/D y D/A, drivers para motor, entradas de sensor y muchas otras características). Contiene un cable USB que le proporciona energía y es utilizado como interfaz de programación, por lo que ninguna otra fuente de poder o cable de programación es requerido [3].

- Características Generales

- Xilinx Spartan 3-E FPGA, de 250.000 compuertas.

- FPGA con multiplicadores de 18-bit, 72Kbits de bloque RAM dual-port, y 500MHz adicionales a la velocidad de operación.

- Puerto USB2 full-speed para la configuración y transferencia de datos hacia el FPGA (utilizando el software Adept 2.0 disponible en descarga gratuita).

- XCF02 Plataforma Flash ROM que almacena las configuraciones del FPGA.

- Frecuencia de oscilación ajustable (25, 50, y 100 MHz), además cuenta con socket para un oscilador extra.

- Reguladores de voltaje incluidos (1.2V, 2.5V, y 3.3V) que permiten el uso de fuentes externas de 3.5V a 5.5V.

- 8 indicadores LED, 4 displays de 7-Segmentos, 4 pulsadores, 8 interruptores, Puerto PS/2, y un puerto VGA de 8-bit.

- Cuatro conectores de 6-pines para Entradas y Salidas del usuario, y compatibles con los circuitos Digilent PMOD.

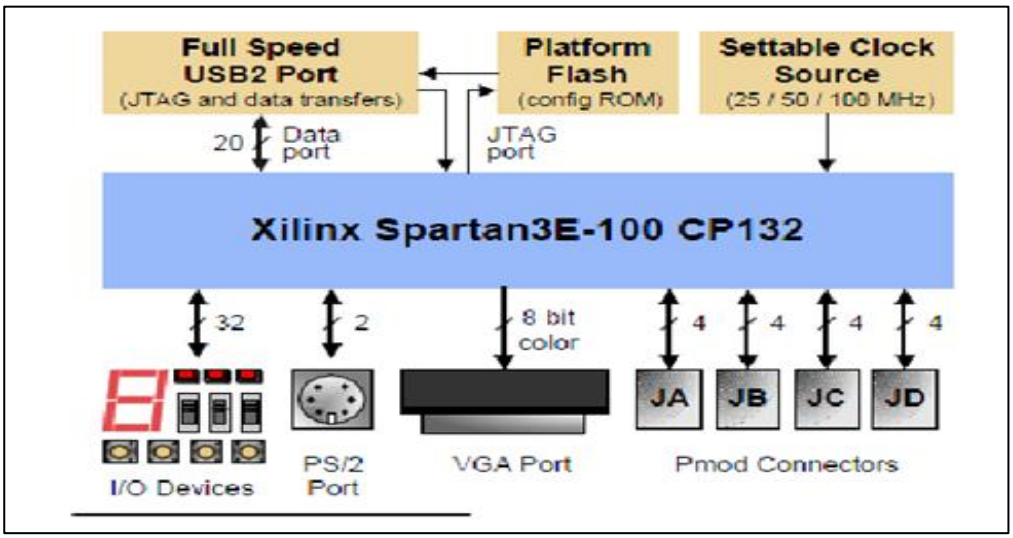
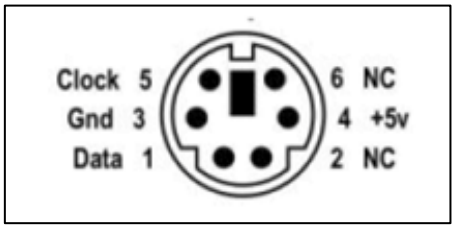


Figura 2.11 Diagrama a bloques Tarjeta Basys2.

Fuente: Manual de referencia Basys2 [3].

2.2.6 PUERTO PS/2

El conector PS/2 o puerto PS/2 posee 6 pines, que es del tipo mini-DIN (Figura 2.12). Adicionalmente al pin alimentación y al pin de tierra, también está un pin de reloj bidireccional y un pin de datos bidireccional. El puerto PS/2 es típicamente conectado entre un dispositivo de entrada como un teclado o un ratón, con respecto a una computadora o la FPGA.



1.- Data (Datos) 2.- Reservado
3.- GND (Tierra) 4.- +5V VDC
5.- Clock (Reloj) 6.- Reservado

Figura 2.12 Pines del puerto PS/2.

Fuente: Haskell R., Hanna D. (2010) [6]

En la mayor parte del tiempo los datos son transferidos del dispositivo al host, en el caso de la FPGA, esta comunicación es denominado device-to-host (Dispositivo al

anfitrión). Si el anfitrión necesita enviar un comando al dispositivo, usa la comunicación host-to-device (Anfitrión al dispositivo) [6].

El dispositivo de entrada siempre generará la señal de reloj independientemente de la dirección del flujo de datos. La frecuencia del reloj tiene que estar en el rango de 10 a 16.7 kHz [6].

En la Figura 2.13, se muestra la comunicación Device-to-host, solo tiene dos líneas, la de datos y la del reloj. El dispositivo de entrada siempre generará la señal de reloj independientemente de la dirección de flujo de datos, la frecuencia del reloj va a estar en un rango entre 10 y 16.7 kHz. También podemos observar que en la línea de datos se envían tramos de 11 bits, teniendo un bit de inicio, 8bits de dato, un bit de paridad y un bit de fin. El bit de paridad estará a uno si hay un número par de unos en los 8 bits de datos, y estará a cero si hay un número impar. Por tanto, contando los 8 bits de datos más el bit de paridad, deberá haber un número impar de unos (por eso es paridad impar). Esto se utiliza para detectar errores.

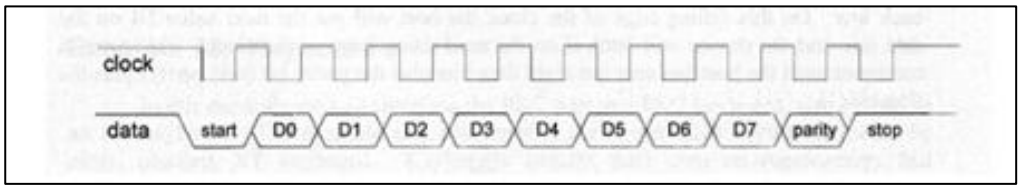


Figura 2.13 Señales enviadas desde el teclado con puerto PS/2.
Fuente: Haskell R., Hanna D [6].

El anfitrión debe realizar la lectura de la señal de datos en los flancos de bajada de la señal del reloj, así como se observa en la Figura 2.14.

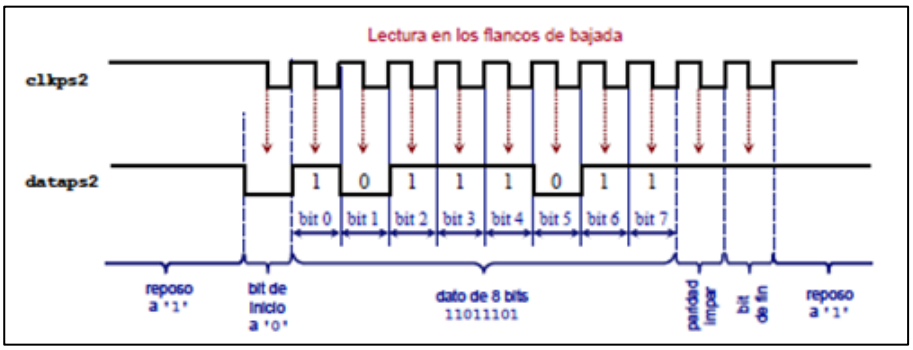


Figura 2.14 Cronograma para una transmisión del teclado.
Fuente: Maxinez, D y Alcalá, J [11].

2.2.7 VIDEO GRAPHICS ARRAY

La sigla VGA proviene de ("Video Graphics Array ó Video Graphics Adapter"), lo que traducido significa arreglo gráfico de video o adaptador gráfico de video. Se trata de un conector semi-trapezoidal con 15 terminales, que se encarga de enviar las señales referentes a los gráficos desde la computadora hasta una pantalla para que sean mostrados al usuario. Por el hecho de permitir la transmisión de datos hacia un dispositivo externo (periférico), desde la computadora, se le denomina puerto.

Lo más básico que se debe saber acerca del VGA es que es un protocolo para ser usado en principio análogo con los dispositivos de salida del CRT (tubo de rayos catódicos). Sin embargo, los monitores digitales se han adaptado al mismo estándar para evitar problemas de compatibilidad. De cualquier forma, en estos dispositivos la pantalla mostrará los pixeles sobre la pantalla de izquierda a derecha y de arriba hacia abajo, para generar una imagen completa o "frame", refrescando la pantalla de acuerdo a la resolución, y además definiendo el color y la intensidad de cada pixel.

La estandarización de estas señales comprende a la norma VGA (Video Graphics Array), la cual tiene como estándar 640 x 480 pixeles en video activo, utilizada para los monitores CRT y vigente para los monitores LCD [13l].

Características del puerto VGA

- En el ámbito de la electrónica comercial se le denomina como conector DB9 ("D-subminiatura tipo B, 15 pin"), esto es D-subminiatura tipo B, con 15 pines.

- El puerto VGA se encarga de enviar las señales desde la computadora hacia la pantalla con soporte de 256 a 16.7 millones de colores y resoluciones desde 640x480 píxeles en adelante.

- Puede estar integrado directamente en la tarjeta principal (Tarjeta Madre), en una tarjeta de video/tarjeta aceleradora de gráficos.

Pines del puerto VGA y tarjeta Basys2

El puerto VGA está compuesto por cinco diferentes señales, dos señales de sincronización (HSYNC y VSYNC) y las tres señales de video (R, G, B) como se muestra en la Figura 2.15. Con mayor detalle se muestra en la Tabla 2.6.

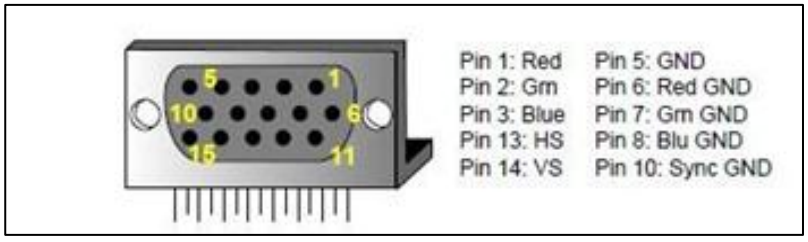


Figura 2.15 Definición de pines del puerto VGA y de la tarjeta Basys2.
Fuente: Manual de referencia de la tarjeta Basys2 [3].

Pin	Nombre	Descripción
1	RED	Señal de vídeo rojo
2	GREEN	Señal de vídeo verde
3	BLUE	Señal de vídeo azul
4	RES	Reservado
5	GND	Tierra
6	RGND	Tierra de la señal rojo
7	GGND	Tierra de la señal verde
8	BGND	Tierra de la señal azul
9	+5V	+5 VDC
10	SGND	Tierra de la señal Sincronismo
11	ID0	Monitor ID Bit 0 (Opcional)
12	SDA	DDC línea de dato serial
13	HSYNC	Señal de Sincronización Horizontal
14	VSYNC	Señal de Sincronización Vertical
15	SCL	DDC línea de dato del reloj

Tabla 2.6 Leyenda de los pines del puerto VGA.

La tarjeta Basys2 usa diez señales de la FPGA para crear un puerto VGA con 8-bits de color y dos señales de sincronización (HS - Señal de sincronización horizontal y VS - Señal de sincronización vertical). Las señales de vídeo de color usan circuitos de divisores de tensión que funcionan como una conjunción con la resistencia terminal de 75 ohmios del display VGA, para crear diferentes niveles de tensión entre las señales de video rojo, verde y azul (Figura 2.16). Estos niveles de tensión van desde 0V (fully off) hasta 0,7V (fully on).

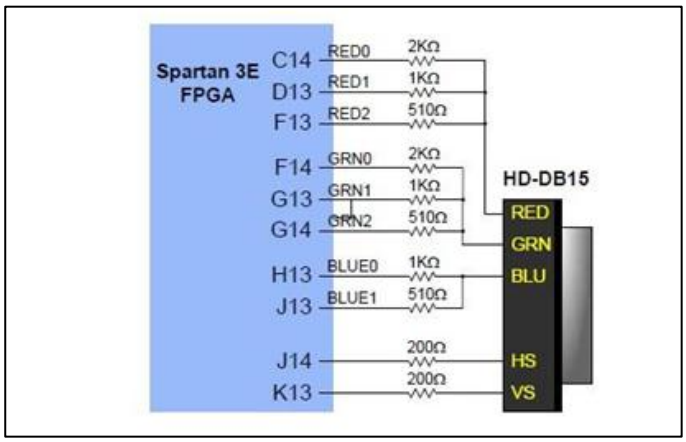


Figura 2.16 Definición de pines del puerto VGA y de la tarjeta Basys2.
Fuente: Manual de referencia de la tarjeta Basys2 [3].

Como las señales de video R, G y B de entrada hacia al monitor son señales analógicas, las salidas desde la FPGA que son digitales necesitan convertirse en señales analógicas. Que es de lo que se encarga el convertidor D/A que posee la tarjeta Basys2, mediante tres circuitos de divisores de tensión (Figura 2.17). Para convertir tres bits se la señal de video R en ocho posibles combinaciones y por ello, ocho niveles de intensidad de señal analógica VR.

De igual forma con las demás señales de video, teniendo tres bits de la señal de video RED, tres bits de la señal de video GREEN y dos bits de la señal de video BLUE. Se obtienen 256 colores diferentes.

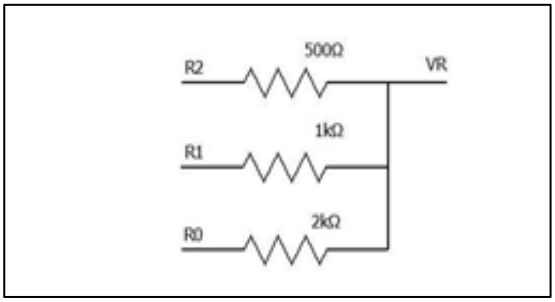


Figura 2.17 Arreglo de resistencias de un color.

Partiendo del circuito de la Tabla 2.7 y asumiendo que la corriente es nula, se obtiene la ecuación (1):

$$VR = (4/7)\,R2 + (2/7)\,R1 + (1/7)\,R0 \qquad (1)$$

R2	R1	R0	VR	Color resultante
0	0	0	0	Negro
0	0	1	1/7	Azul
0	1	0	2/7	Verde
0	1	1	3/7	Cyan
1	0	0	4/7	Rojo
1	0	1	5/7	Fucsia
1	1	0	6/7	Amarillo
1	1	1	7/7	Blanco

Tabla 2.7 Tabla de combinaciones para la obtención de colores.

Funcionamiento del estándar VGA

Luego de haber ilustrado los puertos de los dispositivos involucrados y la manera como se generan la diversa gama de colores ahora, se enfocará en el protocolo de comunicación del puerto VGA. Para ello, se basó en las siguientes fuentes de información: Manual de referencia de la tarjeta Basys2 [3] y Haskell y Hanna [6].

La señal de sincronización horizontal y vertical está definida en la señal diente de sierra la cual es creada a partir de la señal sinusoidal de 60hz. En la señal diente de sierra se pueden distinguir tres partes principales, las cuales están descritas en la Figura 2.18.

La parte principal de la señal corresponde a la rampa continua, en la cual se despliega la información en la pantalla. Abarcando los 640 pixeles para el modelo estándar. Luego de

esta etapa ya no se despliega información en la pantalla, pero el desplazamiento continuo, se conoce como "regreso", definida esta etapa como "back porch" en la que sigue moviéndose de izquierda a derecha hasta que la señal de sincronización vertical provoque que se regrese al lado izquierdo de la pantalla pero modificando la posición una fila hacia abajo.

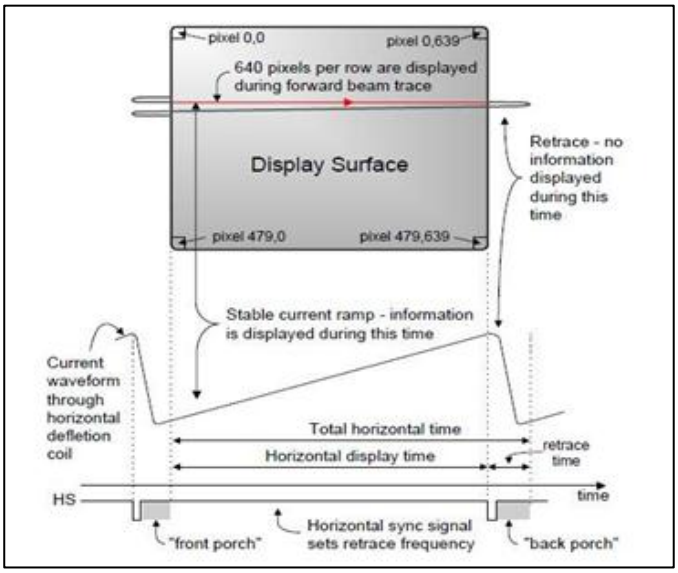


Figura 2.18 Protocolo de comunicación con el puerto VGA.
Fuente: Manual de referencia de la tarjeta Basys2 [3].

En la siguiente parte denominada "front porch" tampoco se despliega información hacia la pantalla, pero se sigue desplazando hacia la derecha hasta que consiga la rampa continua que le permita el despliegue de los 640 pixeles de información de esa línea.

CAPITULO III: MARCO METODOLÓGICO

3.1 DISEÑO Y TIPO DE INVESTIGACIÓN

De acuerdo a los objetivos planteados en el presente Proyecto de Grado, este cumplió con las características de una investigación de tipo factible y especial. Según el manual de trabajos de grado de especialización y maestría y tesis doctorales de la Universidad Pedagógica Experimental Libertador declara que un proyecto factible *"consiste en la investigación, elaboración y desarrollo de una propuesta de un modelo operativo viable para solucionar problemas, requerimientos o necesidades de organizaciones o grupos sociales; puede referirse a la formulación de políticas, programas, tecnologías, métodos o procesos" (2006 – P.13).*

Por otra parte el proyecto es especial ya que:

Consiste en trabajos que lleven a creaciones tangibles, susceptibles de ser utilizadas como soluciones a problemas demostrados, o que respondan a necesidades e intereses de tipo cultural. Se incluyen en esta categoría los trabajos de elaboración de libros de texto y de materiales de apoyo educativo, el desarrollo de software, prototipos y de productos tecnológicos en general, así como también los de creación literaria y artística (2006 – P.14).

3.2 PROCEDIMIENTO METODOLÓGICO UTILIZADO

Para el desarrollo del proyecto, en cuanto a la metodología de la investigación se dividió en seis fases comprendidas entre el cumplimiento de los objetivos planteados y culminación del proyecto.

3.2.1 IDENTIFICACIÓN DE LIMITACIONES DE LA TARJETA LÓGICO PROGRAMABLE BASYS2

En esta etapa se llevó cabo, la búsqueda y recolección de todos los documentos necesarios, para comprender y escoger los elementos a utilizar, en ese sentido, se comprende las siguientes etapas fundamentales:

- Investigar, recopilar y seleccionar la información requerida para determinar las características de la tarjeta lógica programable Basys2. Entre ellos, investigar sobre la composición del Basys2, sus entradas y salidas, y sus características más resaltantes.

- Determinar la capacidad y limitaciones que presenta la tarjeta lógica programable Basys2 actualmente en el laboratorio de Lógica Digital para el desarrollo de este proyecto.

3.2.2 DESCRIPCIÓN DEL SISTEMA MEDIANTE EL LENGUAJE VHDL

En esta etapa se llevó a cabo la realización y organización de la estructura de los bloques a utilizar, a fin de plantear adecuadamente el diseño del sistema, de acuerdo a las siguientes etapas:

- Definir la función de cada uno de los bloques del sistema.

- Describir mediante VHDL la definición de las entidades a utilizar para la realización del sistema.

3.2.3 CÓDIGO DE PROGRAMACIÓN PARA LOGRAR UNA COMUNICACIÓN ENTRE LA TARJETA BASYS2 Y EL TECLADO

En esta etapa, se procedió de la siguiente manera:

- Identificar el comportamiento del protocolo bidireccional serie síncrono de 11 bits.

- Realizar un diseño que reciba los datos del teclado y de esta manera ejecute la comunicación de la tarjeta lógica programable Basys2 con el teclado con puerto PS/2.

- Establecer un bloque compacto de este diseño de comunicación del teclado para su uso en aplicaciones del laboratorio.

3.2.4 CÓDIGO DE PROGRAMACIÓN PARA LOGRAR UNA COMUNICACIÓN ENTRE LA TARJETA BASYS2 Y EL MONITOR

En esta etapa, se procedió de la siguiente manera:

- Identificar el comportamiento de la interfaz de comunicación del puerto VGA.

- Realizar un diseño que reciba los datos del monitor y de esta manera ejecute la comunicación de la tarjeta lógica programable Basys2 con el monitor con puerto VGA.

- Establecer un bloque compacto de este diseño de comunicación del teclado para su uso en aplicaciones del laboratorio.

3.2.5 ENSAYO Y PRUEBA DEL MONITOR Y TECLADO EN CONJUNTO A LA TARJETA LÓGICA PROGRAMABLE BASYS2

En esta etapa, se procedió de la siguiente manera:

- Realizar ensayo y prueba del funcionamiento tanto del monitor como del teclado con la tarjeta lógica programable Basys2 y verificar su correcto funcionamiento para esta manera corregir los errores y fallas que puedan ocurrir.

- Desarrollar diseños de ejemplo para mostrar el funcionamiento en conjunto tanto del teclado como del monitor con la tarjeta lógica programable Basys2.

3.2.6 DESARROLLO DE UNA INTERFAZ GRÁFICA, APLICACIONES Y MANUAL DE USUARIO DE LA TARJETA BASYS 2

En esta etapa, se procedió de la siguiente manera:

- Desarrollar una interfaz gráfica mediante la tarjeta Basys2, donde se ilustren: indicadores Leds y Display 7-Segmentos, por medio del monitor con puerto VGA y un teclado con puerto PS/2.

- Realizar el desarrollo de una práctica combinacional y otra secuencial a fin de comprobar la adaptación correcta y comunicación de los puertos de la tarjeta lógico programable Basys2, además de verificar la complejidad del circuito empleado, y disponibilidad de compuertas para el desarrollo de otras prácticas.

- Describir en un manual de usuario las variables de entradas y salidas involucradas en el bloque de la interfaz gráfica para su fácil entendimiento y el funcionamiento de los sub-programas.

CAPITULO IV: RESULTADOS DE LA INVESTIGACIÓN

En este capítulo se explican los aspectos más resaltantes así como también la fundamentación básica del desarrollo de cada una de las fases de este Proyecto Especial de Grado.

4.1 LIMITACIONES DE LA TARJETA BASYS2

Luego de estudiar los manuales del Basys2 y comparar sus prestaciones con lo requerido en las prácticas de laboratorio se encontró lo siguiente:

- Cuenta con 8 interruptores de entrada y 4 pulsadores (12 entradas en total), lo cual es insuficiente para prácticas tales como: sumador y/o comparadores de 8 bits que requieren de dos números de 8bits, un total de 16 líneas de entradas.

- Cuenta con 7 líneas de segmentos y 8 leds de salida, lo cual es insuficiente para prácticas como sumadores de 8bits que requieren 2 display. El basys2 cuenta con 4 display pero requieren ser multiplexados, lo cual no es posible implementar por el estudiante hasta la décima semana de clase. Aun cuando se empleara un componente de multiplexado para que lo usaran los estudiantes como "caja negra", se siguen teniendo limitantes a 4 displays.

Teniendo que, por limitante de salida no es tan seria como en la entrada, que se requiere un mayor número de entradas adicionales; pero igual sería provechoso para el estudiante manejar más display o leds en el desarrollo de las prácticas.

4.2 DESCRIPCIÓN DEL SISTEMA MEDIANTE EL LENGUAJE VHDL

Entendiendo las limitaciones de la tarjeta Basys2 tanto para el Laboratorio de Lógica Digital como para el desarrollo de este proyecto, se realiza un esquema mediante el editor

esquemático del software ACTIVE-HDL, como se muestra en la Figura 4.1. En la que se presentan los bloques diseñados en este Proyecto de Grado.

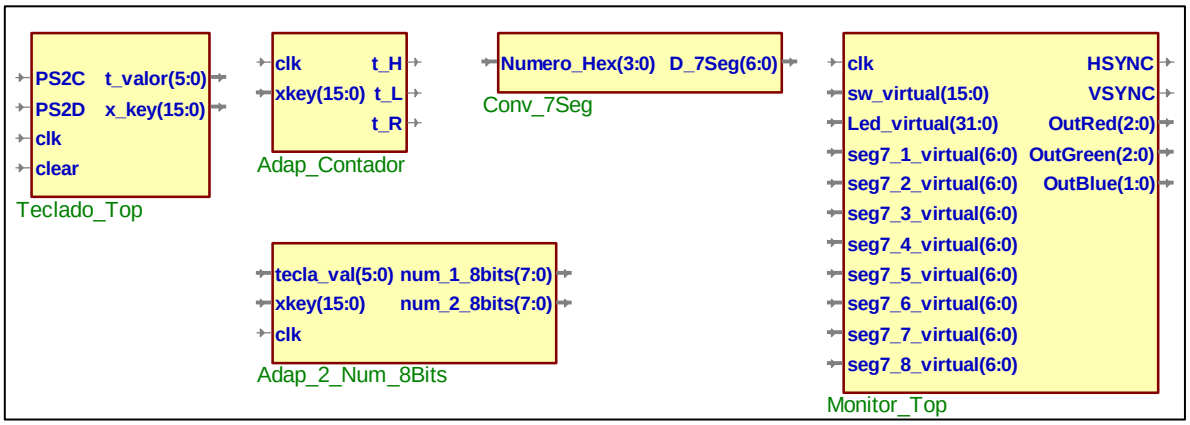


Figura 4.1 Diagrama de conexiones del sistema proporcionado

El sistema diseñado está constituido principalmente por los bloques Teclado_Top y Monitor_Top, que son los módulos encargados de realizar la comunicación entre los puertos de interés; y por los bloques Teclado_Lectura, Adap_Contador, Adap_2_Num_8Bits y conv_7seg, que sirven de intermediarios entre el diseño del usuario y los dos diseños principales encargados de la comunicación con los puertos VGA y PS/2.

Los módulos mostrados, son bloques que se pueden implementar individualmente, pero para ello se debe conocer su funcionamiento, por lo que en esta sección se describirán los subprogramas que usara el usuario, iniciando con Teclado_Top.

Subprograma Teclado_Top

El subprograma Teclado_Top, es un bloque cuya función es establecer la comunicación con el teclado con puerto PS/2 y transmitir la señal de salida en un código adaptado al usuario. Este subprograma está diseñado como bloque con la finalidad de simplificar el diseño de implementación del usuario. Se muestra en forma de diagrama de bloques en la Figura 4.2; y en la Figura 4.3 la declaración de las E/S de la entidad Teclado_Top.

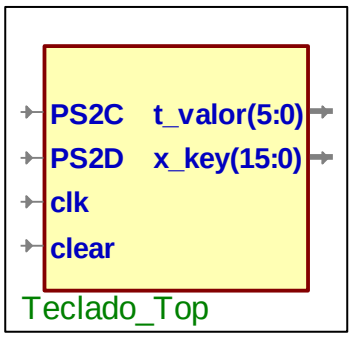


Figura 4.2 Bloque Teclado_Top.

```
entity Teclado_Top is
     port(
          PS2C : in STD_LOGIC;
          PS2D : in STD_LOGIC;
          clk : in STD_LOGIC;
          clear : in STD_LOGIC;
          t_valor : out std_logic_vector(5 downto 0);
          x_key : out std_logic_vector(15 downto 0)
          );
end Teclado_Top;
```

Figura 4.3 Entradas y Salidas de la entidad Teclado_Top.

El bloque Teclado_Top cuenta en la entrada con las señales PS2C y PS2D provenientes del teclado, son señales de reloj y datos del teclado respectivamente, identificados de esa manera en el código de restricciones de la tarjeta Basys2.

También cuenta con una señal de reloj clk y una señal clear para reiniciar la comunicación, que se puede asignar a un botón especifico de la tarjeta. Y por último, en la salida dispone de las señales: xkey y tecla_valor, que tienen como finalidad transmitir la información de datos del teclado, la primera tal como se recibe (código scan), y la segunda decodificada y adaptada al usuario.

Se muestra en la Tabla 4.1 y 4.2 Detalladas las E/S del bloque Teclado_Top.

Nombre	Tipo	Descripción
clk	Std_logic	Señal del reloj del oscilador de la tarjeta basys2.
clr	Std_logic	Señal identificada para reiniciar la comunicación.
PS2D	Std_logic	Señal identificada de los datos enviados desde el teclado.
PS2C	Std_logic	Señal identificada del reloj del teclado.

Tabla 4.1 Descripción de las señales de entrada del bloque Teclado_Top.

Nombre	Tipo	Descripción
t_valor	Std_logic_vector (15:0)	Valor asignado del código scan
x_key	Std_Logic_Vector(15:0)	Código Scan

Tabla 4.2 Descripción de las señales de salida del bloque Teclado_Top.

Este bloque está compuesto por dos subprogramas internos como se muestra en el diagrama de bloques de la Figura 4.4. La cual consta en primera instancia con un subprograma de lectura y luego un convertidor, para decodificar el valor de la tecla pulsada.

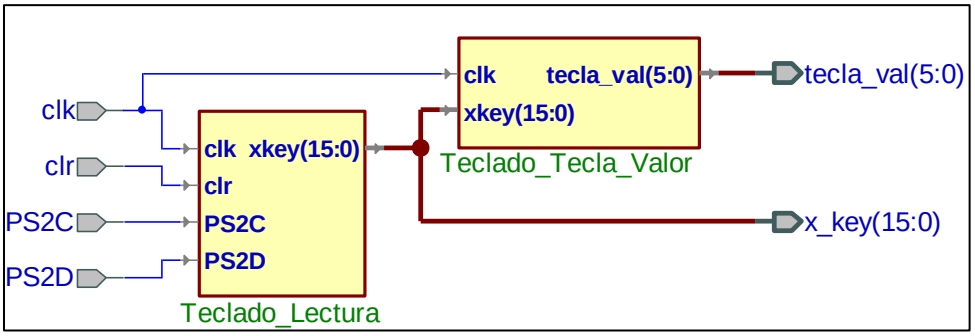


Figura 4.4 Conexión interna del subprograma Teclado_Top.

Subprograma Monitor_Top

Otro diseño principal de comunicación, es el bloque Monitor_Top, el cual efectúa la comunicación con el monitor y de esta manera elabora la interfaz gráfica en el monitor. Este subprograma está diseñado como bloque con la finalidad de simplificar el diseño de implementación del usuario. Se muestra en forma de diagrama de bloques en la Figura 4.5; y en la Figura 4.6 la declaración de las E/S de la entidad Monitor_Top.

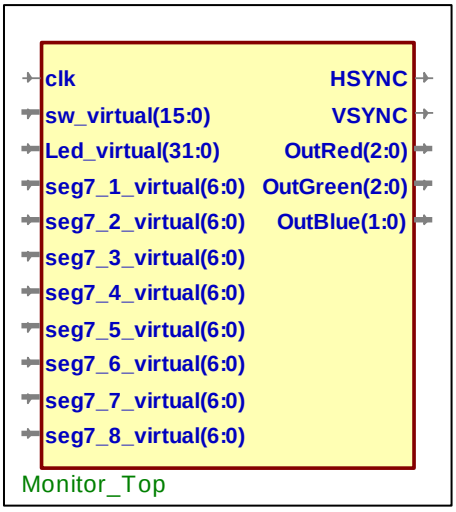


Figura 4.5 Bloque Monitor_Top.

```
entity Monitor_Top is
    port(
    clk              :   in  STD_LOGIC;
    sw_virtual       :   in  std_logic_vector(15 downto 0);
    Led_virtual      :   in  std_logic_vector(31 downto 0);
    seg7_1_virtual   :   in  std_logic_vector(6 downto 0);
    seg7_2_virtual   :   in  std_logic_vector(6 downto 0);
    seg7_3_virtual   :   in  std_logic_vector(6 downto 0);
    seg7_4_virtual   :   in  std_logic_vector(6 downto 0);
    seg7_5_virtual   :   in  std_logic_vector(6 downto 0);
    seg7_6_virtual   :   in  std_logic_vector(6 downto 0);
    seg7_7_virtual   :   in  std_logic_vector(6 downto 0);
    seg7_8_virtual   :   in  std_logic_vector(6 downto 0);
    HSYNC            :   out STD_LOGIC;
    VSYNC            :   out STD_LOGIC;
    OutRed           :   out std_logic_vector(2 downto 0);
    OutGreen         :   out std_logic_vector(2 downto 0);
    OutBlue          :   out std_logic_vector(1 downto 0)
    );
end Monitor_Top;
```

Figura 4.6 Entradas y salidas de la entidad Monitor_Top

Presenta sus entradas y salidas como se muestra en la Tabla 4.3 y Tabla 4.4 respectivamente. La cual se compone, en la entrada con todas las señales que se desean visualizar en el monitor, además de la señal de reloj clk para lograr la sincronización entre los equipos; en la salida de este bloque se encuentran las señales fundamentales para realizar la sincronización, las cuales son HSYNC y VSYNC, que se encargan de la sincronización horizontal y vertical del monitor respectivamente para barrer la pantalla y puntear cada pixel de la pantalla con el color correspondiente a la combinación de colores de las otras salidas de este bloques, es decir OutRed, OutGreen y OutBlue.

Nombre	Tipo	Descripción
clk	Std_logic	Reloj de entrada a 50MHz
sw_virtual	Std_logic_vector (15:0)	Switches virtuales
Led_virtual	Std_logic_vector (31:0)	Leds virtuales
seg7_1_virtual	Std_logic_vector (6:0)	Variable de 7 segmento virtual 1
seg7_2_virtual	Std_logic_vector (6:0)	Variable de entrada del 7 segmento virtual 2
seg7_3_virtual	Std_logic_vector (6:0)	Variable de entrada del 7 segmento virtual 3
seg7_4_virtual	Std_logic_vector (6:0)	Variable de entrada del 7 segmento virtual 4
seg7_5_virtual	Std_logic_vector (6:0)	Variable de entrada del 7 segmento virtual 5
seg7_6_virtual	Std_logic_vector (6:0)	Variable de entrada del 7 segmento virtual 6
seg7_7_virtual	Std_logic_vector (6:0)	Variable de entrada del 7 segmento virtual 7
seg7_8_virtual	Std_logic_vector (6:0)	Variable de entrada del 7 segmento virtual 8

Tabla 4.3 Descripción de las señales de entrada del bloque Monitor_Top.

Nombre	Tipo	Descripción
HSYNC	Std_logic	Señal de Sincronización Horizontal
VSYNC	Std_logic	Señal de Sincronización vertical
OutRed	Std_logic_vector (6:0)	Bits para la pigmentación de los pixeles en rojo
OutGreen	Std_logic_vector (6:0)	Bits para la pigmentación de los pixeles en verde
OutBlue	Std_logic_vector (6:0)	Bits para la pigmentación de los pixeles en azul

Tabla 4.4 Descripción de las señales de salida del bloque Monitor_Top.

La estructura interna, que muestra las interconexiones de los subprogramas internos de este diseño Monitor_Top se muestra en la figura 4.7. Y para una imagen con mayor detalle, en el Apéndice C.

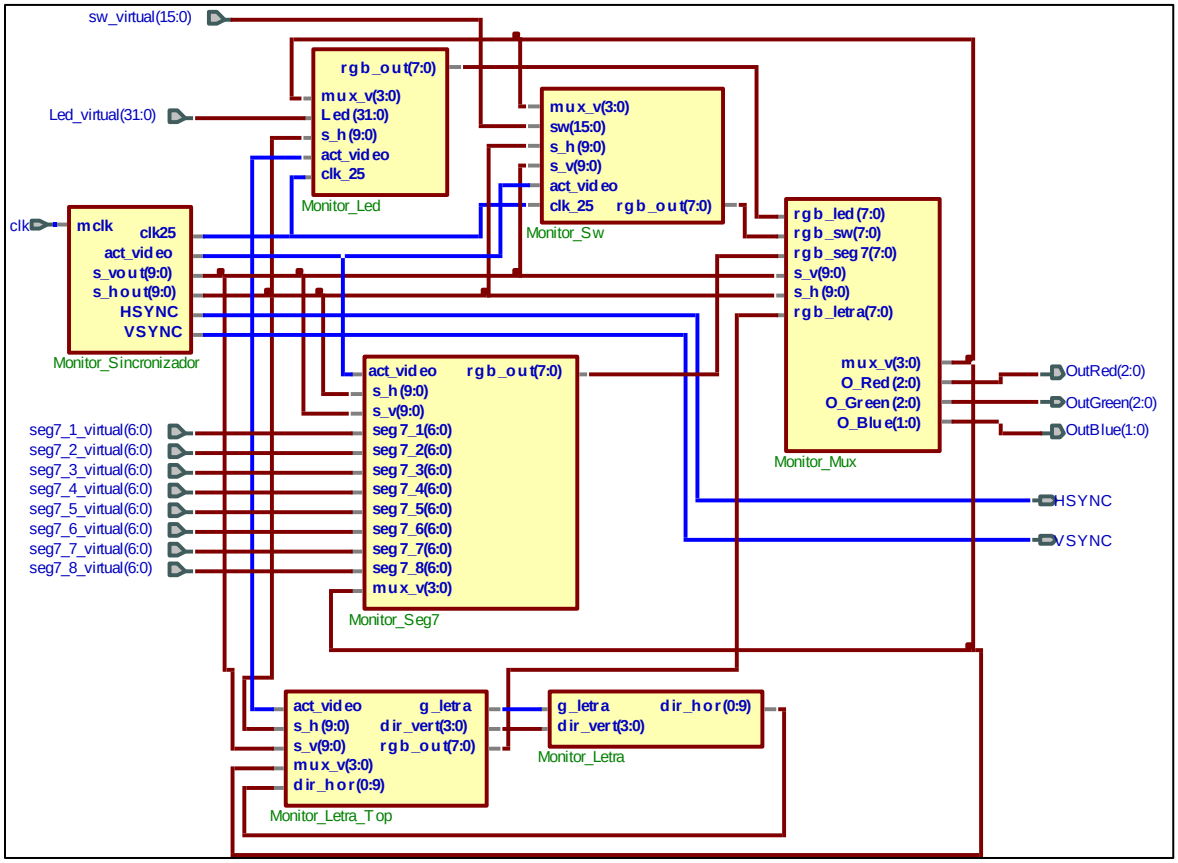


Figura 4.7 Conexión interna del diseño Monitor_Top.

Subprograma Adap_2_Num_8Bits

El subprograma Adap_2_Num_8Bits, es un adaptador que almacena dos números de ocho bits mediante una secuencia de teclas, cada número en una variable interna distinta y contenida en su memoria Flash; conveniente para efectuar suma y otras operaciones combinacionales. El diseño de entradas y salidas de este bloque se muestra en la Figura 4.8 y en la Figura 4.9 la declaración de las E/S de la entidad Adap_2_Num_8Bits.

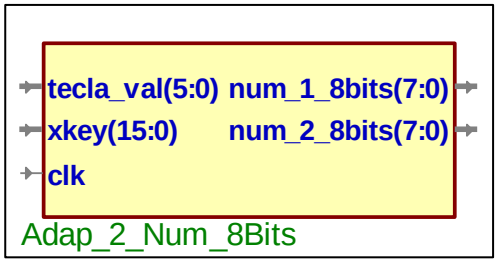


Figura 4.8 Bloque Adap_2_Num_8Bits.

```
entity Adap_2_Num_8Bits is
    port(
    clk : in STD_LOGIC;
    tecla_val:  in std_logic_vector(5 downto 0);
    xkey    : in std_logic_vector(15 downto 0);
    num_1_8bits : out std_logic_vector(7 downto 0);
    num_2_8bits : out std_logic_vector(7 downto 0)
    );
end Adap_2_Num_8Bits;
```

Figura 4.9 Entradas y salidas de la entidad Adap_2_Num_8Bits.

Este bloque presenta en la entrada la señal de xkey para identificar en código scan la tecla presionada; y la señal tecla_val para almacenar los Bytes de la tecla pulsada, que al culminar la secuencia se transmite por las señales de salida num_1_8Bits y num_2_8Bits, dependiendo de la secuencia tomada.

Del bloque se detallan sus variables de entrada en la Tabla 4.5 y sus variables de salida en la Tabla 4.6.

Nombre	Tipo	Descripción
tecla_val	std_logic_vector (5:0)	Valor asignado del codigo scan
xkey	std_logic_vector (15:0)	Código scan del teclado

Tabla 4.5 Descripción de las señales de entrada de Adap_2_Num_8Bits.

Nombre	Tipo	Descripción
num_1_8bits(7:0)	std_logic	Número 1 de 8 bits
num_2_8bits(7:0)	std_logic	Número 2 de 8 bits

Tabla 4.6 Descripción de las señales de salida del bloque Adap_2_Num_8Bits.

La secuencia de teclas para almacenar los Bytes de la tecla presionada es:

- Se presiona la tecla F1 o F2 para especificar qué número añadir.
- Se presionan dos teclas consecutivas (las dos teclas deben tener valor hexadecimal de 0 a F) (el primer hexadecimal es el más significativo y el 2do hexadecimal el menos significativo):
    - Se presiona la primera tecla.
    - Luego se presiona la segunda.
- Se pulsa ENTER (utilizada para finalizar la toma de datos).

En el momento de seleccionar los hexadecimales, si se presiona una tecla que no sea numérica o de la A-F, entonces el subprograma no lo toma en cuenta y sigue esperando hasta que se presione una tecla numérica o de la A-F. Si se presiona la tecla F1 o F2 y luego se presiona una tecla numérica o de la A-F y por último se presiona la tecla ENTER, entonces el hexadecimal será el más significativo y el menos significativo será cero.

Subprograma Adap_Contador

El subprograma Adap_Contador, es un adaptador de las señales provenientes del subprograma Teclado_Top para el uso en aplicaciones secuenciales. Diseñado para que el usuario disponga de tres señales que se comporten como un pulsador, con un “1” lógico en caso de ser pulsadas y con un “0” lógico en caso contrario. El diseño de entradas y salidas de este bloque se muestra en la Figura 4.10, y en la Figura 4.11 la declaración de las E/S.

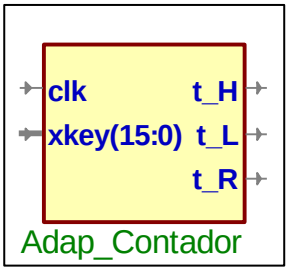


Figura 4.10 Bloque Adap_Contador.

```
entity Adap_Contador is
     port(
        clk : in std_logic;
        xkey: in std_logic_vector(15 downto 0);
        t_H: out std_logic;
        t_L: out std_logic;
        t_R: out std_logic
        );
end Adap_Contador;
```

Figura 4.11 Entradas y salidas de la entidad Adap_Contador.

Este bloque posee en la entrada la señal de reloj clk de 50MHz, lo que le permite recibir en sincronía los datos de xkey y realizar operaciones secuenciales. Del bloque se detallan sus variables de entrada en la Tabla 4.7 y sus variables de salida en la Tabla 4.8.

Nombre	Tipo	Descripción
clk	Std_logic	Señal de reloj a 50 MHz
xkey	Std_logic_vector (15:0)	Código scan del teclado

Tabla 4.7 Descripción de las señales de entrada del bloque Adap_Contador.

Nombre	Tipo	Descripción
T_H	Std_logic	Cuenta ascendente en 1
T_L	Std_logic	Cuenta descendente en 1
T_R	Std_logic	Reinicia la cuenta

Tabla 4.8 Descripción de las señales de salida del bloque Adap_Contador.

Para realizar el envío de las señales como valores binarios se determinó que al pulsar las teclas correspondientes (H, L y R) se envía un valor lógico alto e inmediatamente al despulsar se envía el valor lógico bajo mediante las señales de salida T_H, T_L y T_R. Con este diseño se garantiza que incluso presionando varias teclas al mismo tiempo, las teclas seguirán siendo independientes unas de otras.

Subprograma Conv_7Seg.

Tiene además un adaptador denominado Conv_7Seg, que tiene como función la de convertir la señal para el display 7-Segmentos, una señal de cuatro bits en la entrada de esta entidad es suficiente para convertirla en una señal de 7 bits adaptada a los siete segmentos del monitor. El diseño de entradas y salidas de este bloque se muestra en la Figura 4.12 y en la Figura 4.13 la declaración de las E/S de la entidad Conv_7Seg.

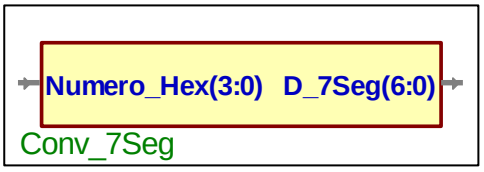


Figura 4.12 Bloque Conv_7Seg.

```
entity Conv_7Seg is
     port(
          Numero_Hex : in STD_LOGIC_VECTOR(3 downto 0);
          D_7Seg : out STD_LOGIC_VECTOR(6 downto 0)
          );
end Conv_7Seg;
```

Figura 4.13 Entradas y salidas de la entidad Conv_7Seg.

Del bloque se detallan sus variables de entrada en la Tabla 4.9 y sus variables de salida en la Tabla 4.10.

Nombre	Tipo	Descripción
Numero_Hex	std_logic_vector (3:0)	Numero de 4 bits

Tabla 4.9 Descripción de las señales de entrada del bloque Adap_Contador.

Nombre	Tipo	Descripción
D_7Seg	std_logic_vector (6: 0)	Adaptada el número para el 7-Segmentos virtual.

Tabla 4.10 Descripción de las señales de salida del bloque Adap_Contador.

De los que se resalta principalmente los módulos del Teclado_Top y Monitor_Top, que son los subprogramas bases para realizar la comunicación entre los puertos de interés, en cuanto los subprogramas adap_contador y adap_2_num_8bits y Conv_7Seg son los que ejercen de intermediarios entre los subprogramas bases de la comunicación y el diseño del usuario.

4.3 CÓDIGO DE PROGRAMACIÓN PARA LOGRAR UNA COMUNICACIÓN ENTRE LA BASYS2 Y EL TECLADO

La tarjeta Basys2 no realiza el protocolo de comunicación del puerto PS/2 conectado a un teclado, pero reserva los pines PS2C y PS2D del puerto PS/2 para que el usuario realice dicho protocolo de comunicación.

El protocolo de comunicación en este caso, de un teclado con puerto PS/2, consiste en enviar una cadena de bits en forma de señal, cada vez que se pulsa o despulsa una tecla. Esta cadena de bits se envía a la frecuencia del reloj interno del teclado. Cada vez que viene un flanco ascendente de la señal del reloj interno, se va enviado un bit de dato y así sucesivamente, como se muestra en la Figura 4.14.

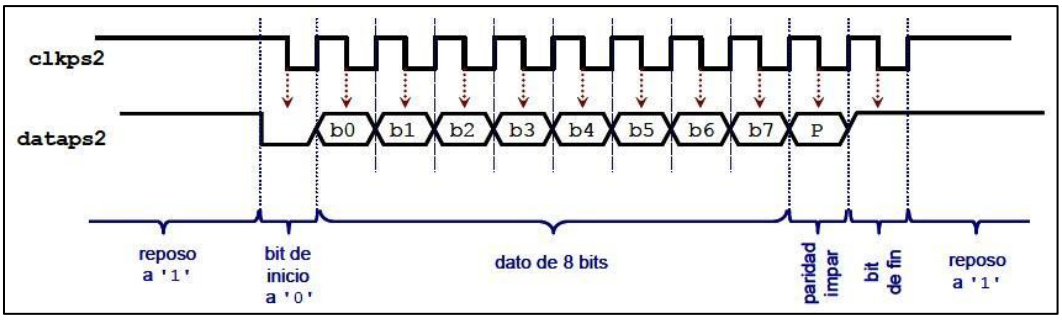


Figura 4.14 Cronograma de comunicación con el puerto PS/2.
Fuente: Maxinez, D y Alcalá, J [11].

Por la señal del pin PS2D, se envía la cadena de bits de datos, la cual está compuesta por un bit de inicio, el dato de 8 bits que se envía desde el bit menos significativo, le sigue un bit de paridad para verificar la validez de la trama y un bit de stop para informar que ha culminado la cadena de bits. La señal permanece en un uno lógico cuando no hay datos que enviar. Como se muestra en la Figura 4.14.

Se debe leer en los flancos descendentes de la señal del reloj del teclado, para poder coincidir correctamente la señal de reloj con los datos de la trama, como se indica en la Figura 4.14.

La descripción de las partes que componen la cadena de bits en forma de señal del pin PS2D, se encuentra en la Tabla 4.11.

BIT	FUNCIÓN
START	El primer bit de START le indica al teclado el inicio de la trama, normalmente en estado bajo para iniciar la trama.
8 BITS DE DATOS	Se envía desde el teclado una secuencia de 8 bits empezando por el bit menos significativo de la trama en forma de código scan , para cada tecla, pulsada o despulsada.
PARIDAD	Este bit le indica al teclado cuando la trama de datos sea válida.
STOP	Este último bit de stop estará en uno lógico para informar al teclado la culminación de la trama.

Tabla 4.11 Descripción de las señales enviadas desde el teclado al host.

Hasta ahora se sabe cómo recibir la información de un dispositivo de entrada con puerto PS/2, lo que no sabemos es el significado que tienen esas tramas recibidas. Ahora, se tomará como dispositivo de entrada el teclado, no se sabe la relación que existe entre las teclas que pulsamos y lo que se envía por el PS/2. La información suministrada es tomada de Machado, Borromeo y Rodriguez [12], y Haskell y Hanna [6].

Para un teclado con puerto PS/2, cada tecla presionada es identificada por un código scan. El código es asociado con una tecla física. Tanto así que la tecla Shift izquierda y derecha poseen un código scan distinto. Cuando se presiona una tecla el código scan Make es enviado al puerto PS/2. Cuando se libera la tecla el código scan Break es enviado al puerto PS/2. Los códigos scan Make y Break, de todas las teclas se encuentran en la Tabla 4.12.

Key	Make	Break
A	1C	F0,1C
B	32	F0,32
C	21	F0,21
D	23	F0,23
E	24	F0,24
F	2B	F0,2B
G	34	F0,34
H	33	F0,33
I	43	F0,43
J	3B	F0,3B
K	42	F0,42
L	4B	F0,4B
M	3A	F0,3A
N	31	F0,31
O	44	F0,44
P	4D	F0,4D
Q	15	F0,15
R	2D	F0,2D
S	1B	F0,1B
T	2C	F0,2C
U	3C	F0,3C
V	2A	F0,2A
W	1D	F0,1D
X	22	F0,22
Y	35	F0,35
Z	1A	F0,1A
0	45	F0,45
1	16	F0,16
2	1E	F0,1E
3	26	F0,26
4	25	F0,25
5	2E	F0,2E
6	36	F0,36
7	3D	F0,3D
8	3E	F0,3E
9	46	F0,46

Key	Make	Break
`	0E	F0,0E
-	4E	F0,4E
=	55	F0,55
\	5D	F0,5D
BKSP	66	F0,66
SPACE	29	F0,29
TAB	0D	F0,0D
CAPS	58	F0,58
L Shift	12	F0,12
R Shift	59	F0,59
L Ctrl	14	F0,14
R Ctrl	E0,14	E0,F0,14
L Alt	11	F0,11
R Alt	E0,11	E0,F0,11
L GUI	E0,1F	E0,F0,1F
R GUI	E0,27	E0,F0,27
Apps	E0,2F	E0,F0,2F
Enter	5A	F0,5A
ESC	76	F0,76
Scroll	7E	F0,7E
Insert	E0,70	E0,F0,70
Home	E0,6C	E0,F0,6C
Page Up	E0,7D	E0,F0,7D
Page Dn	E0,7A	E0,F0,7A
Delete	E0,71	E0,F0,71
End	E0,69	E0,F0,69
[	54	F0,54
]	5B	F0,5B
;	4C	F0,4C
'	52	F0,52
,	41	F0,41
.	49	F0,49
/	4A	F0,4A
PrntScrn	E0,7C, E0,12	E0,F0,7C, E0,F0,12

Key	Make	Break
F1	05	F0,05
F2	06	F0,06
F3	04	F0,04
F4	0C	F0,0C
F5	03	F0,03
F6	0B	F0,0B
F7	83	F0,83
F8	0A	F0,0A
F9	01	F0,01
F10	09	F0,09
F11	78	F0,78
F12	07	F0,07
Num	77	F0,77
KP /	E0,4A	E0,F0,4A
KP *	7C	F0,7C
KP -	7B	F0,7B
KP +	79	F0,79
KP EN	E0,5A	E0,F0,5A
KP .	71	F0,71
KP 0	70	F0,70
KP 1	69	F0,69
KP 2	72	F0,72
KP 3	7A	F0,7A
KP 4	6B	F0,6B
KP 5	73	F0,73
KP 6	74	F0,74
KP 7	6C	F0,6C
KP 8	75	F0,75
KP 9	7D	F0,7D
U Arrow	E0,75	E0,F0,75
L Arrow	E0,6B	E0,F0,6B
D Arrow	E0,72	E0,F0,72
R Arrow	E0,74	E0,F0,74
Pause	E1,14, 77,E1, F0,14, F0,77,	None

Tabla 4.12 Códigos scan Make y Break de un teclado inglés norteamericano. Fuente: Haskell R., Hanna D [6].

Podemos observar en la Tabla 4.12 y Figura 4.15, que para todas las letras y dígitos, el código scan Make es un byte y el código scan Break es el mismo byte precedido de un byte hexadecimal F0. Para algunas teclas el código scan Make es de dos bytes el cual el primer byte hexadecimal es E0, y su código scan Break tiene tres bytes. Hay dos teclas que son casos especiales, la tecla PrntScrn y Pausa, el código scan Make es de cuatro bytes y ocho bytes respectivamente.

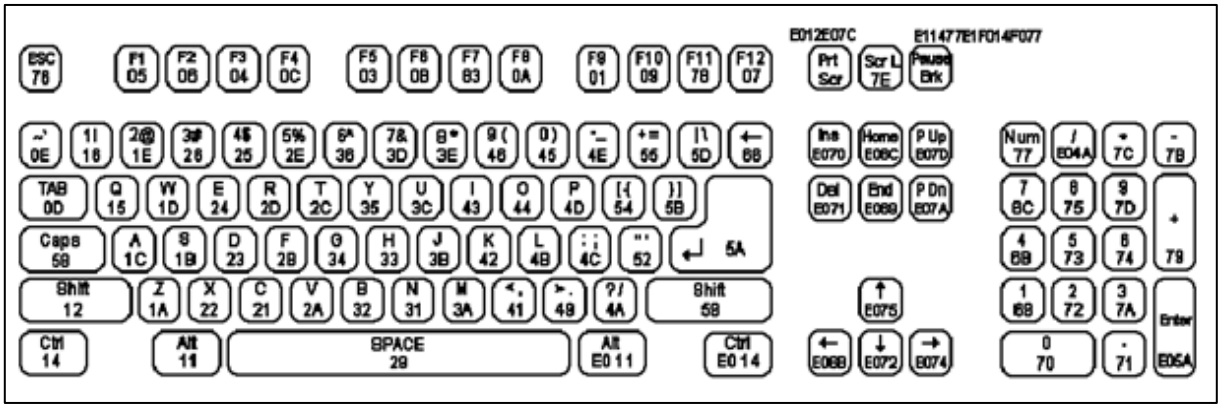


Figura 4.15 Códigos scan de un teclado inglés norteamericano.
Fuente: Haskell R., Hanna D [6]

Los códigos scan no tienen relación con los códigos ASCII de cada carácter del teclado. Ahora, para escribir una letra en mayúscula o minúscula, se debe chequear si se está presionando la tecla Shift derecha o izquierda y presionar alguna tecla de letra antes de soltar la tecla Shift, o en tal caso la tecla CapsLock (Bloqueo de mayúscula).

Como por ejemplo, para escribir una letra en minúscula como la "f", se pulsa la tecla "f" y luego se despulsa, entonces se enviarán los siguientes bytes por el puerto PS/2 como en la Figura 4.16:

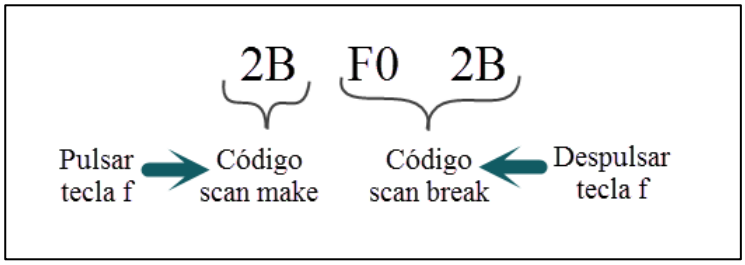


Figura 4.16 Cadena de bytes para describir una "f" minúscula.

Ahora, si se quiere escribir la letra "f" en mayúscula, se presiona la tecla Shift izquierdo y luego la tecla "f", se libera la tecla "f" y luego la tecla Shift izquierdo, y se enviará los siguientes bytes por el puerto PS/2, como se muestra en la Figura 4.17.

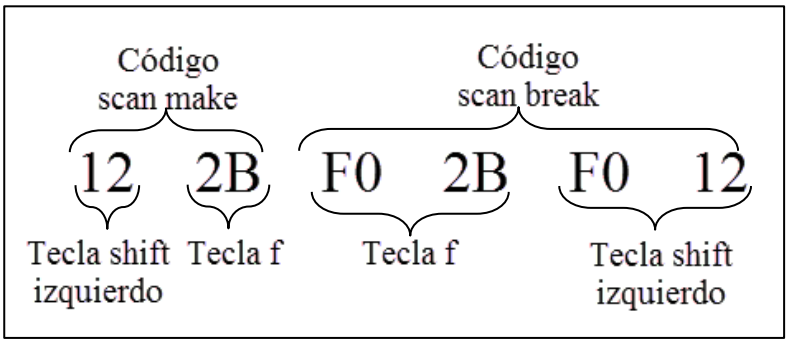


Figura 4.17 Cadena de bytes para describir una "f" mayúscula.

Existe otro caso, que es cuando se deja presionado una tecla del teclado, esto se denomina typematic, si se deja presionado por un tiempo de 0.25 a 1 segundo, se enviará una sola vez el código scan Make de esa tecla, pero si se deja presionado por más de 1 segundo, entonces se enviará el código scan Make seguido de la misma continuamente a una velocidad de 2 a 30 caracteres por segundo. El typematic se puede configurar, tanto el tiempo como la velocidad, al enviar un byte que codifica al comando 0x3F del teclado.

Existen varios tipos de teclado, y cada teclado tiene diferentes códigos scan para la misma tecla, así como se muestra en el apéndice 4.a.

Luego de determinar el tipo de teclado que se usará, y se identifica las variables PS2C y PS2D, se realiza un diseño en VHDL, capaz de establecer la comunicación entre el teclado con puerto PS/2 y la tarjeta BASYS2.

El módulo que realiza este protocolo de comunicación es Teclado_Lectura, el cual se encuentra contenido en Teclado_Top como se muestra en la Figura 4.18.

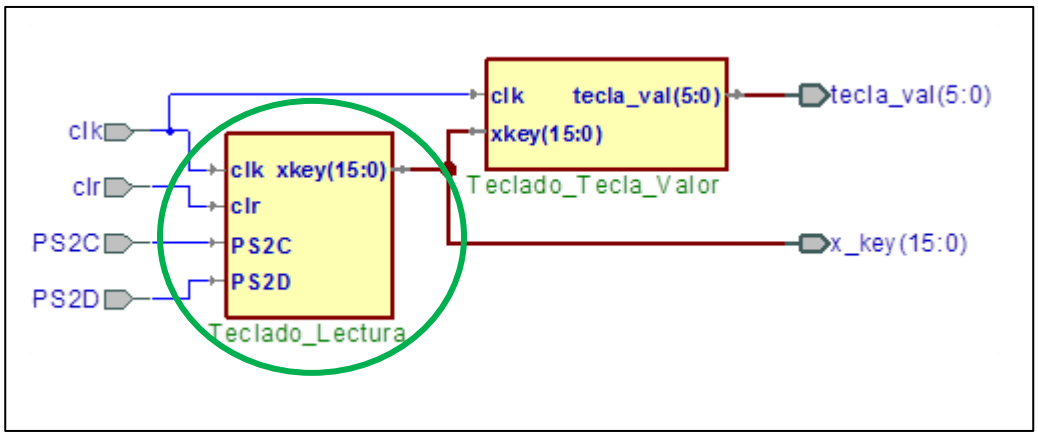


Figura 4.18 Conexión del bloque Teclado_Lectura en el bloque Teclado_Top.

Subprograma Teclado_Lectura

Este subprograma se encarga de realizar la lectura de las teclas de un teclado por medio del pin PS2C y PS2D, y muestra dos bytes del código scan del teclado. En la Figura 4.19 se muestra la entidad de dicho subprograma.

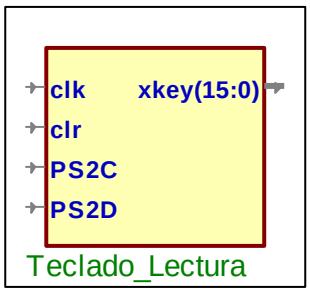


Figura 4.19 Entidad del subprograma Teclado_Lectura.

Las variables de entradas y salidas de la entidad del subprograma Teclado_Lectura se detallan en la Tabla 4.13 y en la Tabla 4.14 respectivamente.

Nombre	Tipo	Descripción
Clk	Std_logic	Señal del reloj aplicado.
Clr	Std_logic	Señal clear para reiniciar la comunicación.
PS2D	Std_logic	Señal de la data del teclado.
PS2C	Std_logic	Señal del reloj del teclado.

Tabla 4.13 Descripción de las señales de entrada del bloque Teclado_Lectura.

Nombre	Tipo	Descripción
xkey	Std_logic_vector (15:0)	Código scan

Tabla 4.14 Descripción de las señales de salida del bloque Teclado_Lectura.

En la Figura 4.20, se describe el funcionamiento del subprograma Teclado_Lectura por medio de un diagrama de flujo.

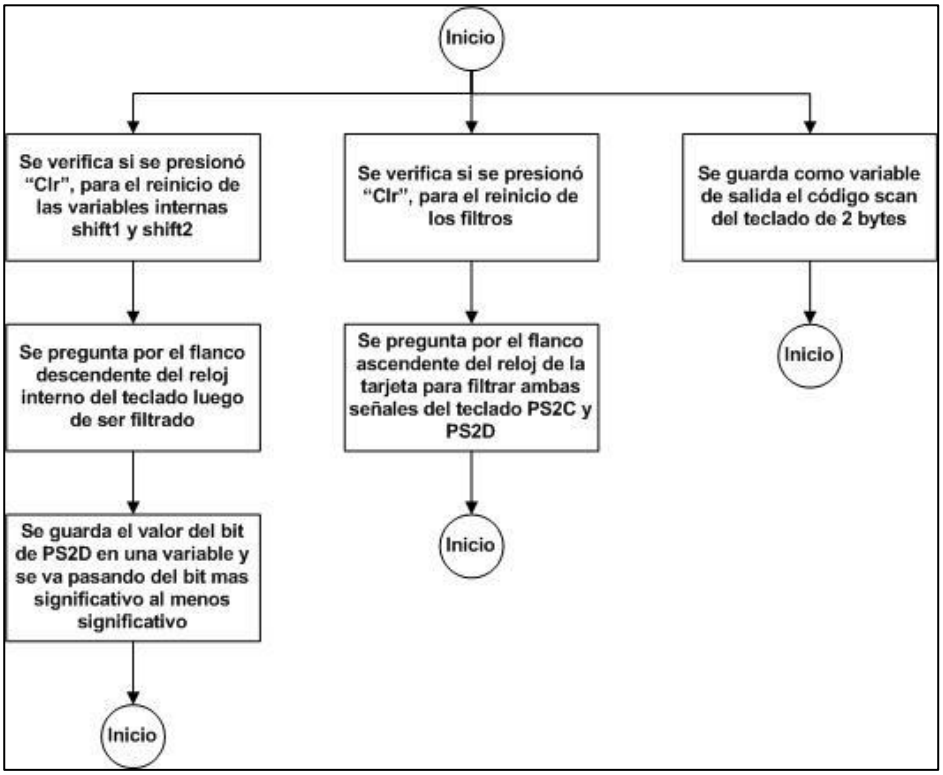


Figura 4.20 Comportamiento del diseño del Teclado_Lectura.

El filtro es para prevenir algún ruido que tenga las señales de los pines PS2C y PS2D, y no tomar una lectura errónea.

Después que se filtran las señales, se guarda el valor de la señal PS2D en el instante que ocurre un flanco descendente del reloj interno del teclado. Este valor se coloca en el bit más significativo al menos significativo formando así una cadena de bits, por último se toman de dos cadenas de bits consecutivos, los bits de datos y se mandan por la variable de salida “xkey”, un valor de dos bytes.

La variable “xkey” consiste en cuatro hexadecimales provenientes de la señal del pin PS2D, con estos cuatro hexadecimales se puede determinar si una tecla esta pulsada, se mantiene pulsada o se despulsó, así como se muestra en la Figura 4.21.

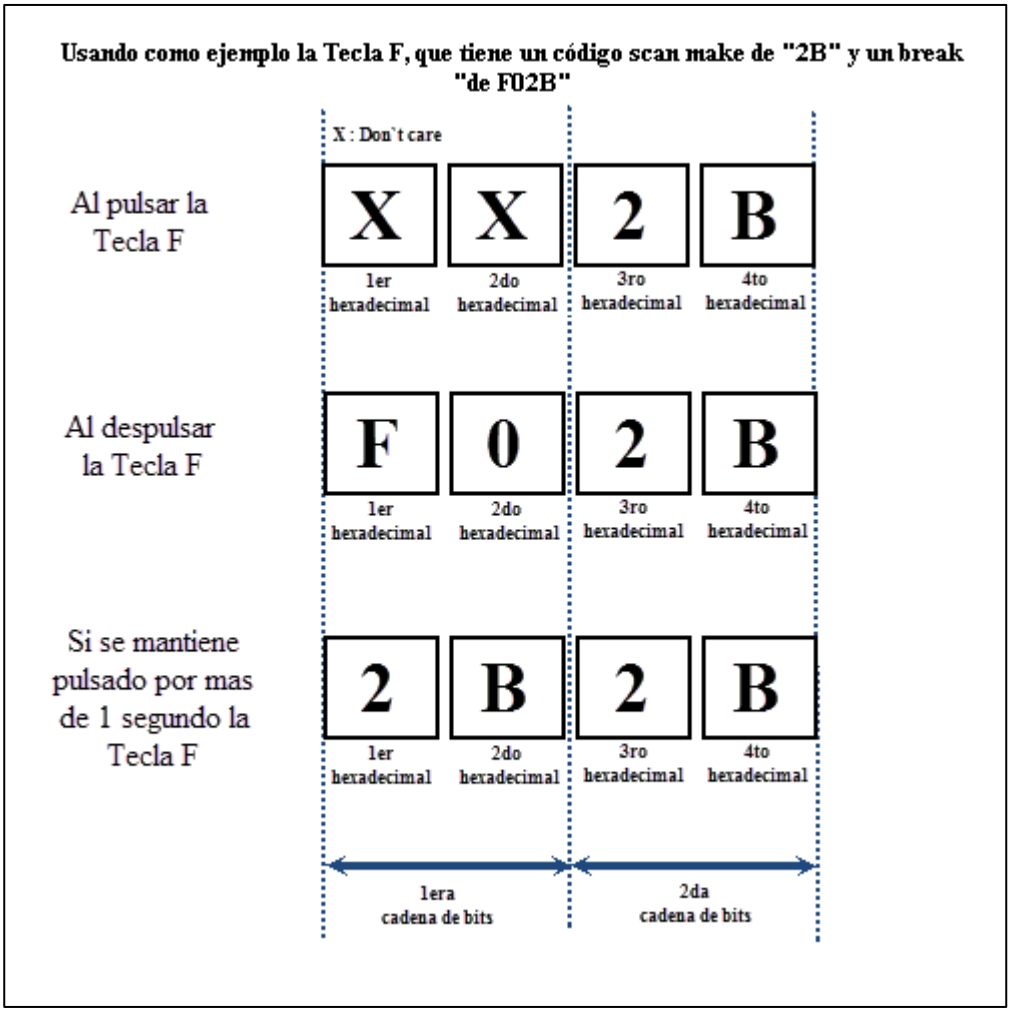


Figura 4.21 Diferentes lecturas del pin PS2D del teclado, de acuerdo a su caso.

Entonces al preguntar por los primeros dos hexadecimales si son “F0” sabemos si la tecla esta despulsada o no, y con los dos últimos hexadecimales se conoce cuál es la tecla que se está ejecutando.

Como el valor de la variable de salida “xkey” no sigue un orden como el código ASCII, entonces se diseña otro módulo denominado Teclado_Tecla_Valor, en donde está contenido dentro de teclado_Top, así como se muestra en la Figura 4.22.

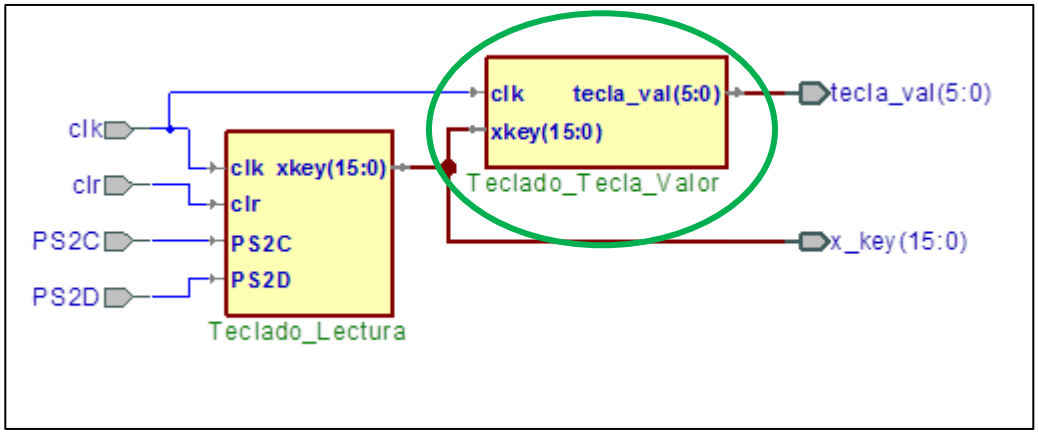


Figura 4.22 Conexión del bloque Teclado_Tecla_Valor en el bloque Teclado_Top.

Subprograma Teclado_Tecla_Valor

El subprograma Teclado_Tecla_Valor, tiene la función de convertir la señal xkey del código scan a un código propio a fin de adaptarlo al usuario. El diseño de entradas y salidas de este bloque se presenta en la en la Figura 4.23.

Figura 4.23 Bloque Teclado_Tecla_Valor.

Del bloque se detallan sus variables de entrada en la Tabla 4.15 y sus variables de salida en la Tabla 4.16.

Nombre	Tipo	Descripción
Clk	Std_logic	Señal del reloj del oscilador de la tarjeta basys2.
Xkey	Std_logic_vector	Código scan

Tabla 4.15 Descripción de las señales de entrada del bloque Teclado_Tecla_Valor.

Nombre	Tipo	Descripción
tecla_val	Std_logic_vector (5:0)	Señal de la tecla

Tabla 4.16 Descripción de las señales de salida del bloque Teclado_Tecla_Valor.

Este bloque presenta en la entrada la señal clk, para escanear el estado de xkey con la misma frecuencia en la que se envía esa señal desde el subprograma Teclado_Lectura.

Por cada flanco ascendente de la señal de reloj clk se realiza la decodificación de xkey, de acuerdo a la Tabla 4.17; para transmitirla mediante la señal de salida tecla_val. Pero una vez que se haya soltado la tecla, momento en el que se tiene una mejor lectura del dato. De lo contrario la señal de xkey tendrá valor por defecto de "110010".

Tecla	Valor en binario	Tecla	Valor en binario	Tecla	Valor en binario
0	000000	H	010001	Y	100010
1	000001	I	010010	Z	100011
2	000010	J	010011	F1	100100
3	000011	K	010100	F2	100101
4	000100	L	010101	F3	100110
5	000101	M	010110	F4	100111
6	000110	N	010111	F5	101000
7	000111	O	011000	F6	101001
8	001000	P	011001	F7	101010
9	001001	Q	011010	F8	101011
A	001010	R	011011	F9	101100
B	001011	S	011100	F10	101101
C	001100	T	011101	F11	101110
D	001101	U	011110	F12	101111

E	001110	V	011111	ENTER	110000
F	001111	W	100000		
G	010000	X	100001		

Tabla 4.17 Código diseñado para adaptar la señal de xkey al usuario

4.4 CÓDIGO DE PROGRAMACIÓN PARA LOGRAR UNA COMUNICACIÓN ENTRE LA TARJETA BASYS2 Y EL MONITOR

Así como la tarjeta BASYS2 no realiza el protocolo de comunicación del puerto PS/2, tampoco lo hace con el puerto VGA, pero si tiene reservado los pines de salida HSYNC, VSYNC, OutRed, OutGreen, y OutBlue para realizar dicho protocolo de comunicación.

El protocolo de comunicación consiste en enviar valores a los pixeles de la pantalla del monitor, mediante un barrido de izquierda a derecha por líneas horizontales y de arriba hacia abajo por líneas verticales. Se empieza desde el envío del primer pixel de arriba a la izquierda, continuando por un recorrido horizontal de izquierda a derecha, como se indica en la Figura 4.24. El píxel va a la misma frecuencia que la señal de reloj aplicada.

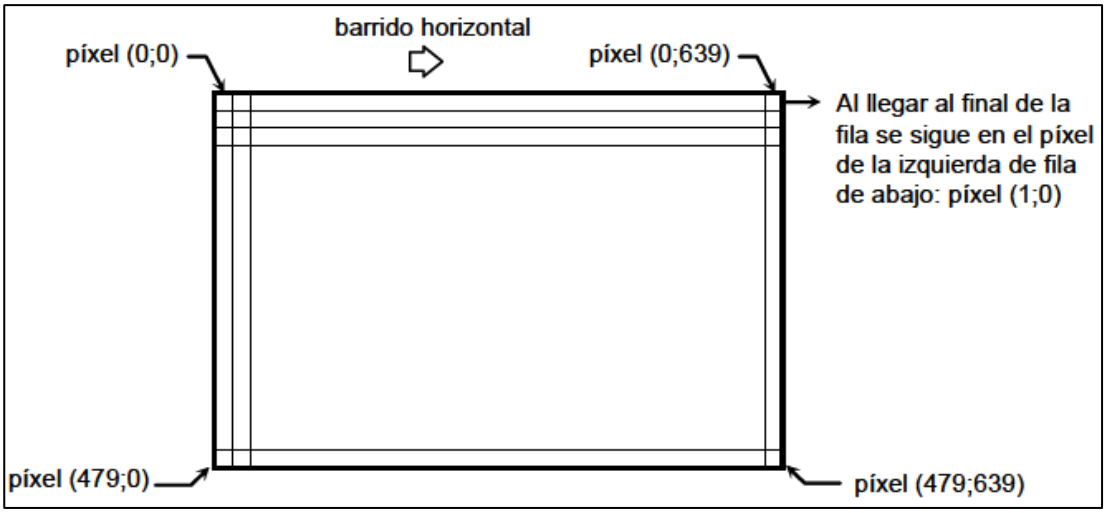


Figura 4.24 Pixeles en una pantalla con resolución de 640x480.
Fuente: Diseño de sistemas digitales con VHDL. [12]

El sincronismo horizontal (HSYNC) efectúa el barrido del pixel en líneas horizontales y el sincronismo vertical (VSYNC) efectúa el barrido del pixel en una línea vertical, al llegar a la final se hace un cambio de pantalla cuando se finaliza el barrido de los pixeles por toda la pantalla, esta vuelve al punto de inicio, arriba a la izquierda.

Sincronismo horizontal (HSYNC)

Es una señal cuadrática que se le envía al monitor en donde el ancho de la ventana activa indica la cantidad de pixeles que realiza el barrido horizontal. No todos los pixeles en el ancho de la ventana activa están en la zona de visualización de la pantalla, porque tiene que haber un intervalo de tiempo entre la señal de sincronismo y el envío de la información de los pixeles, a estos se les llama borde delantero y trasero, también se les denomina front porch y back porch respectivamente.

La zona de visualización de la pantalla es controlada por una señal interna (video activo), esta señal indicará si el pixel se encuentra dentro de los límites de la zona de visualización.

Cuando termina el ancho de la ventana activa del sincronismo horizontal, ocurre un retraso, significa que el pixel retorna al inicio de la línea horizontal.

El borde delantero (Front porch), es la zona activa de la señal de sincronismo después de finalizar la zona de visualización y el borde trasero (Back porch), es la zona activa de la señal de sincronismo antes de empezar la zona de visualización, así como se indica en la Figura 4.25.

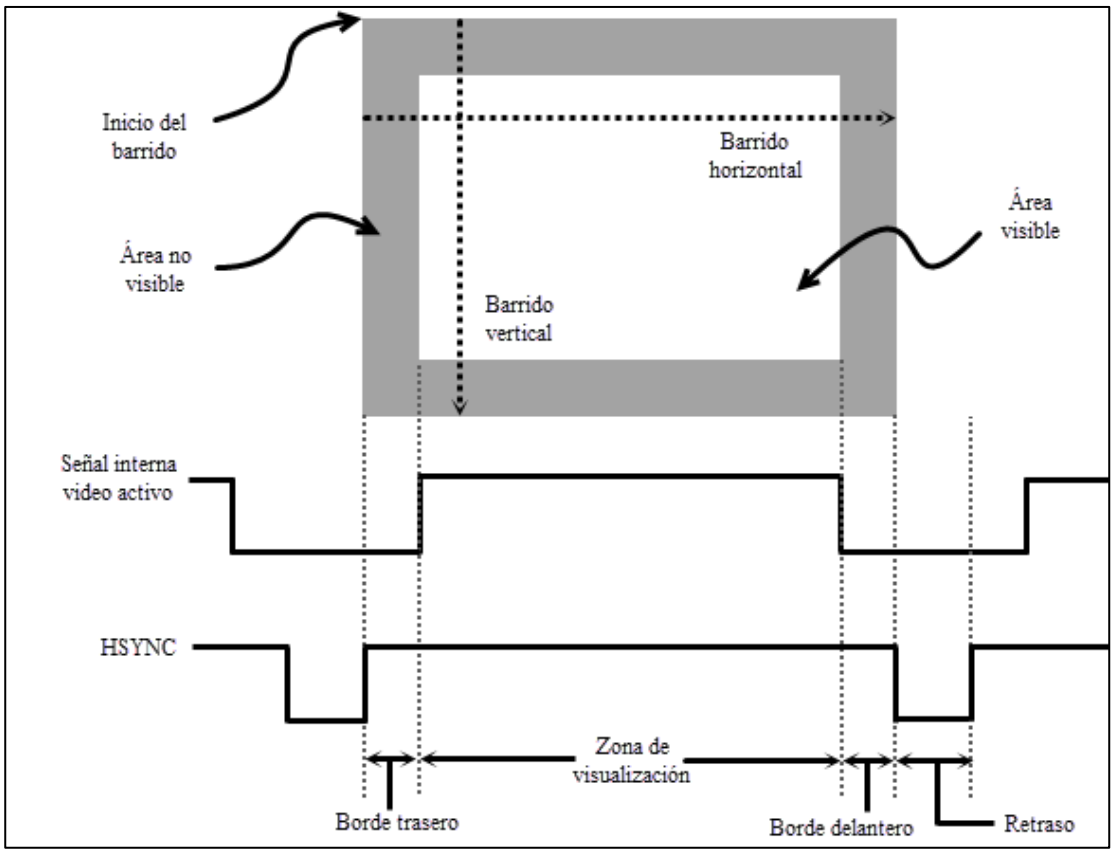


Figura 4.25 Pixeles en una pantalla con resolución de 640x480.

Sincronismo vertical (VSYNC)

Al igual que el sincronismo horizontal, es una señal cuadrática que se le envía al monitor en donde el ancho de la ventana activa indica el número de filas de píxeles que realiza el barrido vertical, cada fila de píxeles está compuesta por una línea de píxeles horizontal así como se muestra en la Figura 4.26.

Cada ancho de la ventana activa de la señal de sincronismo vertical es una pantalla de visualización en el monitor, la frecuencia de la misma indicará el número de pantallas por segundo.

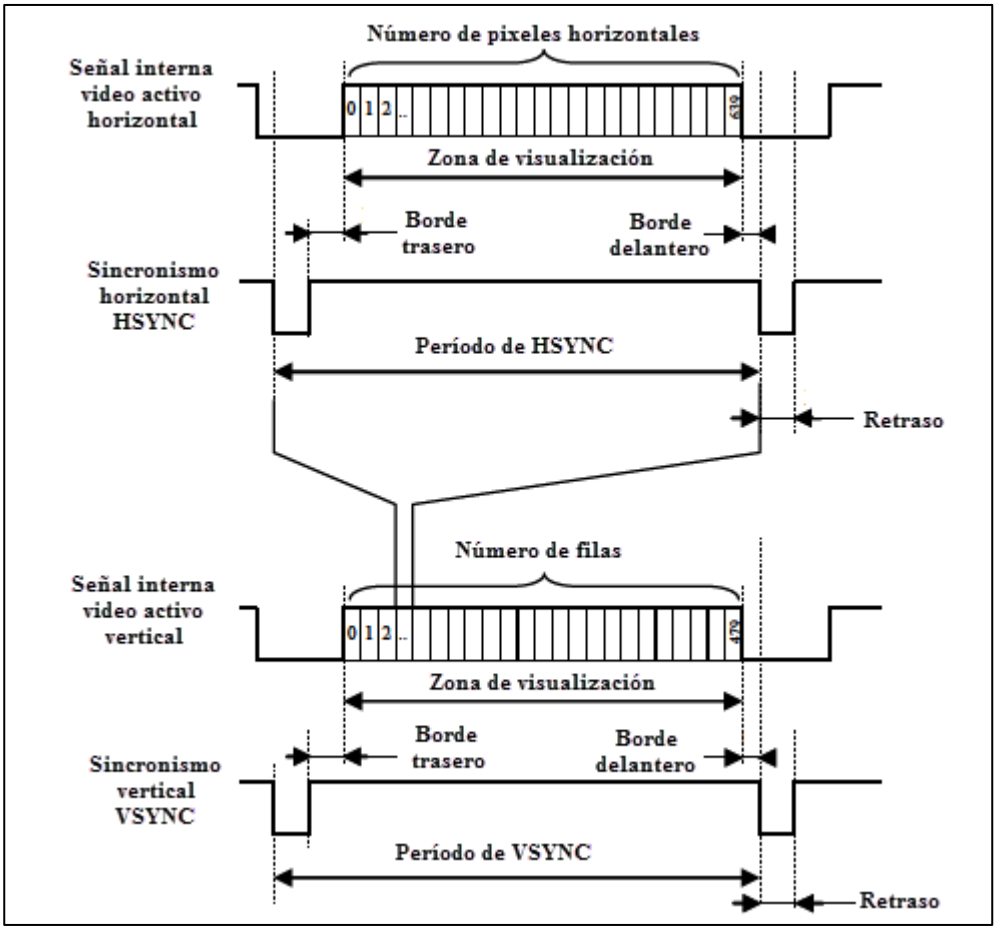


Figura 4.26 Señales de sincronismo VSYNC y HSYNC del puerto VGA.

El estándar de la VGA trabaja a varias frecuencias, dependiendo de la resolución que se quiera. En la Figura 4.27, para una resolución de 640x800, se muestra la especificación de la línea horizontal en píxeles y no en tiempos, también indica el número de píxeles que llevan los bordes.

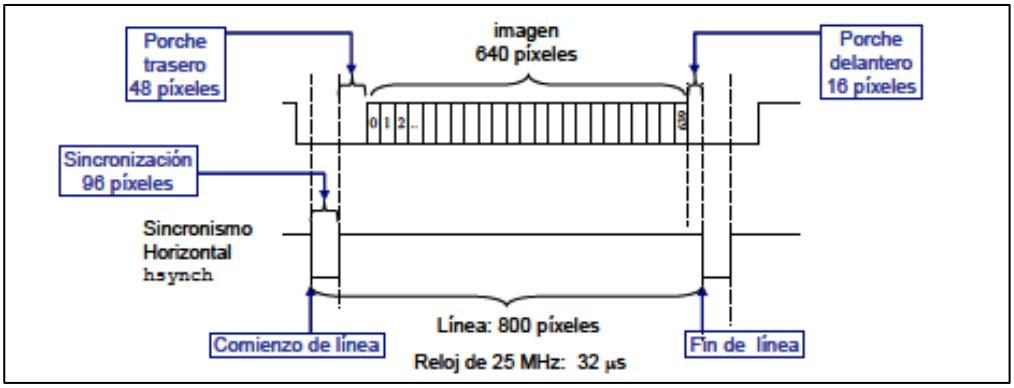


Figura 4.27 Cronograma de la línea horizontal en píxeles para una resolución de 640x800.

Fuente: Diseño de sistemas digitales con VHDL. [12]

Los valores de las señales de sincronismos, los bordes delantero y trasero, la frecuencia de reloj y la zona de visualización, todos estos se encuentran tabulados, así como se indica la Tabla 4.18.

Formato	Reloj MHz	Horizontal (en píxeles)					Vertical (en líneas)				
		Vídeo activo	Porche delantero	Sincr.	Porche trasero	Total	Vídeo activo	Porche delantero	Sincr.	Porche trasero	Total
640x480@60Hz	25	640	16	96	48	800	480	9	2	29	520
640x480@60Hz	25,175	640	16	96	48	800	480	11	2	31	524
800x600@60Hz	40,000	800	40	128	88	1056	600	1	4	23	628
800x600@72Hz	50,000	800	56	120	64	1040	600	37	6	23	666
1024x768@60Hz	65,000	1024	24	136	160	1344	768	3	6	29	806
1024x768@75Hz	75,000	1024	24	136	144	1328	768	3	6	29	806

Tabla 4.18 Valores para diversas resoluciones para los monitores con puerto VGA

Fuente: Diseño de sistemas digitales con VHDL. [12]

Para una resolución de 640x480, una frecuencia de actualización de pantalla de 60Hz y una frecuencia de reloj aplicado de 25MHz, se tiene como la Figura 4.28.

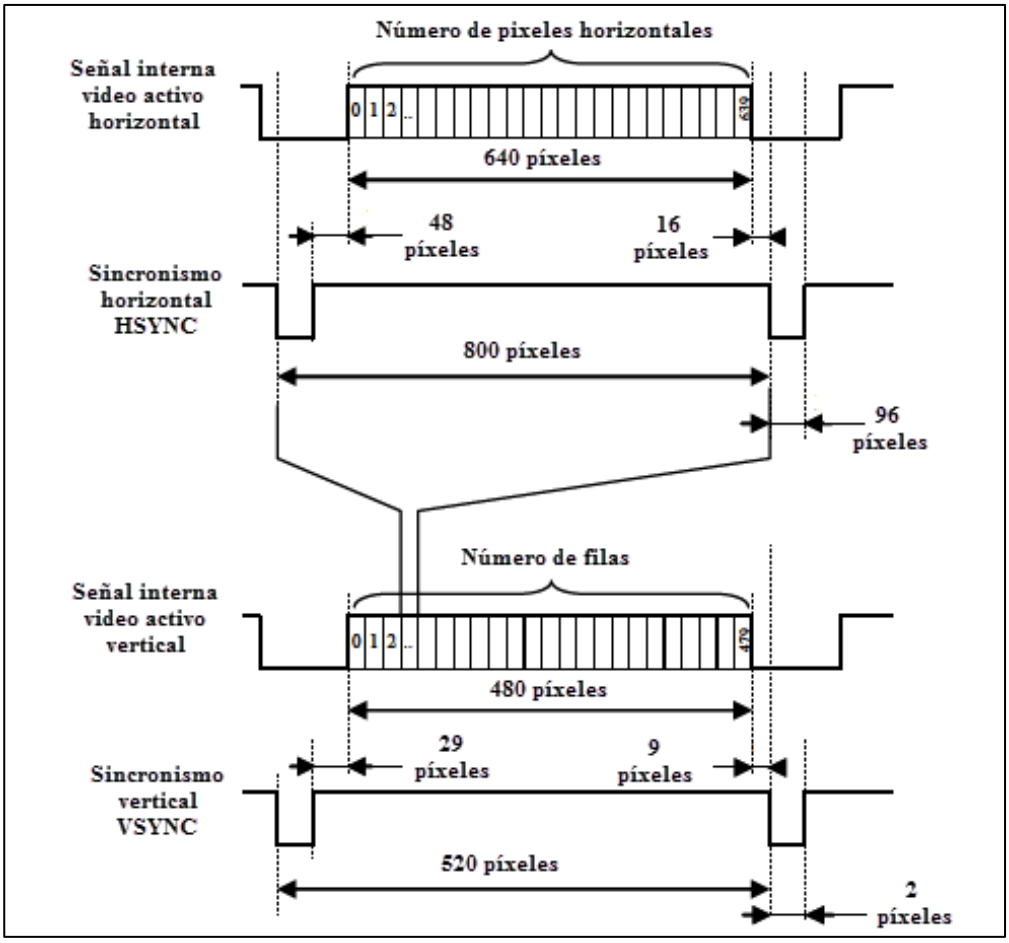


Figura 4.28 Valores en píxeles de la señal de sincronismo horizontal y vertical para una resolución de 640x480.

La información que se envía a los píxeles, determina la tonalidad de los mismos, por medio de los puertos OutRed de 3 bits, OutGreen de 3 bits y OutBlue de 2 bits, la combinación de estos puertos permiten una paleta de 255 colores.

Ahora que se identificaron las variables HSYNC, VSYNC, OutRed, OutGreen y OutBlue se procede a realizar el bloque en VHDL del protocolo de comunicación del puerto VGA.

Se diseña un módulo denominado Monitor_Sincronizador, en donde está contenido dentro de Monitor_Top, así como se muestra en la Figura 4.29.

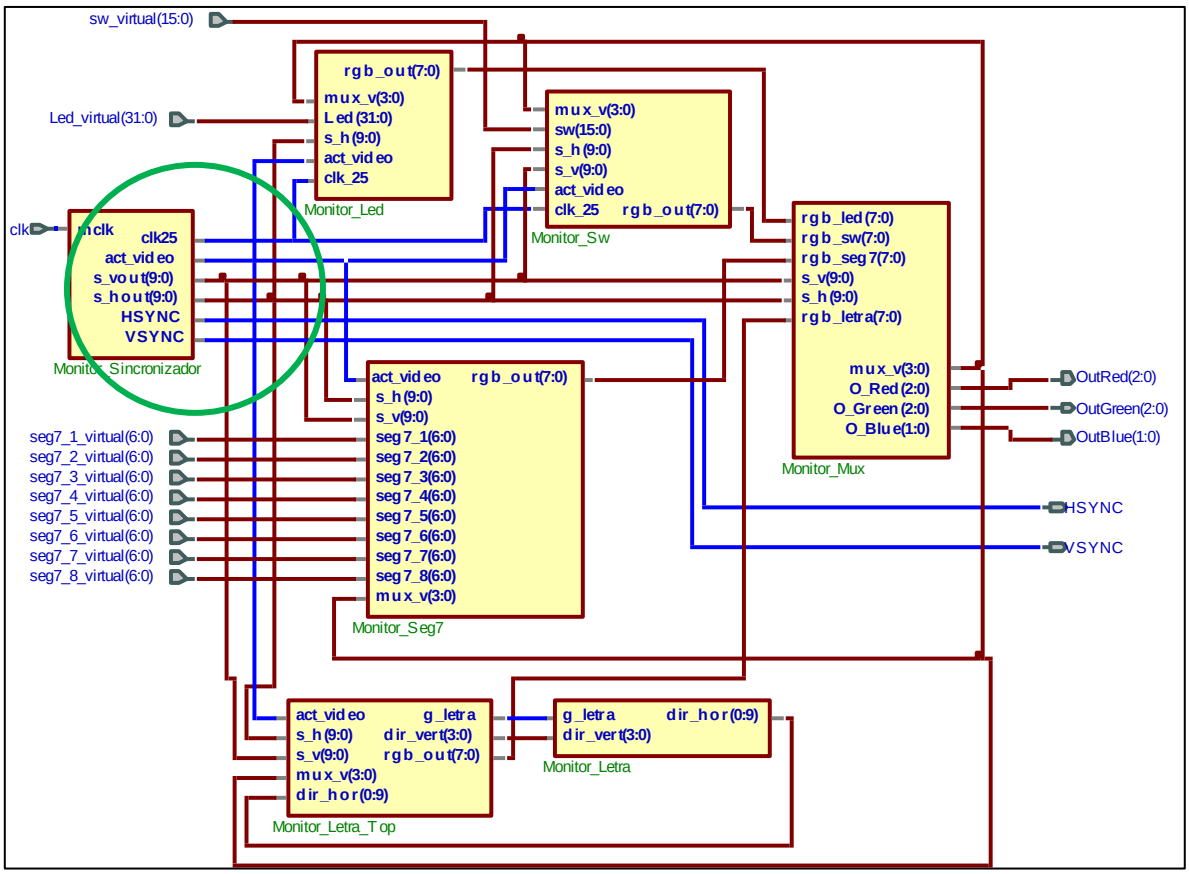


Figura 4.29 Conexión del bloque Monitor_Sincronizador en el bloque Monitor_Top.

Subprograma Monitor_Sincronizador

Este bloque se diseñó para una resolución de 640x480, con una frecuencia de actualización de pantalla de 60Hz y una frecuencia de reloj aplicado de 50MHz, a pesar de que en la Tabla 4.18 se dice que debe ser de 25 MHz, es por ello que se hizo un divisor de frecuencia para llevar de 50MHz a 25MHz.

En la Figura 4.30, se muestra la entidad en un bloque del subprograma que realiza la sincronización del puerto VGA y en la Tabla 2 y 3 se muestran las entradas y salidas de dicha entidad.

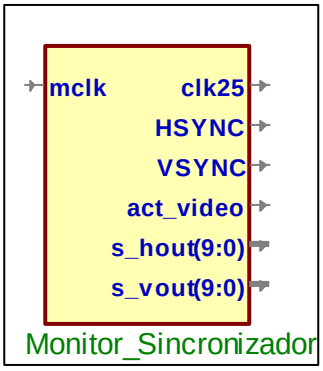


Figura 4.30 Entidad del subprograma Monitor_Sincronizador.

De la Figura 4.31. Se detallan sus variables de entrada en la Tabla 4.19 y sus variables de salida en la tabla Tabla 4.20, del subprograma Monitor_Sincronizador.

Nombre	Tipo	Descripción
Mclk	Std_logic	Señal del reloj del oscilador de la tarjeta basys2.

Tabla 4.19 Descripción de las señales de entrada del subprograma Monitor_Sincronizador.

Nombre	Tipo	Descripción
Clk25	Std_logic_vector (5:0)	Señal de reloj a 25MHz
HSYNC	Std_Logic	Señal sincrónica horizontal
VSYNC	Std_Logic	Señal sincrónica vertical
Act_Video	Std_Logic	Zona de visualización, si es un 1 lógico, indica que el pixel se encuentra en dicha zona
S_Hout	Std_Logic_Vector (9:0)	Contador horizontal que indica la posición del pixel en la línea horizontal
S_Vout	Std_Logic_Vector (9:0)	Contador vertical que indica la posición de la fila en que se encuentra el pixel

Tabla 4.20 Descripción de las señales de salida del subprograma Monitor_Sincronizador.

Con el contador horizontal y vertical, se determina la posición del pixel en la pantalla del monitor. En la Figura 4.31, se describe el funcionamiento del subprograma Monitor_Sincronizador por medio de un diagrama de flujo.

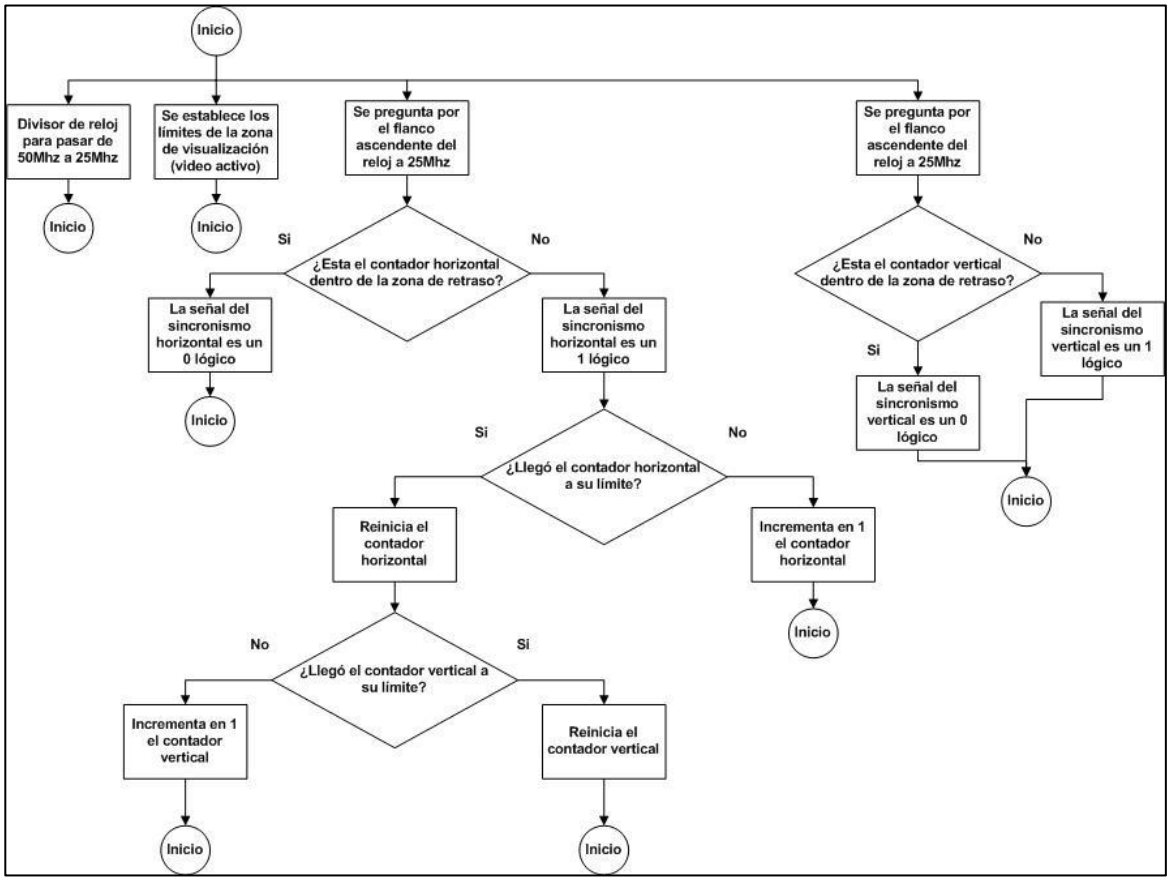


Figura 4.31 Comportamiento del subprograma Monitor_Sincronizador.

Para el contador horizontal y vertical se colocó como punto de origen de ambas señales de sincronismo cuando la zona de visualización empieza, así como se indica en la Figura 4.32.

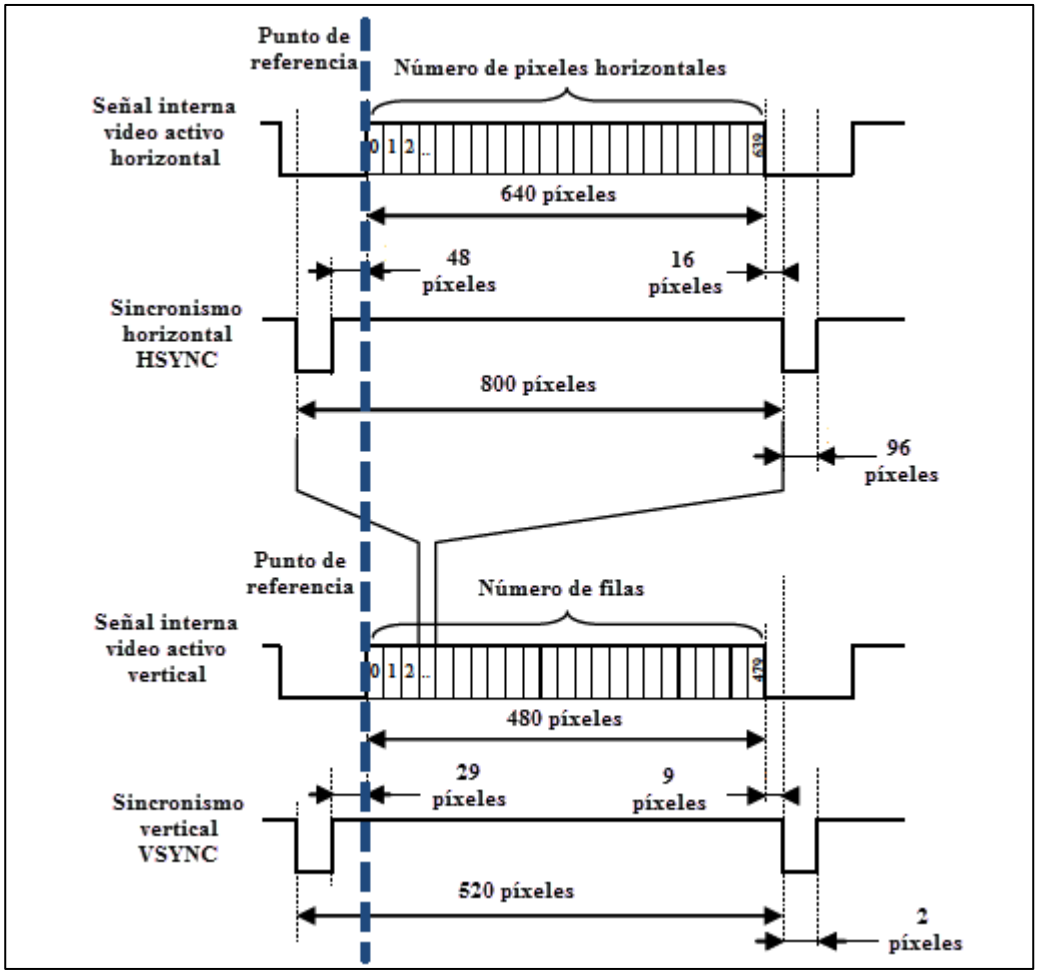


Figura 4.32 Señal de sincronismo horizontal y vertical con punto de referencia, para el contador horizontal y vertical del subprograma Monitor_Sincronizador.

4.5 ENSAYO Y PRUEBA DEL MONITOR Y TECLADO EN CONJUNTO A LA TARJETA BASYS2

Ejemplo VGA Stripes

Al ya tener el subprograma Monitor_Sincronizador, el cual se encarga del protocolo de comunicación del puerto VGA, se verifica su correcto funcionamiento al efectuar el ejemplo 71. VGA stripes de Haskell R., Hanna D. [6], la cual consiste en mostrar en la pantalla del monitor 15 líneas horizontales verdes y rojas.

Se necesita crear dos subprogramas, uno denominado VGA_Stripes que se encarga de trazar las 15 líneas horizontales verdes y rojas, y otro Top_VGA_Stripes que se encarga de realizar el conexionado entre el subprograma VGA_Stripes y el Monitor_Sincronizador junto con las entradas y salidas necesarias de la tarjeta BASYS2.

Subprograma VGA_Stripes

Para la elaboración de este subprograma, esta requiere de la ubicación del pixel tanto en la línea horizontal como el número de fila se encuentra, además de conocer si se encuentra en la zona de visualización, de salida se tienen las variables para la pigmentación del píxel, las señales RGB (OutRed, OutGreen, OutBlue). Así como se muestra en la Figura 4.33.

Figura 4.33 Entidad del subprograma VGA_Stripes.

De la Figura 4.33. Se detallan sus variables de entrada en la Tabla 4.21 y sus variables de salida en la Tabla 4.22, del subprograma VGA_Stripes.

Nombre	Tipo	Descripción
Vidon	Std_Logic	Señal Video activo, indica si se encuentra en la zona de visualización.
Hc	Std_Logic_Vector(9:0)	Contador horizontal que indica la posición del pixel en la línea horizontal
Vc	Std_Logic_Vector(9:0)	Contador vertical que indica la posición de la fila en que se encuentra el pixel

Tabla 4.21 Descripción de las señales de entrada del subprograma VGA_Stripes.

Nombre	Tipo	Descripción
Red	Std_Logic_Vector (2:0)	Señal de salida, bits de color rojo
Green	Std_Logic_Vector(2:0)	Señal sincrónica horizontal
Blue	Std_Logic_Vector(1:0)	Señal sincrónica vertical

Tabla 4.22 Descripción de las señales de salida del subprograma VGA_Stripes.

Trabajando a una resolución de 640x480 y una señal de reloj aplicada de 50MHz, pero esta es llevada a 25Mhz por un divisor de reloj interno que se encuentra en el subprograma Monitor_Sincronizador, tenemos que:

Se quieren 15 bandas de líneas horizontales de color rojo y 15 verdes intercaladas entre sí, dividiendo el número de filas, 480 entre 30 bandas de líneas horizontales, da como resultado que el ancho de la banda tiene que ser de 16 filas. Entonces si tomamos el 4to bit del contador vertical, ese bit representa 2^4=16 filas.

Ahora para cambiar entre dos colores las bandas de líneas horizontales consecutivamente, se asignó a la variable de salida “Red” el valor de 4to bit del contador vertical, a la variable de salida “Green” el mismo valor del 4to bit del contador vertical pero negado y la variable de salida “Blue” se le asigna ceros lógicos, obteniendo como resultado que cuando el 4to bit del contador vertical es un cero lógico, entonces pigmenta el ancho de banda de color verde, si es un uno lógico pigmenta de color rojo y así sucesivamente. El código del subprograma se encontrará en el apéndice C.

Subprograma Top_VGA_Stripes

En este subprograma se realiza el conexionado entre los subprogramas VGA_Stripes, Monitor_Sincronizador y los pines de la tarjeta BASYS2 así como en la Figura 4.34.

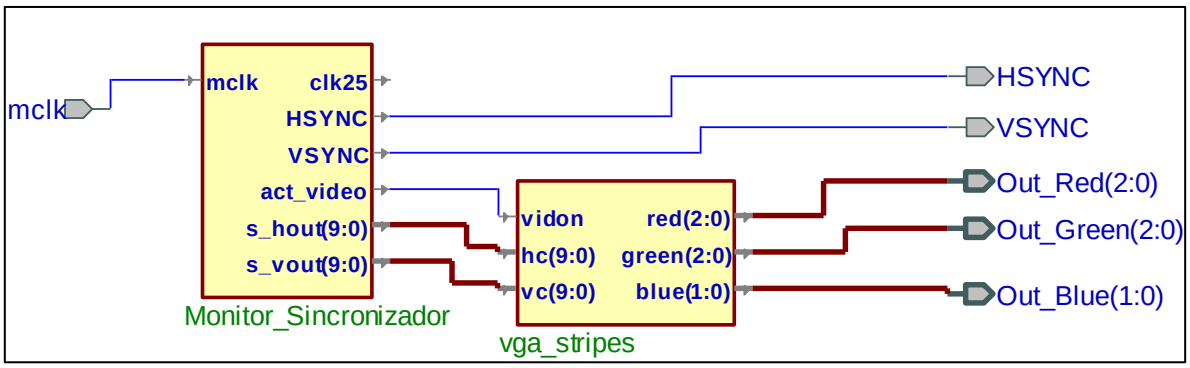


Figura 4.34 Esquema de conexión del Subprograma Top_VGA_Stripes.

De la Figura 4.35. Se muestra la entidad y se detallan sus variables de entrada en la Tabla 4.23 y sus variables de salida en la Tabla 4.24, del subprograma VGA_Stripes.

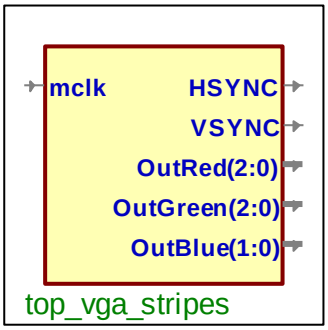


Figura 4.35 Entidad del subprograma VGA_Stripes.

Nombre	Tipo	Descripción
Mclk	Std_Logic	Señal del reloj aplicado a 50MHz

Tabla 4.23 Descripción de las señales de entrada del subprograma VGA_Stripes.

Nombre	Tipo	Descripción
HSYNC	Std_Logic	Señal de sincronismo horizontal
VSYNC	Std_Logic	Señal de sincronismo vertical
Red	Std_Logic_Vector (2:0)	Señal de salida, bits de color rojo

Green	Std_Logic_Vector(2:0)	Señal de salida, bits de color verde
Blue	Std_Logic_Vector(1:0)	Señal de salida, bits de color azul

Tabla 4.24 Descripción de las señales de salida del subprograma VGA_Stripes.

Al tener los subprogramas, se compilan y se sintetiza el diseño. Antes de realizar la implementación si se está usando el oscilador interno de la tarjeta BASYS2, entonces se debe usar el archivo .ucf denominado Basys2_Mon_OsciladorInt.ucf, esta se encuentra en el Apéndice D.

En la Figura 4.36 se muestra el resultado en la pantalla usando el oscilador interno de la tarjeta BASYS2, de esto se nota que en los bordes laterales de las bandas de colores, inestabilidad de la señal, esto ocurre ya que el oscilador interno de la tarjeta BASYS2 es inestable para aplicaciones con el puerto VGA, en el manual de usuario de la tarjeta BASYS2, recomiendan el uso de un oscilador de cristal externo si se quiere trabajar con el puerto VGA, ya que el oscilador de cristal es más estable.

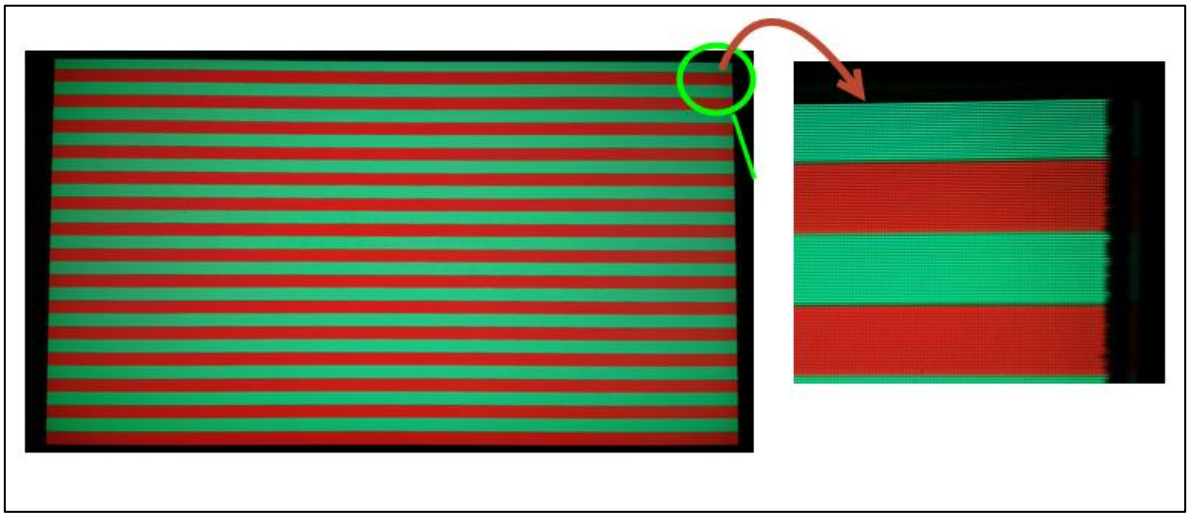


Figura 4.36 Resultado del Ejemplo VGA_Stripes usando oscilador interno.

Al realizar el mismo diseño pero en vez de usar el oscilador interno de la tarjeta BASYS2, se usa un oscilador de cristal externo de 50MHz, con las especificaciones descritas en el datasheet que se encuentra ubicado en el Apéndice D, se conexiona con los

respectivos pines de la tarjeta BASYS2 junto con el oscilador externo. Además se debe cambiar el archivo.ucf, ya que el pin para el oscilador externo es distinto a la del oscilador interno, en la tarjeta BASYS2. El archivo .ucf a usar en este caso es Basys2_Mon_OsciladorExt, que se encuentra en el Apendice D.

En la Figura 4.37, se observa que ahora en los bordes laterales de las bandas no se refleja ruido sino una señal estable y fija.

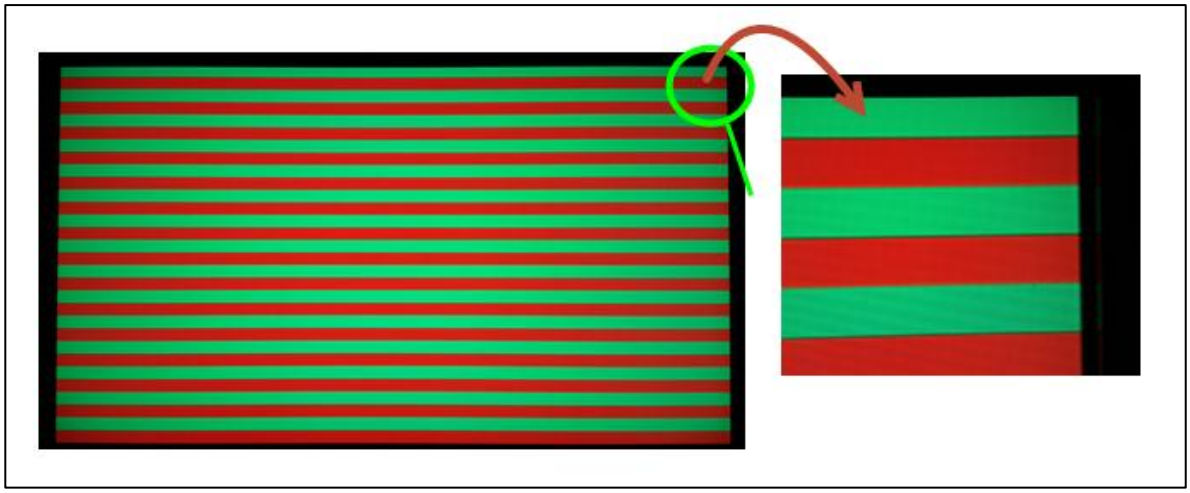


Figura 4.37 Resultado del Ejemplo VGA_Stripes usando oscilador externo.

Ejemplo VGA_Pulsador

Se quiere que al pulsar la Tecla “Enter” se muestre en la pantalla del monitor un cuadrado de color verde, y al no estar pulsada dicha tecla, el mismo cuadrado sea de color azul.

Para el desarrollo de este diseño se requiere los subprogramas que realizan el protoolo de comunicación del puerto VGA y el puerto PS/2 conectado a un monitor y un teclado respectivamente, dichos subprogramas son Monitor_Sincronizador y Teclado_Lectura.

Ahora se requiere de dos subprogramas adicionales, uno en donde se realice el cuadrado en el monitor, el control de ella y su pigmentacíon, este subprograma se

denominará VGA_Pulsador, y el otro subprograma realizará el conexionado de todos los subprogramas junto con los pines de la tarjeta BASYS2.

Subprograma VGA_Pulsador

En la Figura 4.38. Se muestra la entidad del subprograma VGA_Pulsador. Esta se encarga de ubicar la sección rectangular en la pantalla, además de controlarla de modo que al leer la tecla "Enter" esta cambie de color a verde, sino se pigmenta de color azul.

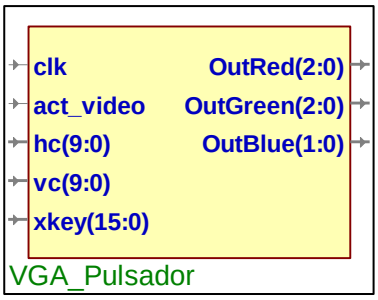


Figura 4.38 Entidad del subprograma VGA_Pulsador.

De la Figura 4.39. Se detallan sus variables de entrada en la Tabla 4.25 y sus variables de salida en la Tabla 4.26, del subprograma VGA_Pulsador.

Nombre	Tipo	Descripción
Act_Video	Std_Logic	Señal que indica si el pixel está en la zona de visualización
Hc	Std_Logic_Vector(9:0)	Contador horizontal que indica la posición del pixel en la línea horizontal
Vc	Std_Logic_Vector(9:0)	Contador vertical que indica la posición de la fila en que se encuentra el pixel
Xkey	Std_Logic_Vector(15:0)	Código scan de 2 Bytes

Tabla 4.25 Descripción de las señales de entrada del subprograma VGA_Pulsador.

Nombre	Tipo	Descripción
OutRed	Std_Logic_Vector(2:0)	Señal que pigmenta el pixel de color rojo
OutGreen	Std_Logic_Vector(2:0)	Señal que pigmenta el pixel de color verde
OutBlue	Std_Logic_Vector(1:0)	Señal que pigmenta el pixel de color azul

Tabla 4.26 Descripción de las señales de salida del subprograma VGA_Pulsador.

En la Figura 4.39. Se describe el funcionamiento del subprograma VGA_Pulsador.

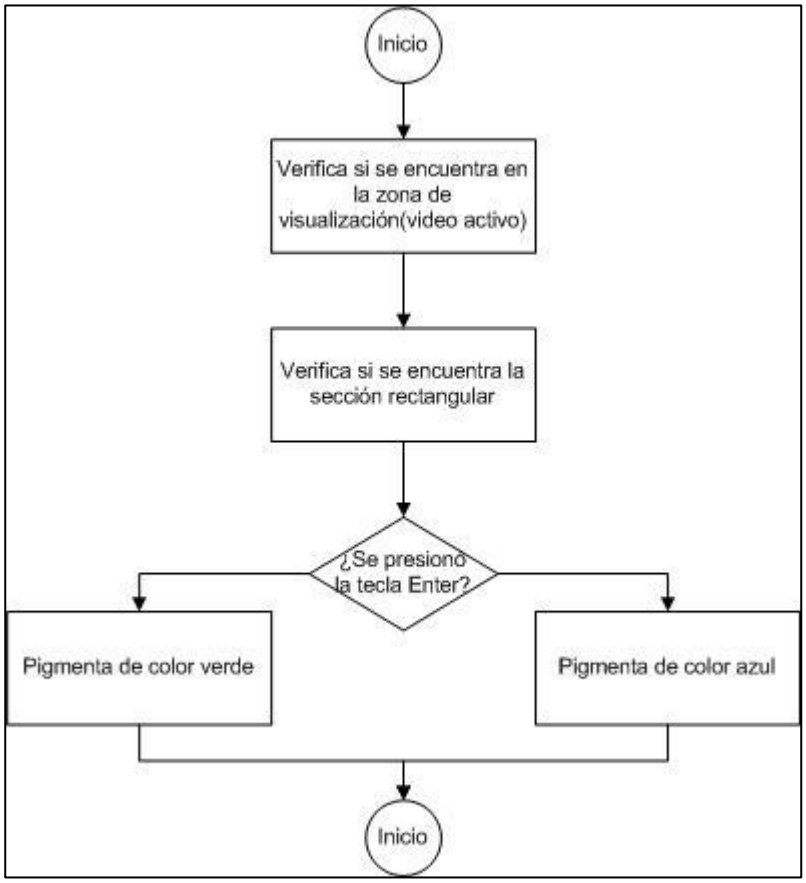


Figura 4.39 Comportamiento del subprograma VGA_Pulsador.

Subprograma Top_VGA_Pulsador

En este subprograma se realiza el conexionado entre los subprogramas VGA_Pulsador, Monitor_Sincronizador, Teclado_Lectura y los pines de la tarjeta BASYS2, así como se muestra en la Figura 4.40.

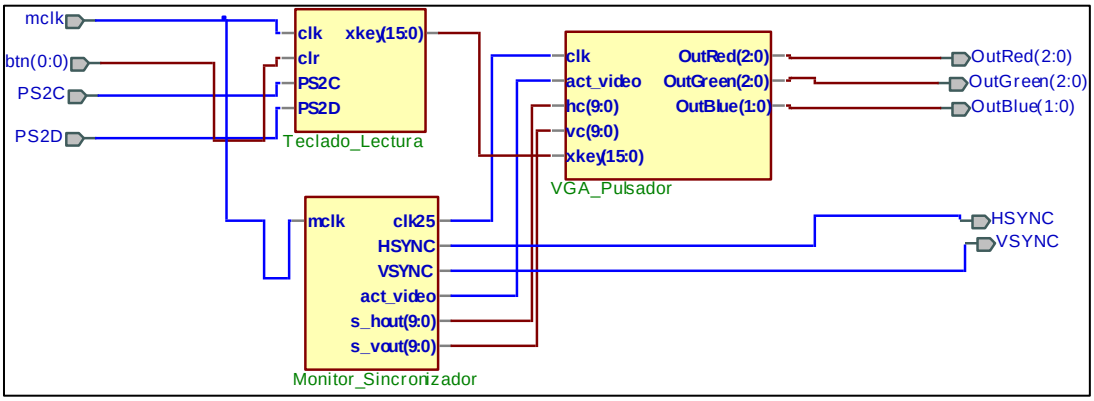


Figura 4.40 Esquema de conexión del Subprograma Top_VGA_Pulsador.

De la Figura 4.41. Se muestra la entidad del subprograma Top_VGA_Pulsador y detallan sus variables de entrada en la Tabla 4.27 y sus variables de salida en la Tabla 4.28, del subprograma VGA_Pulsador.

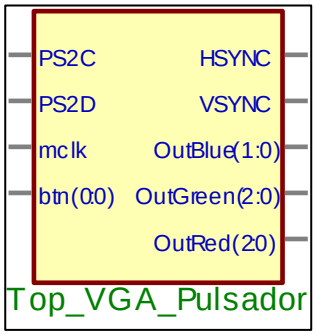


Figura 4.41 Entidad del subprograma Top_VGA_Pulsador.

Nombre	Tipo	Descripción
Mclk	Std_Logic	Señal del reloj aplicado a 50MHz
Btn(0:0)	Std_Logic_Vector	Pulsador
PS2C	Std_Logic	Señal de la data del teclado.
PS2D	Std_Logic	Señal del reloj del teclado.

Tabla 4.27 Descripción de las señales de entrada del subprograma Top_VGA_Pulsador.

Nombre	Tipo	Descripción
HSYNC	Std_Logic	Señal de sincronismo horizontal
VSYNC	Std_Logic	Señal de sincronismo vertical
OutRed	Std_Logic_Vector(2:0)	Señal de salida, bits de color rojo
OutGreen	Std_Logic_Vector(2:0)	Señal de salida, bits de color verde
OutBlue	Std_Logic_Vector(1:0)	Señal de salida, bits de color azul

Tabla 4.28 Descripción de las señales de salida del subprograma Top_VGA_Pulsador.

Al igual que el ejemplo anterior, dependiendo del oscilador a usar, interno o externo, se elige su respectivo archivo .ucf. Si se va a usar el oscilador interno de la tarjeta BASYS2, entonces debe seleccionar el archivo Basys2_Mon_Tec_OsciladorInt.ucf, que se encuentra en el Apendice D. Si se va a usar el oscilador externo, entonces se selecciona el archivo Basys2_Mon_Tec_OsciladorExt, que se encuentra en el Apendice D.

4.6 DESARROLLO DE UNA INTERFAZ GRÁFICA, APLICACIONES Y MANUAL DE USUARIO DE LA TARJETA BASYS 2

El propósito de efectuar la comunicación con los puertos PS/2 y VGA de la tarjeta Basys2, es la de facilitar al usuario un mayor número de entradas y salidas, debido a la

necesidad de desarrollar aplicaciones que requieran mayor número de E/S de las que suministra físicamente la tarjeta BASYS2.

Es por esto que se realiza el diseño de una interfaz gráfica adaptada al usuario, contando con los elementos de la estructura de la Figura 4.42, que para este caso se desea que disponga en el monitor con: 32 leds, 16 indicadores estilo interruptores y ocho (8) displays 7-Segmentos.

Figura 4.42 Interfaz modelo a realizar en un monitor.

Como se quiere incorporar a este nivel distintos elementos de interacción en la pantalla del monitor, existe el riesgo de que coincidan las señales de RGB en el puerto, alterando la imagen en el monitor.

Subprograma Monitor_Mux

Por esta razón se diseñó la entidad Monitor_Mux, encargada de organizar por zonas el área del monitor que se desea dibujar. Este subprograma se encuentra contenido en el bloque Monitor_Top, como se muestra en la Figura 4.43. El diseño de entradas y salidas de este bloque se muestra en la Figura 4.44.

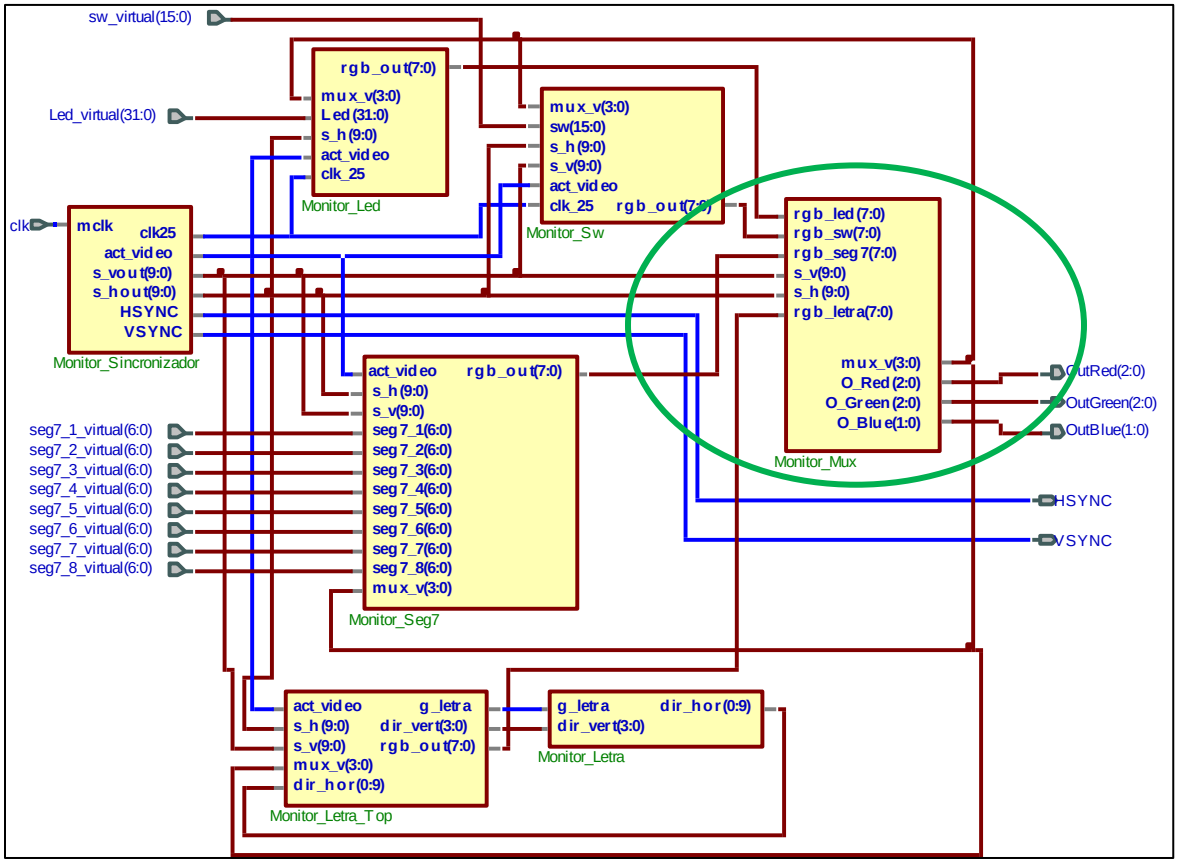


Figura 4.43 Conexión de Monitor_Mux en el bloque Monitor_Top.

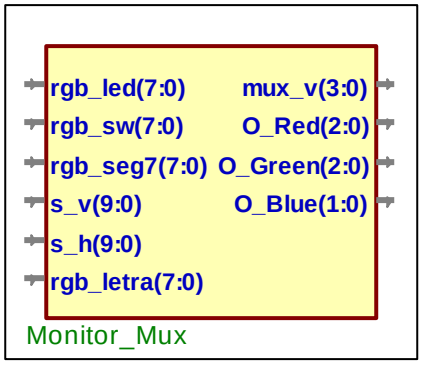


Figura 4.44 Bloque Monitor_Mux.

Del bloque se detallan sus variables de entrada en la Tabla 4.29 y sus variables de salida en la Tabla 4.30.

Nombre	Tipo	Descripción
rgb_letra	Std_logic_vector (7:0)	Señal RGB de las letras
rgb_led	Std_logic_vector (7:0)	Señal RGB de los Leds
rgb_sw	Std_logic_vector (7:0)	Señal RGB de los switches
rgb_seg7	Std_logic_vector (7:0)	Señal RGB de los 7 segmentos
s_v	Std_logic_vector (9:0)	Posición del píxel vertical
s_h	Std_logic_vector (9:0)	Posición del pixel horizontal

Tabla 4.29 Descripción de las señales de entrada del bloque Monitor_Mux.

Nombre	Tipo	Descripción
mux_v	Std_logic_vector (3:0)	Señal de salida del multiplexor
O_Red	Std_logic_vector (2:0)	Señal de salida, bits de color rojo
O_Green	Std_logic_vector (2:0)	Señal de salida, bits de color verde
O_Blue	Std_logic_vector (1:0)	Señal de salida, bits de color azul

Tabla 4.30 Descripción de las señales de salida del bloque Monitor_Mux.

Con la señal de salida mux_v se obtienen organizados los tramos disponibles del monitor, lo que le indica a los demás subprogramas de la interfaz donde operar. También cuenta en la salida con la señal O_Red, O_Green y O_Blue, propias para pigmentar cada píxel correspondiente hacia el monitor mediante el puerto VGA. En el Figura 4.45, se muestra el comportamiento de la entidad Monitor_Mux.

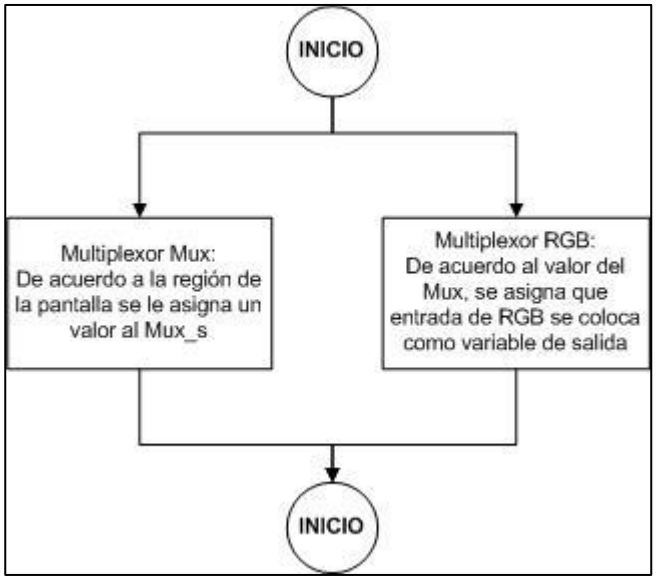


Figura 4.45 Comportamiento de la entidad Monitor_Mux.

Este subprograma dispone en la entrada al contador horizontal (s_h) y vertical (s_v) para reconocer en todo momento la ubicación del pixel en la pantalla del monitor, y así organizar en que segmento del monitor (Tabla 4.31). Se le permitirá transmitir la señal de rbg_out de cada sub-programa de la interfaz, de acuerdo a la Tabla 4.32.

mux_v	s_v	s_h	Descripción del segmento
"0000"	(31 - 40)	(266 - 305)	Título "leds"
"0001"	(51 - 70)	(51 - 595)	1ra hilera de led
"0010"	(76 - 85)	(41 - 600)	1ra descripción hilera de led
"0011"	(96 - 115)	(51 - 595)	2da hilera de led
"0100"	(121 - 130)	(41 - 605)	2da descripción hilera de led
"0101"	(151 - 160)	(251 - 420)	Titulo interruptores "switches"
"0110"	(171 - 210)	(51 - 595)	Hilera de interruptor
"0111"	(216 - 225)	(41 - 600)	Descripción hilera de interruptor
"1000"	(246 - 255)	(251 - 420)	Título "7display"
"1001"	(264 - 310)	(46 - 505)	Hilera de 7segmento

Tabla 4.31 Descripción de las zonas de video establecidas por Monitor_Mux.

rgb_out	Segmento indicado por mux_v
rgb_letra	“0000”, “0010”, “0100”, “0101”, “0111”, “1000”, “1010”.
rgb_led	“0001”, “0011”.
rgb_sw	“0110”.
rgb_seg7	“1001”.

Tabla 4.32 Organización establecida por Monitor_Mux para la salida de video

Subprograma Monitor_Led

Para ubicar los Leds en la interfaz del monitor se diseñó la entidad Monitor_Led, encargada de dibujar en la pantalla un par de hileras de 16 cuadros de 20x20 píxeles cada uno y con separación de 15 píxeles entre cada cuadro, en representación de un led, que estará de un color neutro de estar desactivo y un color claro de estar activo. Este subprograma se encuentra contenido en Monitor_Top, como se muestra en la Figura 4.46; y el diseño de entradas y salidas de este bloque en la Figura 4.47.

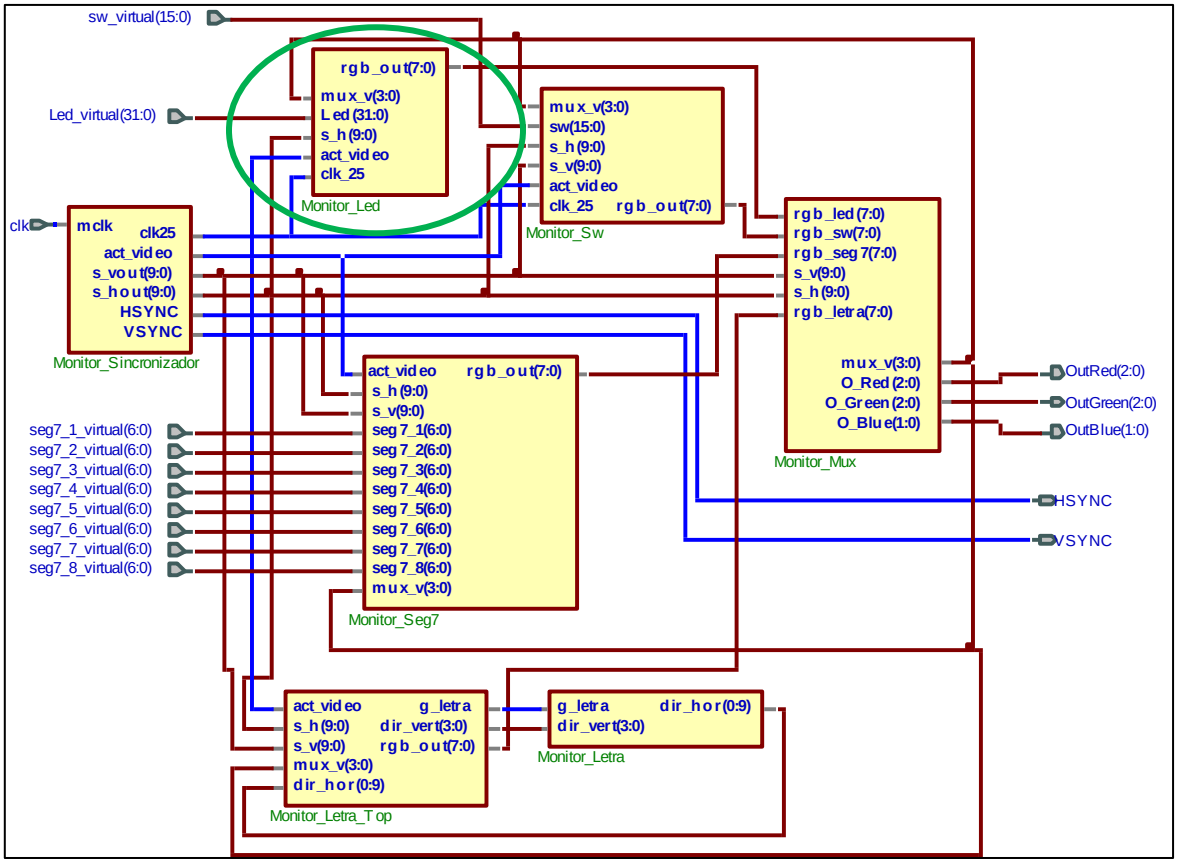


Figura 4.46 Conexión de Monitor_Led en el diseño de Monitor_Top.

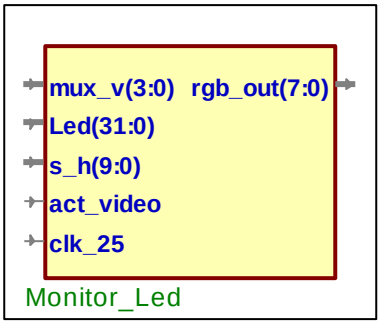


Figura 4.47 Bloque Monitor_Led.

Del bloque se detallan sus variables de entrada en la Tabla 4.33 y sus variables de salida en la Tabla 4.34.

Nombre	Tipo	Descripción
clk	Std_logic	Señal del reloj a 50Mhz
act_video	Std_logic	Señal que indica si se encuentra en la matriz visual
s_h	Std_logic_vector (9:0)	Posición del pixel horizontal
mux_v	Std_logic_vector (3:0)	Señal de multiplexor
Led	Std_logic_vector (31:0)	Leds

Tabla 4.33 Descripción de las señales de entrada del bloque Monitor_Led.

Nombre	Tipo	Descripción
rgb_out	Std_logic_vector (7:0)	Señal de salida RGB de los Leds

Tabla 4.34 Descripción de las señales de salida del bloque Monitor_Led.

Cuenta en la entrada con la señal de s_h para determinar en qué punto de la línea horizontal va a pigmentar el pixel, en conjunto con la señal mux_v que determina la región en la que estará operativo este sub-programa; dispone de la señal act_video para asegurar de que este en la zona activa de video del monitor y la señal de Led, la cual es enviada desde el programa del usuario o cualquier otro programa, como petición para encender el led de la interfaz del monitor.

También cuenta en la salida con la señal rgb_out, la cual sirve para informar al sub-programa Monitor_Mux el color a transmitir al puerto VGA en el pixel correspondiente. Como se muestra en la Figura 4.48.

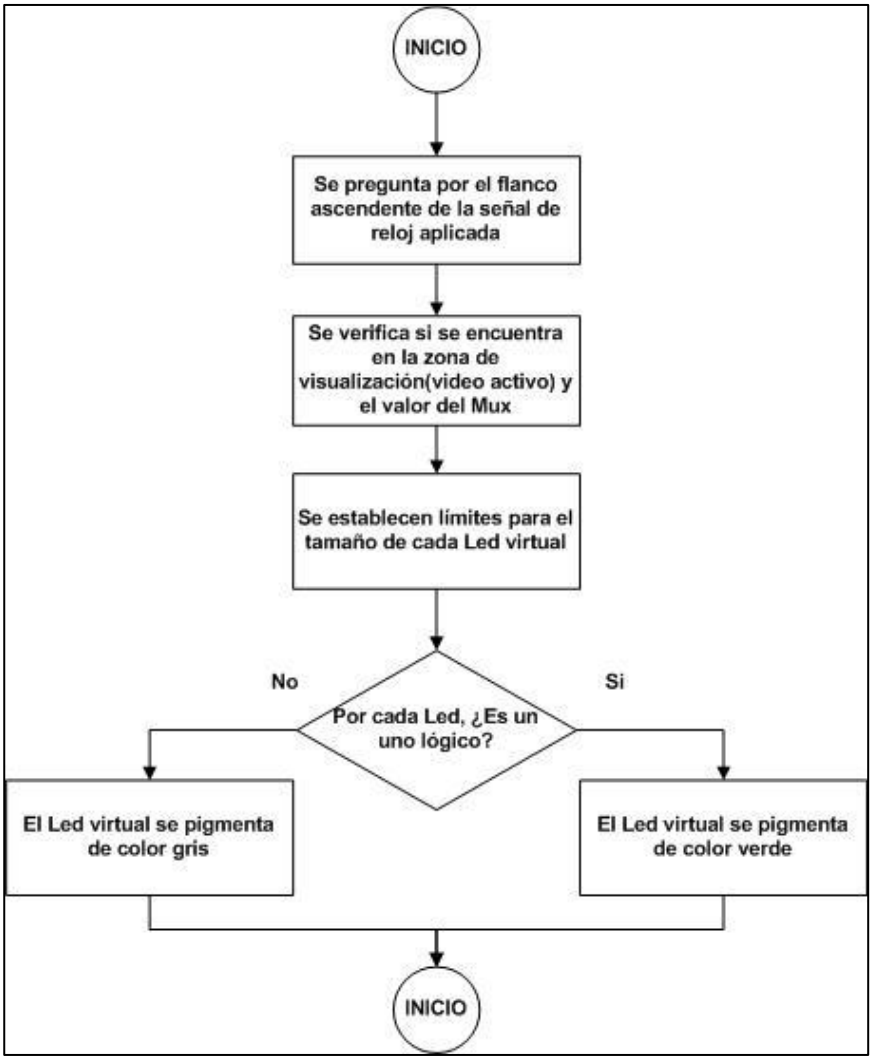


Figura 4.48 Comportamiento de la entidad Monitor_Led.

Donde se indica la lógica asociada del subprograma Monitor_Led para representar un led en el monitor, esta misma lógica la repite para toda la hilera de 16 leds y luego con la segunda hilera de leds. Lo que se representa como leds virtuales en el monitor, mostrados en la Figura 4.49.

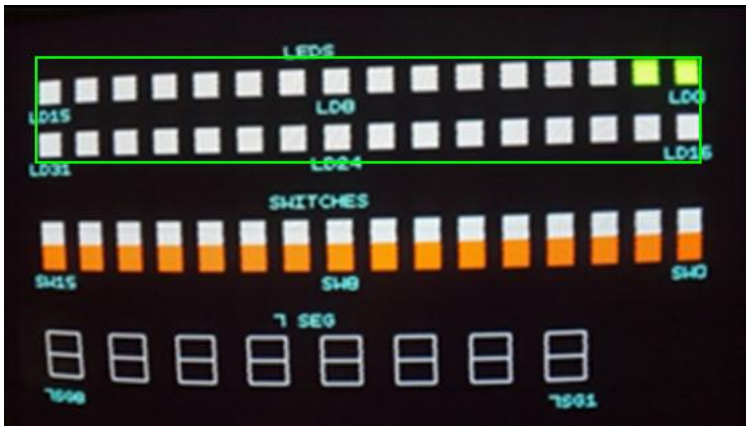


Figura 4.49 Leds virtuales representados en el monitor.

Subprograma Monitor_Sw

Del mismo modo que el subprograma Monitor_Led se orienta la entidad Monitor_Sw. Pero en este caso se dibuja en la pantalla una hilera de 16 rectángulos de 20x40 pixeles y con separación de 15 píxeles entre cada rectángulo, en representación de un indicador de cambio de posición tipo interruptor, este rectángulo se compone de dos bloques de 20x20 píxeles y la lógica se presenta en la Tabla 4.35.

Posición del indicador	Activo	Desactivo
Cuadro superior	Color Claro	Color Neutro
Cuadro inferior	Color Neutro	Color Opaco

Tabla 4.35 Lógica del indicador tipo Switches.

Este subprograma se encuentra contenido en el bloque Monitor_Top, como se muestra en la Figura 4.50; y el diseño de entradas y salidas de este bloque en la Figura 4.51.

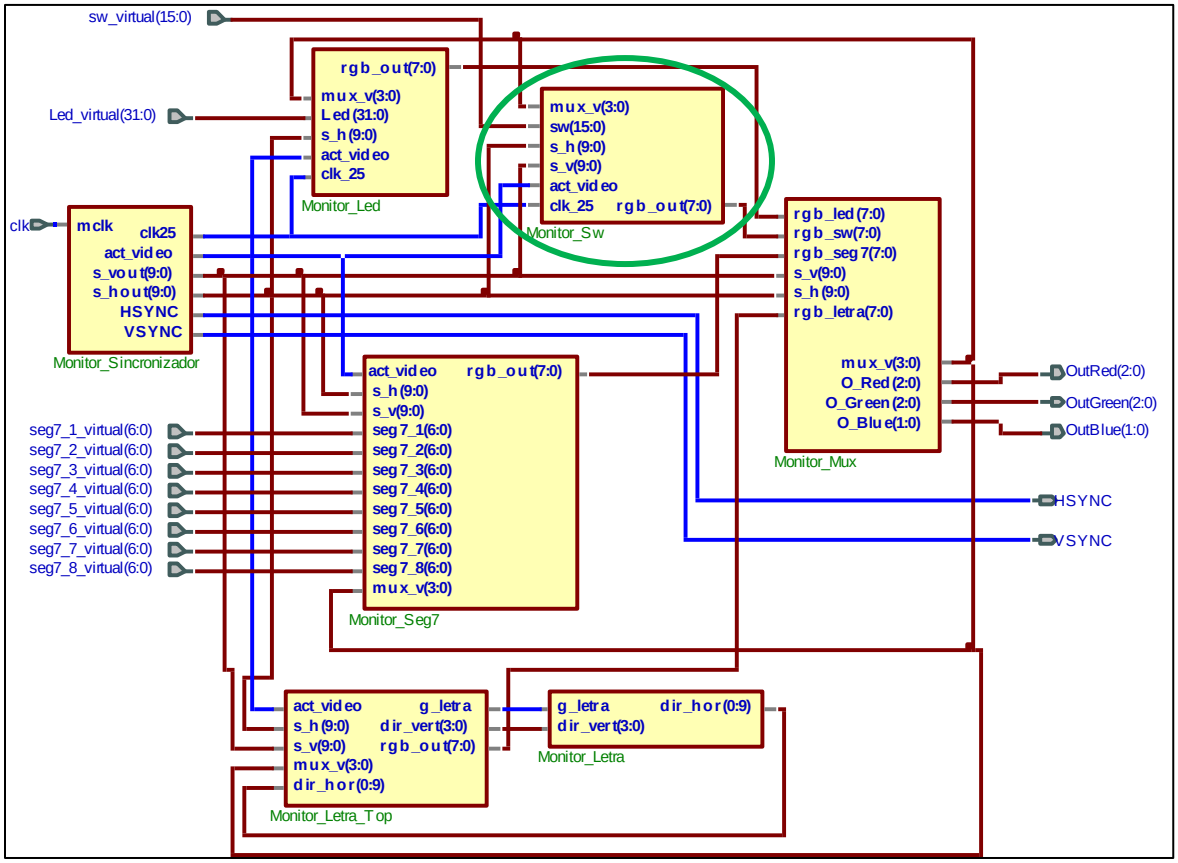


Figura 4.50 Conexión de Monitor_Sw en el diseño de Monitor_Top.

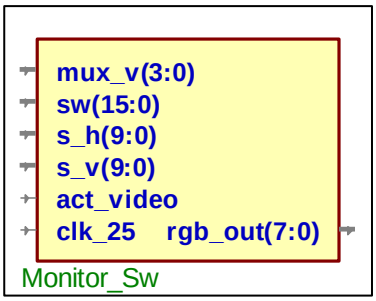


Figura 4.51 Bloque Monitor_Sw.

Del bloque se detallan sus variables de entrada en la Tabla 4.36 y sus variables de salida en la Tabla 4.37

Nombre	Tipo	Descripción
clk	Std_logic	Señal del reloj a 50Mhz
act_video	Std_logic	Señal que indica si se encuentra en la matriz visual
s_h	Std_logic_vector (9:0)	Posición del pixel horizontal
s_v	Std_logic_vector (9:0)	Posición del pixel vertical
mux_v	Std_logic_vector (3:0)	Señal de multiplexor
sw	Std_logic_vector (15:0)	Señal de indicador tipo Switches

Tabla 4.36 Descripción de las señales de entrada del bloque Monitor_Sw.

Nombre	Tipo	Descripción
rgb_out	Std_logic_vector (7:0)	Señal de salida RGB de los Switches

Tabla 4.37 Descripción de las señales de salida del bloque Monitor_Sw.

Mantiene la misma lógica que el sub-programa Monitor_Led, de la cual, la única novedad en sus entradas y salidas es la señal de s_v, puesto que no basta con pigmentar toda la región permitida por el mux_v, sino que debe diferenciar entre el cuadro superior y el cuadro inferior de cada indicador tipo switches.

De la misma forma cuenta en la salida con la señal rgb_out, la cual informa al subprograma Monitor_Mux el color a transmitir al puerto VGA en el pixel en que se encuentre. Dibujando en el monitor los leds (Figura 4.52).

Figura 4.52 indicadores Switches virtuales representados en el monitor.

Subprograma Monitor_Seg7

Para disponer de los display 7-Segmentos en la interfaz gráfica se elabora el diseño del bloque Monitor_Seg7, la cual realiza la representación de ocho (8) siete segmentos en la pantalla, manteniendo la misma lógica que los display de 7-Segmentos físicos de la tarjeta, diferenciándose en que no son de ánodo común, sino de cátodo común, es decir que cada segmento se activa con un ‘uno’ lógico y con la ventaja de que estos 7-Segmentos se mantienen fijos en la pantalla sin necesidad de activarlos individualmente y sin emplear divisores de frecuencia para obtener algún número mayor a un dígito. Este subprograma se encuentra contenido en el bloque Monitor_Top, como se muestra en la Figura 4.53 y el diseño de E/S de este bloque se muestra en la Figura 4.54.

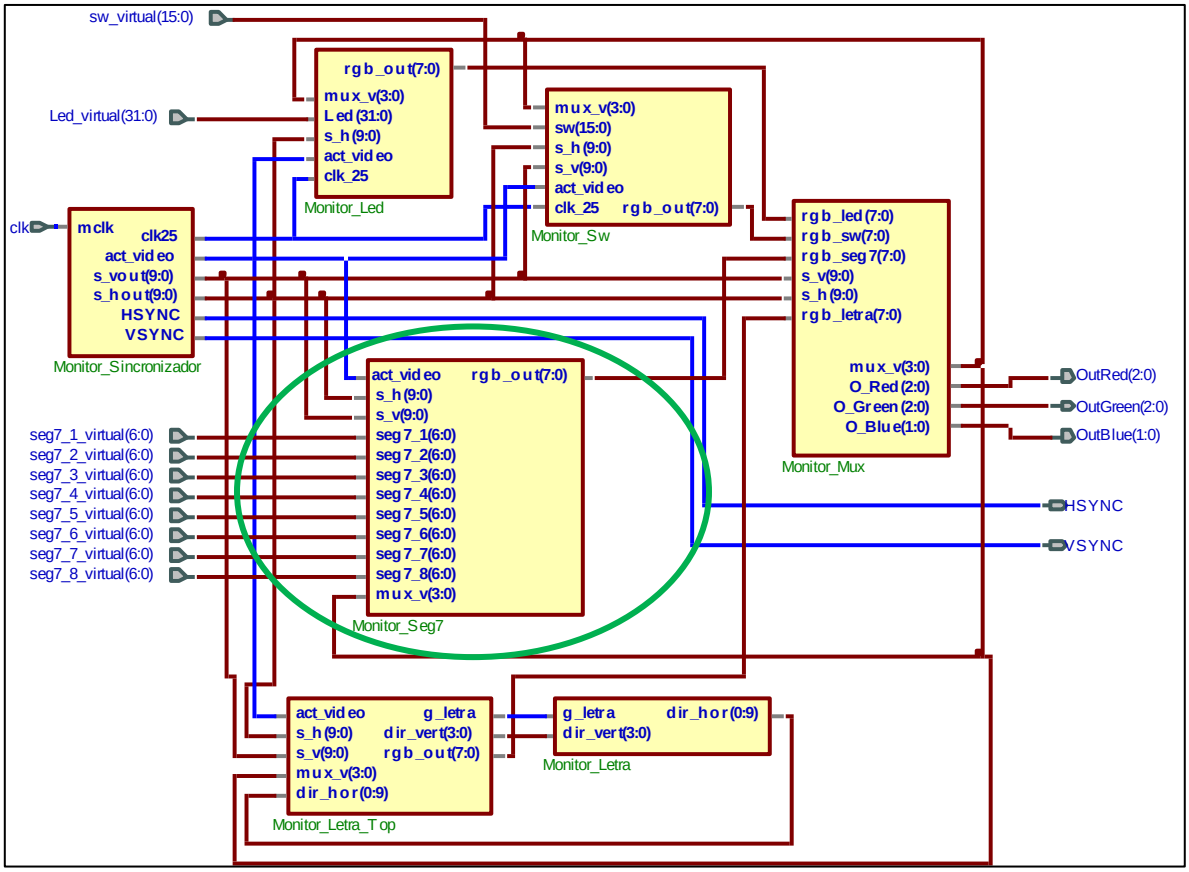


Figura 4.53 Conexión de Monitor_Seg7 en el diseño de Monitor_Top.

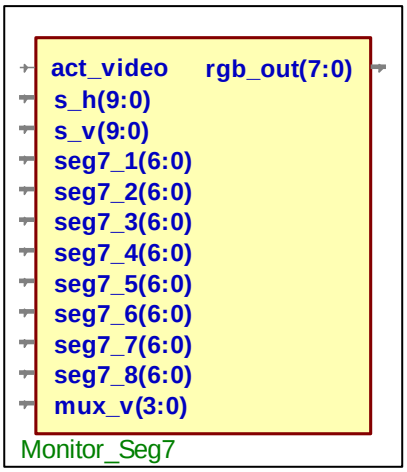


Figura 4.54 Bloque Monitor_Seg7.

Del bloque se detallan sus variables de entrada en la Tabla 4.38 y sus variables de salida en la Tabla 4.39.

Nombre	Tipo	Descripción
act_video	Std_logic	Señal que indica si se encuentra en la matriz visual
mux_v	Std_logic_vector (3:0)	Señal de multiplexor
seg7_1	Std_logic_vector (6:0)	Señal 7 segmento 1
seg7_2	Std_logic_vector (6:0)	Señal 7 segmento 2
seg7_3	Std_logic_vector (6:0)	Señal 7 segmento 3
seg7_4	Std_logic_vector (6:0)	Señal 7 segmento 4
seg7_5	Std_logic_vector (6:0)	Señal 7 segmento 5
seg7_6	Std_logic_vector (6:0)	Señal 7 segmento 6
seg7_7	Std_logic_vector (6:0)	Señal 7 segmento 7
seg7_8	Std_logic_vector (6:0)	Señal 7 segmento 8
s_v	Std_logic_vector (9:0)	Posición del píxel vertical
s_h	Std_logic_vector (9:0)	Posición del pixel horizontal

Tabla 4.38 Descripción de las señales de entrada del bloque Monitor_Seg7.

Nombre	Tipo	Descripción
rgb_out	Std_logic_vector (7:0)	Señal de salida RGB de los 7 segmentos

Tabla 4.39 Descripción de las señales de salida del bloque Monitor_Seg7.

Este bloque dispone en la entrada con las señales s_h y s_v, act_video y mux_v para reconocer en todo momento el pixel horizontal o vertical del monitor en donde se sitúa y verificar de esta manera que se encuentre en una zona permitida de video. Y además cuenta en la entrada con la señal de ocho (8) Displays 7-Segmentos, que sirven a este subprograma para identificar qué display se pigmentara en caso de que se encuentre en la zona de la pantalla reservada por el subprograma Monitor_Mux. El comportamiento de la entidad Monitor_Seg7 se muestra en la Figura 4.55.

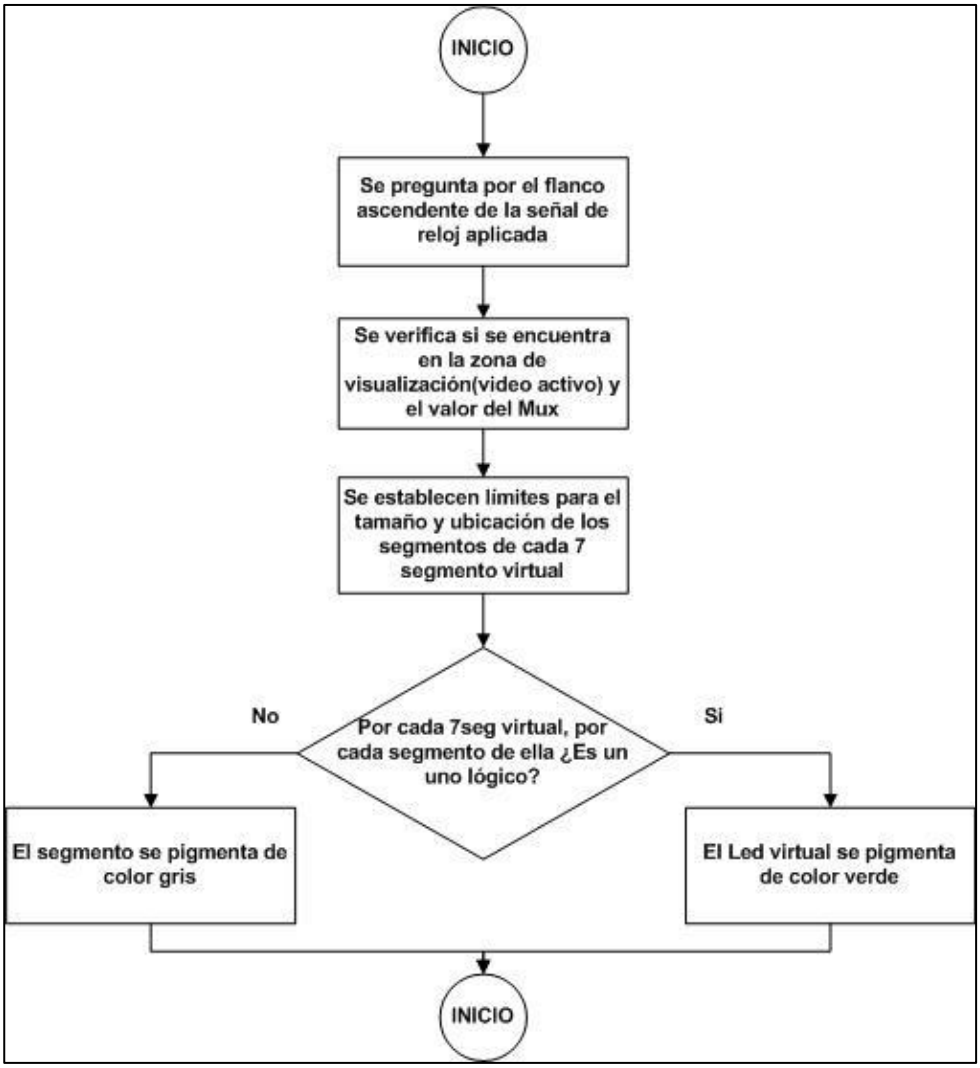


Figura 4.55 Comportamiento de la entidad Monitor_Seg7.

Donde se indica la rama a seguir para pigmentar de un color claro, de estar activo, un pixel del display 7-Segmento (Figura 4.56) a través de la señal de salida rbg_out.

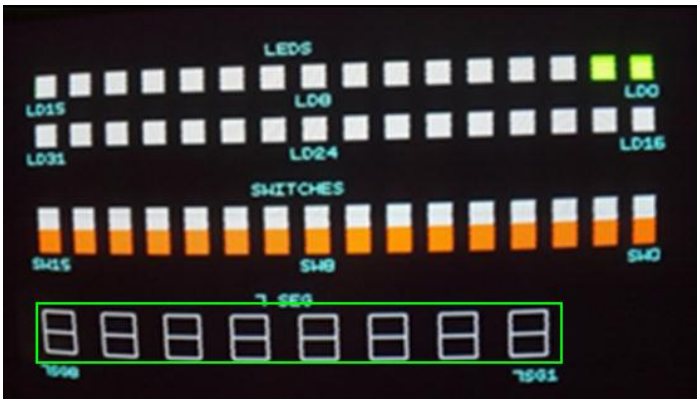


Figura 4.56 Display 7-Segmentos virtuales representados en el monitor.

Subprograma Monitor_Letra_Top

El subprograma Monitor_Letra_Top es una plantilla que realiza el ensamblaje de la letra, empleando las variables de Monitor_Letra, a fin de trazar en los segmentos reservados para las letras cada una de las palabras de la interfaz, como lo son; el título de los leds “Leds”, de los indicadores de cambio de posición “Switches”, de los display 7-Segmentos “7Display”, como la identificación de los primeros y últimos dispositivos. Este subprograma se encuentra contenido en el bloque Monitor_Top, como se muestra en la Figura 4.57 y el diseño de E/S de este bloque se muestra en la Figura 4.58.

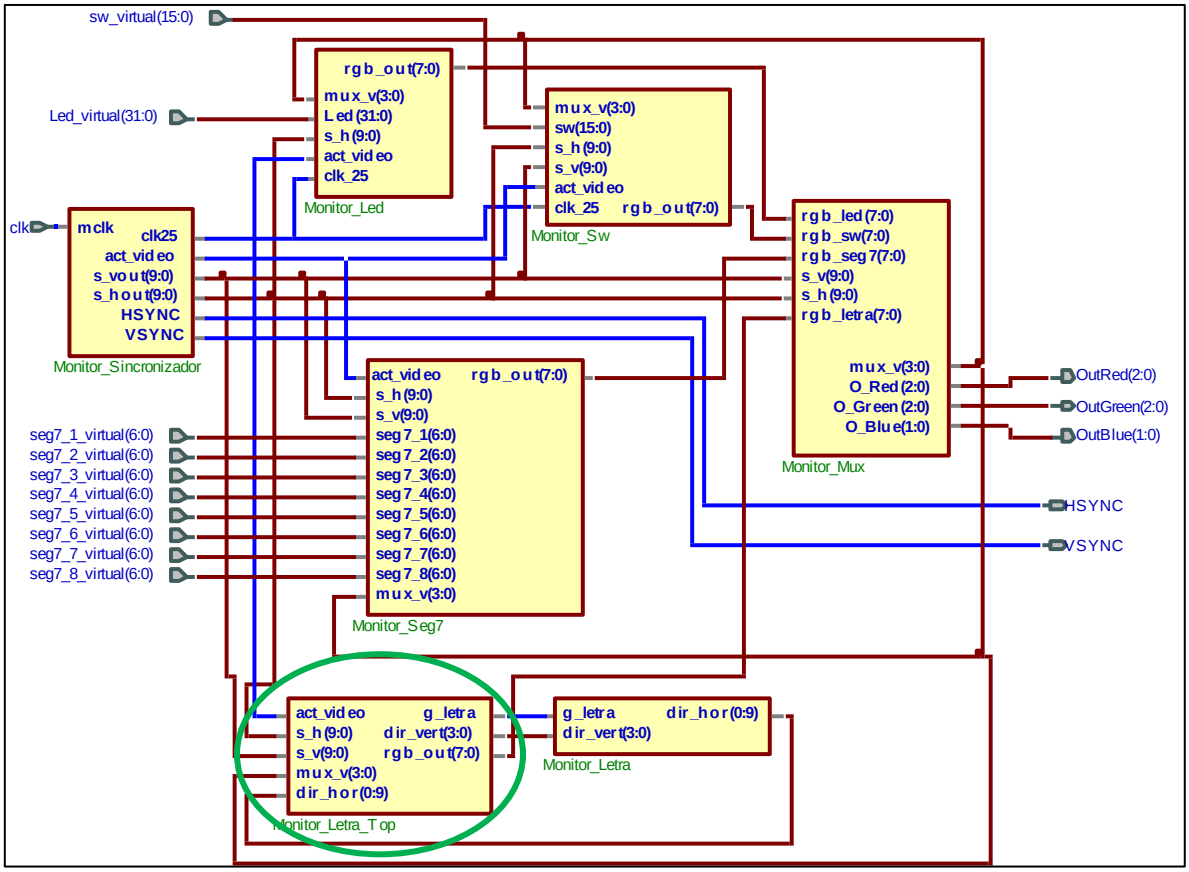


Figura 4.57 Conexión de Monitor_Letra_Top en el diseño de Monitor_Top.

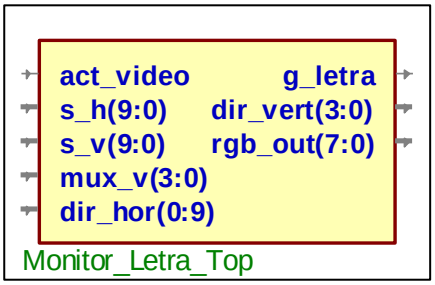


Figura 4.58 Bloque Monitor_Letra_Top.

Del bloque se detallan sus variables de entrada en la Tabla 4.40 y sus variables de salida en la Tabla 4.41. Mientras que el comportamiento de la entidad Monitor_Letra_Top se muestra en la Figura 4.59.

Nombre	Tipo	Descripción
act_video	Std_logic	Señal que indica si se encuentra en la matriz visual
s_h	Std_logic_vector (9:0)	Posición del pixel horizontal
s_v	Std_logic_vector (9:0)	Posición del píxel vertical
mux_v	Std_logic_vector (3:0)	Señal de multiplexor
dir_hor	Std_logic_vector (0:9)	Señal de posición del pixel horizontal en la matriz de pixel de la letra

Tabla 4.40 Descripción de las señales de entrada del bloque Monitor_Letra_Top.

Nombre	Tipo	Descripción
g_letra	Integer	Asignación de la letra
dir_vert	Std_logic_vector (3:0)	Señal de posición de la fila vertical de pixel en la letra
rgb_out	Std_logic_vector (9:0)	Señal de salida RGB de los Leds

Tabla 4.41 Descripción de las señales de salida del bloque Monitor_Letra_Top.

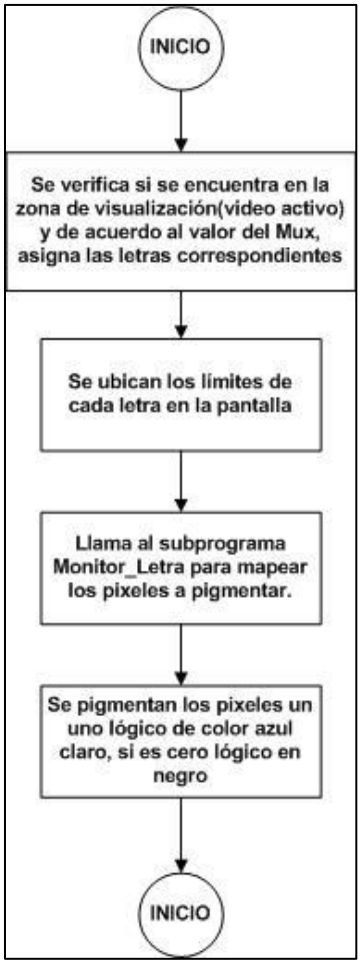


Figura 4.59 Comportamiento de la entidad Monitor_Letra_Top.

Este bloque posee en la entrada las acostumbradas señales (s_h, s_v, act_video y mux_v) para identificar en qué zona de la pantalla del monitor se encuentra actualmente, y además cuenta con la señal de entrada dir_hor proveniente del sub-programa Monitor_Letra, encargada de transmitir la cadena de 10 bits, correspondiente a la letra y la fila solicitada a través de las señales de salida g_letra y dir_vert respectivamente. Mientras que rgb_out sirve para transmitir el color a pigmentar en la pantalla.

Subprograma Monitor_Letra

Definidos los elementos virtuales más resaltantes de la interfaz gráfica, corresponde desarrollar las letras que desempeñen de títulos e identificadores de los dispositivos

virtuales. Esa es la función del subprograma Monitor_Letra en agrupación con Monitor_Letra_Top.

Primeramente se describe el subprograma Monitor_Letra, encargada de almacenar en un cuadro de 10x10 pixeles el mapa de bits de cada letra, desde el número 0 al 9 y desde la letra A hasta la Z, además de emitir la fila de bits que se le solicite para pigmentar la letra deseada en la pantalla. Como se muestra en la Figura 4.60 en forma de ejemplo, el mapa de bits del número cero. Este subprograma se encuentra contenido en el bloque Monitor_Top, como se muestra en la Figura 4.61 y cuyo diseño de E/S en la Figura 4.62.

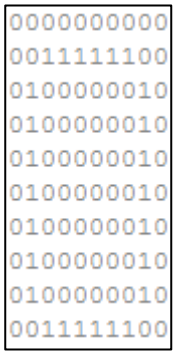


Figura 4.60 Mapa de bits del número 0 en el sub-programa Monitor_Letra.

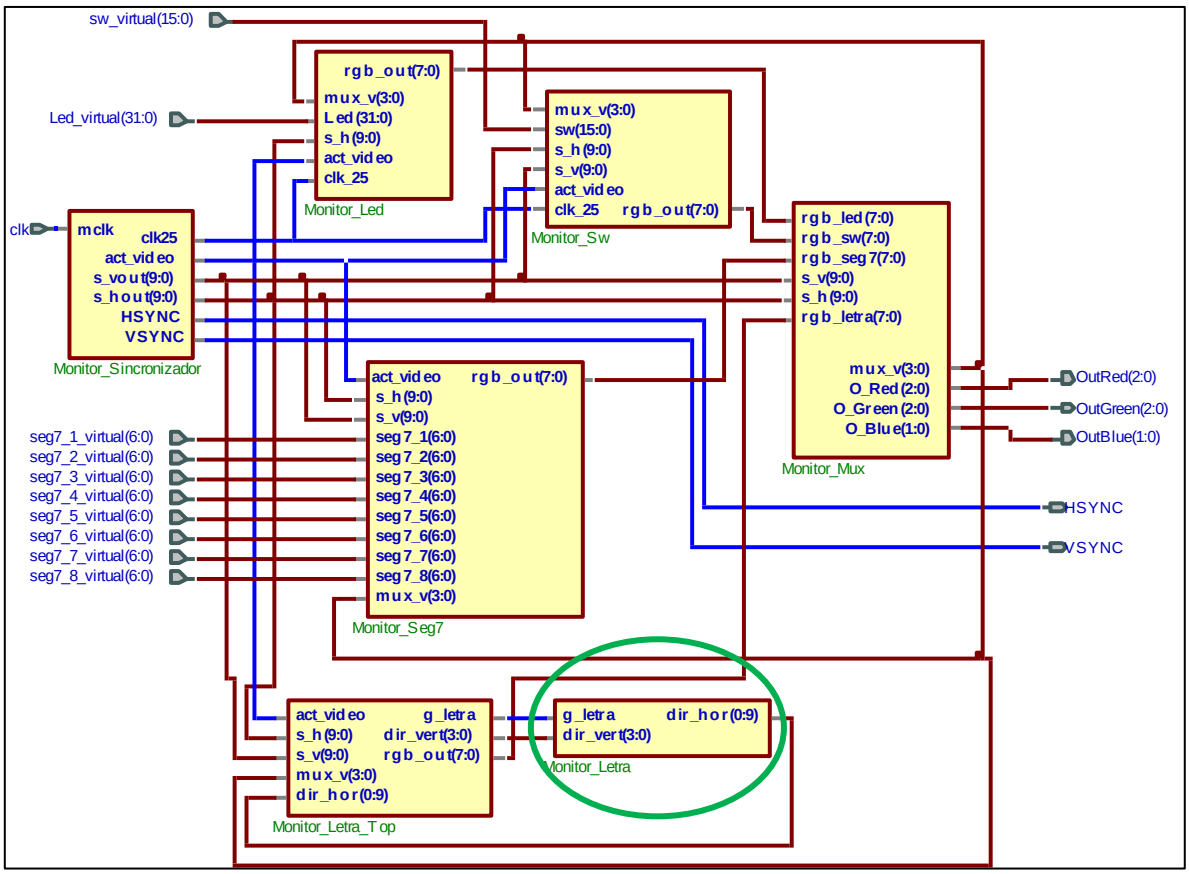


Figura 4.61 Conexión de Monitor_Letra en el diseño de Monitor_Top.

Figura 4.62 Bloque Monitor_Letra.

Del bloque se detallan sus variables de entrada en la Tabla 4.42 y sus variables de salida en la Tabla 4.43. Así como el comportamiento de la entidad Monitor_Letra se muestra en la Figura 4.63.

Nombre	Tipo	Descripción
g_letra	Integer	Asignación de letra.
dir_vert	Std_logic_vector (3:0)	Señal de posición de la fila vertical de pixel en la letra.

Tabla 4.42 Descripción de las señales de entrada del bloque Monitor_Letra.

Nombre	Tipo	Descripción
dir_hor	Std_logic_vector (0:9)	Señal de posición del pixel horizontal en la matriz de pixel de la letra.

Tabla 4.43 Descripción de las señales de salida del bloque Monitor_Letra.

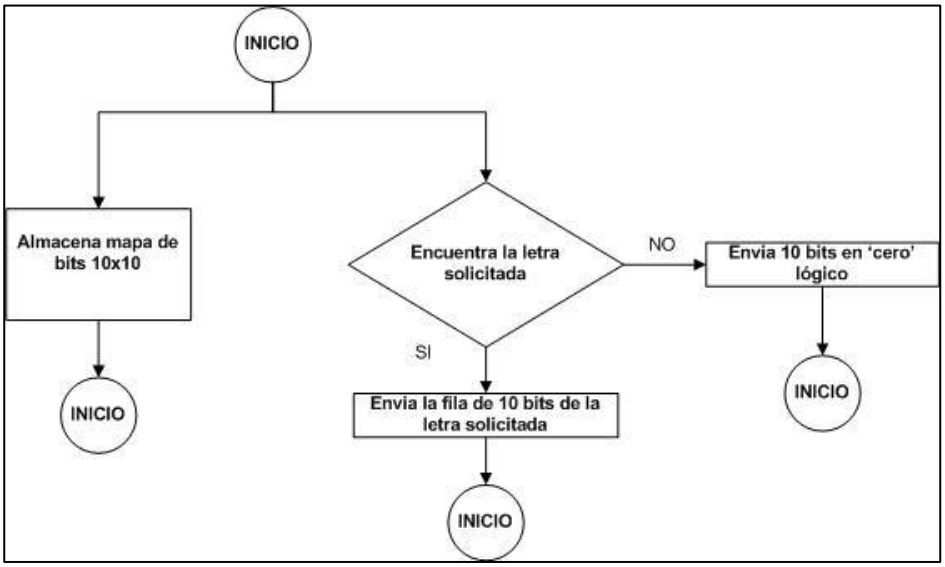


Figura 4.63 Comportamiento de la entidad Monitor_Letra.

El bloque posee en la entrada la señal g_letra, proveniente del sub-programa Monitor_Letra_Top cuya ocupación es identificar la letra a ensamblar en la pantalla. Una vez reconocida la letra, se emplea la señal de entrada dir_vert para reconocer en qué línea vertical del mapa de bits se encuentra. Y por último dispone en la salida con la señal dir_hor que envía la cadena de 10 bits, correspondiente a la fila seleccionada de la matriz de píxeles de la letra indicada, a fin de pigmentar con esa plantilla la pantalla a medida que se realiza el barrido.

A través de todos los bloques explicados en esta sección y con la incorporación del bloque Monitor_Sincronizador se obtienen los subprogramas internos que componen el bloque Monitor_Top, diseñado para la comunicación de la tarjeta Basys2 con el monitor con puerto VGA y realizar la interfaz gráfica adaptada para el usuario. El diseño definitivo de la interfaz se muestra en la Figura 4.64.

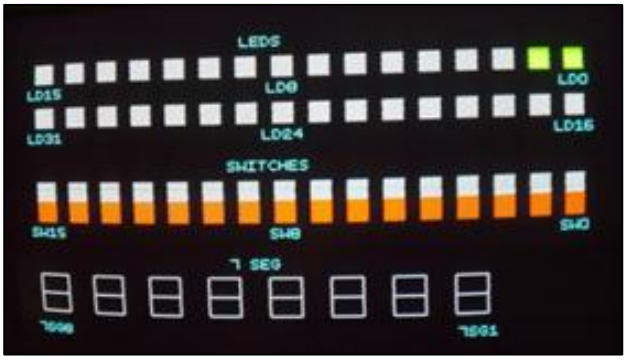


Figura 4.64 Diseño definitivo de la interfaz.

Ahora con respecto al teclado, la señal del teclado que se maneja hasta ahora es la del código de scan, procedente del subprograma Monitor_Lectura, por lo que se diseñaron dos subprogramas adicionales para adaptar la señal del teclado a las necesidades del usuario.

CAPITULO V: CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- El diseño de la interfaz gráfica, así como la efectiva comunicación entre los puertos VGA y PS/2 de la tarjeta BASYS2, cumple con los objetivos planteados en este Trabajo Especial de Grado.

- Los diseños realizados se componen de 14 subprogramas, constituidos en forma modular, lo que permite seleccionar los subprogramas a ser incorporados para un manejo de los puertos VGA y PS/2 de la tarjeta Basys2 adaptado a los requerimientos del estudiante y del Laboratorio de Lógica Digital.

- A pesar de que los subprogramas y los ejemplos se realizaron con el ambiente Active HDL, la lógica de programación sirve para aplicarlo a otros ambientes así como el Xilinx ISE, entre otros.

- Se dominan los protocolos de comunicación tanto del teclado como del monitor con la tarjeta Basys2. Para el caso del monitor se efectúa el barrido de la señal en la pantalla, identificando el ancho a considerarse para realizar el escaneo de la pantalla sesenta veces por segundo logrando así su visualización. En el caso del teclado se organizan las señales recibidas y se transmiten solo desde el segundo bit de la trama hasta el noveno, los cuales representan el dato de la tecla accionada o no.

- El estudio de los protocolos de comunicación del puerto VGA y PS/2 permiten el uso de más variables de entradas y salidas físicas para distintas aplicaciones.

- Conociendo las limitaciones de la tarjeta Basys2 para este proyecto, se incorpora un oscilador externo para poder apreciar la señal con estabilidad y mayor calidad en el monitor, mediante el uso del puerto VGA.

- Mediante el lenguaje de descripción de hardware (VHDL) se estableció la lógica del sistema a realizar y las señales involucradas de cada subprograma.

5.2 RECOMENDACIONES

- Implementar el uso de bibliotecas del software empleado ACTIVE-HDL del diseño realizado para mayor practicidad a la hora de importar los diseños.

- Se sugiere que los códigos desarrollados estén al resguardo y disposición únicamente de la Cátedra de Lógica Digital de la Universidad de Carabobo, para su posterior modificación y/o mejoría.

- Notificar a los usuarios para que no modifiquen el diseño si no tienen la debida autorización de la Catedra de Lógica Digital.

- Extrapolar esta aplicación para su uso con otras tarjetas lógicos programables, tal como la NEXYS2, en el desarrollo de otras áreas de investigación.

- Realizar los ejemplos ilustrados tanto en el desarrollo de este Trabajo Especial de Grado como en el manual de usuario, para el rápido entendimiento de las variables de entradas y salidas que componen cada subprograma además de minimizar los inconvenientes al momento de compilar, sintetizar e implementar.

- Examinar el manual de usuario antes de utilizar algunos de los subprogramas, en caso que se presenten dudas o inconvenientes al usarlo.

- Para el desarrollo de nuevas aplicaciones mediante el uso de la tarjeta lógico programable Basys2 es necesario identificar y verificar los archivos con extensión .ucf antes de su implementación, ya que pueden generar errores al momento de la compilación.

- Con el fin de optimizar el uso de la tarjeta Basys2, se recomienda evaluar la posibilidad de usar otros softwares, tales como Xilinx o Altera.

- Considerar la incorporación de una memoria PROM en la tarjeta lógico programable Basys2 para potenciar la capacidad de almacenamiento y respuesta de la tarjeta en el desarrollo de nuevos diseños.

- Evaluar como una posible fuente adicional de ingreso económico para la Universidad de Carabobo, el desarrollo de aplicaciones de entretenimiento didáctico, así como la creación de aplicaciones o juegos mediante el uso del puerto PS/2 y/o VGA de la tarjeta Basys2.

- Proponer como tema para otro Trabajo Especial de Grado, el desarrollo de una interfaz gráfica en donde se haga uso del puerto PS/2 conectado a un ratón.

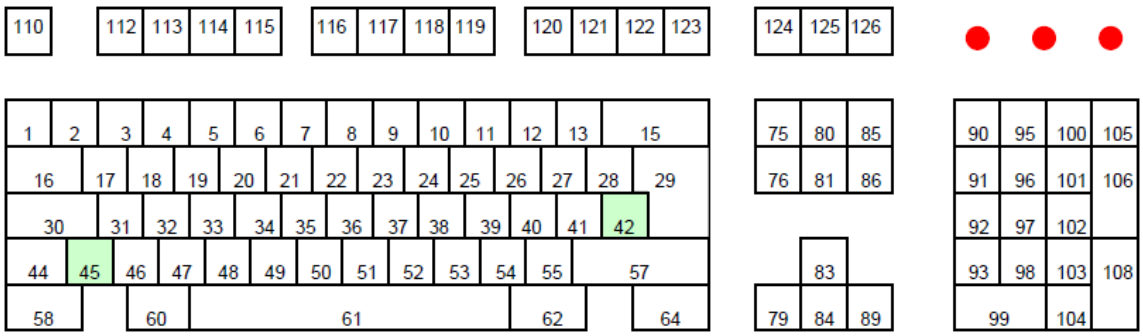
APÉNDICES

APENDICE A

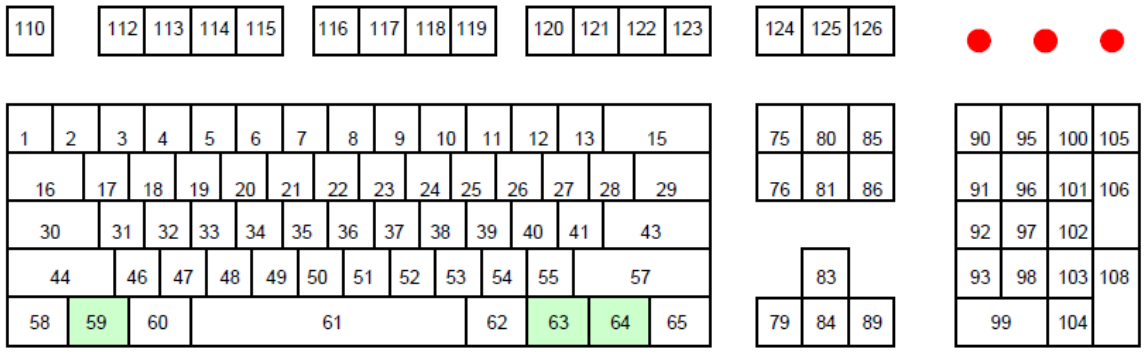
Esquemas de distribución de las teclas de un teclado con puerto PS/2

Distribución de las teclas

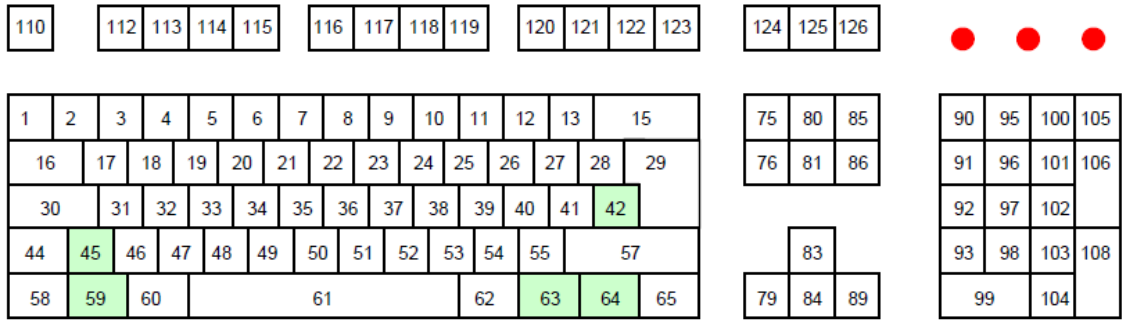
102 Teclas (Europa)



104 Teclas (US)



105 Teclas (Europa)



APENDICE B

Manual de Usuario para el uso de puerto VGA y PS/2 conectado a un monitor y teclado

Este manual se explica partiendo de la creación de un proyecto nuevo, sin embargo esto no impide el uso de los subprogramas en proyectos ya creados con cualquier otro diseño.

Requisitos previos:

- Guía rápida de Síntesis e Implementación (con VHDL). Prof. Demetrio Rey Lago [14]
- Los módulos que se usarán como componentes en su diseño serán:

❖ Para el uso del puerto VGA conectado a un monitor:
- Monitor_Top.vhd

Los módulos internos que componen al módulo Monitor_Top.vhd son:
- Monitor_Sincronizador.vhd
- Monitor_Led.vhd
- Monitor_Sw.vhd
- Monitor_Seg7.vhd
- Monitor_Letra.vhd
- Monitor_Letra_Top.vhd
- Monitor_Mux.vhd

❖ Para el uso del puerto PS/2 conectado a un teclado:
- Teclado_Top.vhd

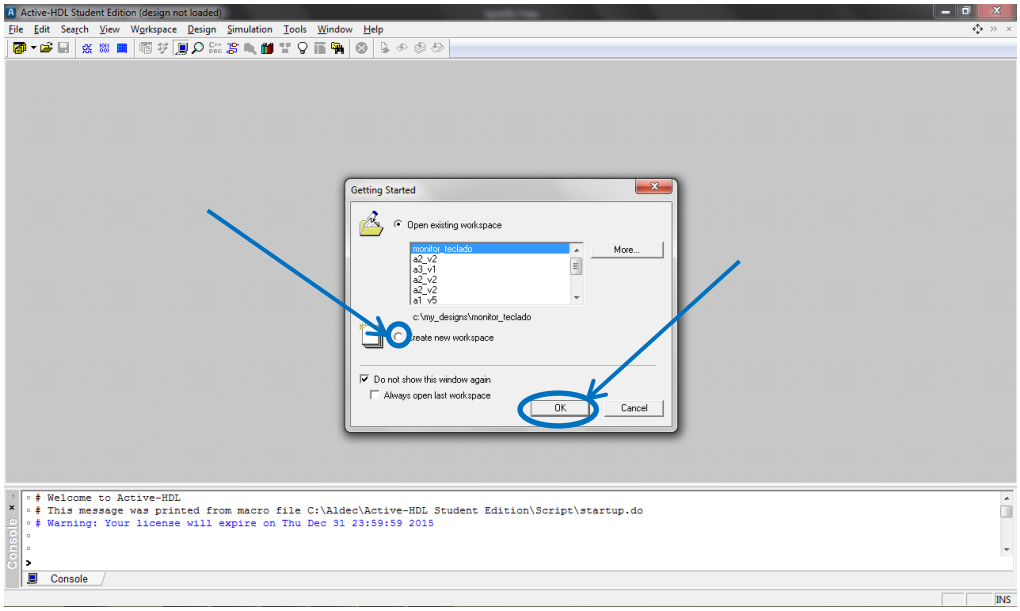
Los módulos internos que componen al módulo Teclado_Top.vhd son:
- Teclado_Lectura.vhd
- Teclado_Tecla_Valor.vhd

❖ Para los ejemplos que se describen mas adelante requieren módulos adaptadores y conversores:
- Adap_2_Num_8Bits.vhd
- Adap_Contador.vhd
- Conv_7Seg.vhd

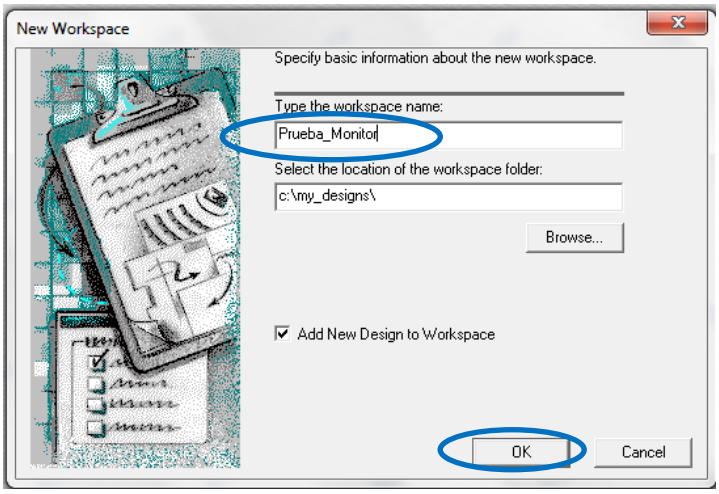
- Si se tiene oscilador externo de 50MHz, debe elegir entre uno de los siguientes archivos de restricción:
  - ✓ Archivo *Basys2_Tec_OsciladorExt.ucf*
  - ✓ Archivo *Basys2_Mon_OsciladorExt.ucf*
  - ✓ Archivo *Basys2_Mon_Tec_OsciladorExt.ucf*

- Si no se tiene oscilador externo de 50MHz, debe elegir entre uno de los siguientes archivos de restricción:
  - ✓ Archivo *Basys2_Tec_OsciladorInt.ucf*
  - ✓ Archivo *Basys2_Mon_OsciladorInt.ucf*
  - ✓ Archivo *Basys2_Mon_Tec_OsciladorInt.ucf*

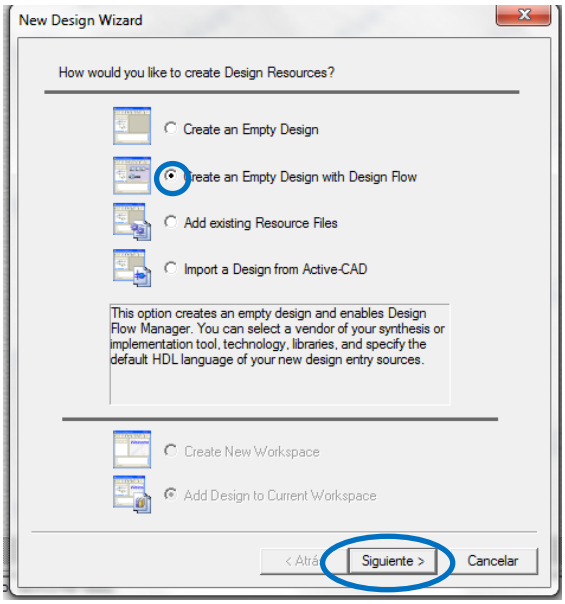
1. Crear un nuevo proyecto.



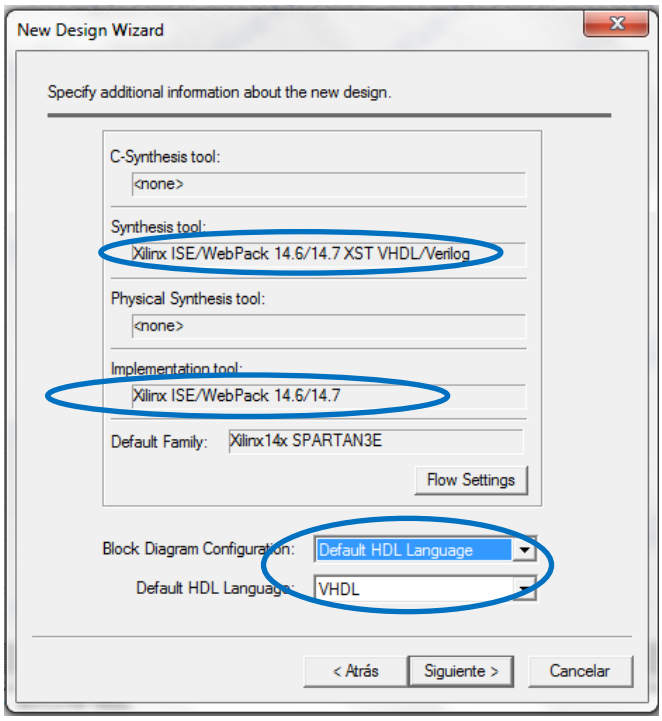
Al abrir el programa Active-HDL, se presentará la siguiente pantalla, para crear un nuevo proyecto, marcan “Create new workspace” luego presionar “OK”.



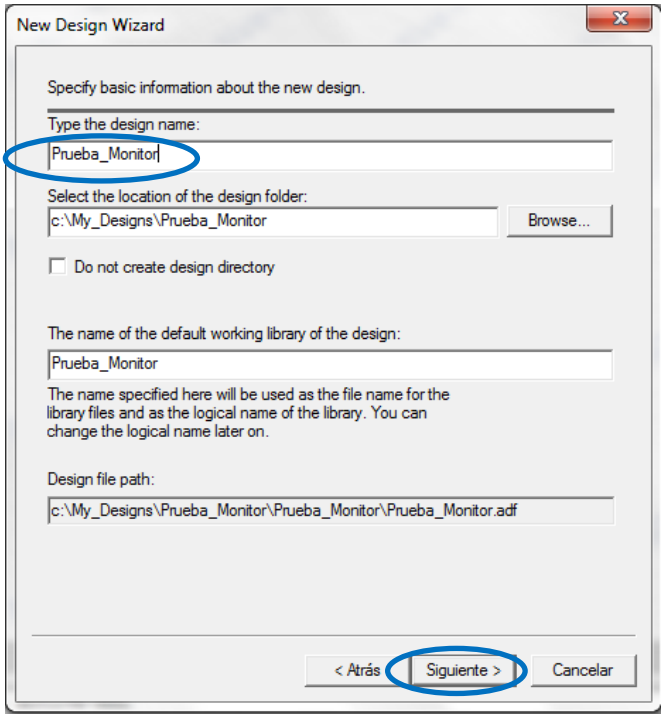
Se designa un nombre al proyecto, cualquier nombre que se prefiera, en este caso se designará “Prueba_Monitor”, se presiona “OK”.



Marcar “Create an Empty Design with Design Flow” y se presiona “Siguiente”.



Revisar que las herramientas, configuración del diagrama del bloque y el lenguaje a utilizar sean las mostradas.



Se designa un nombre de diseño al proyecto, puede ser el mismo que el nombre del Proyecto y se presiona “Siguiente”.

Para mayor información consulte la Guia Active-HDL 7.2 Student Edition [14].

2. Como agregar un oscilador externo

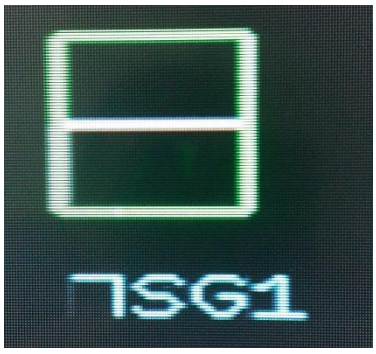
¿Por qué agregar un oscilador externo si ya la tarjeta BASYS2 tiene un oscilador interno?.

Para aplicaciones con uso del puerto VGA de la FPGA Basys2 usando su oscilador interno, la señal en el monitor resulta inestable, debido a que este oscilador no presenta la misma calidad en su señal que un oscilador externo de cristal, así como se indica en el manual de usuario de la tarjeta BASYS2. En las siguientes imágenes se muestra la diferencia entre usar un oscilador interno y un oscilador externo para el ejemplo 1 de este manual de usuario, en donde se puede ver que al usar un oscilador externo de cristal, la imagen en el monitor es más estable:

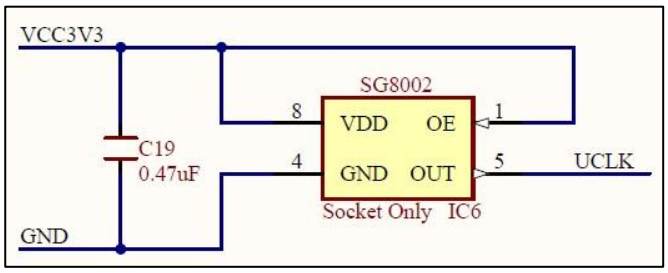
Resultado en la pantalla del monitor

Con oscilador interno:

Con oscilador externo:

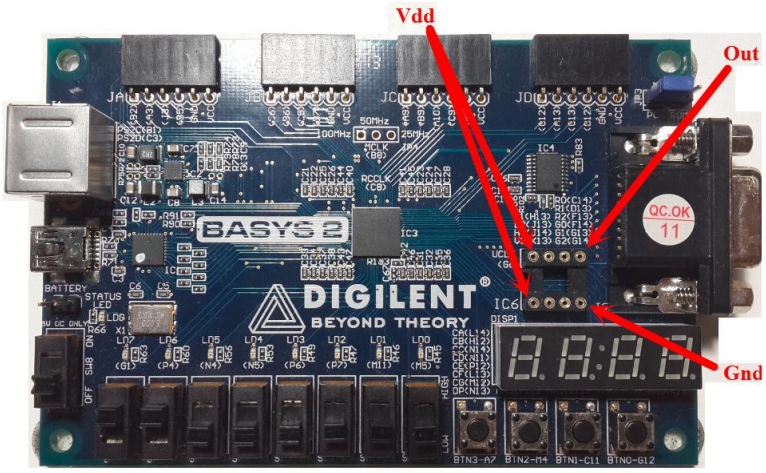


Antes de agregar el oscilador externo, primero se debe conocer los pines en donde va conectado el oscilador externo en la tarjeta BASYS2 y tener adicionalmente el datasheet del oscilador externo a agregar.



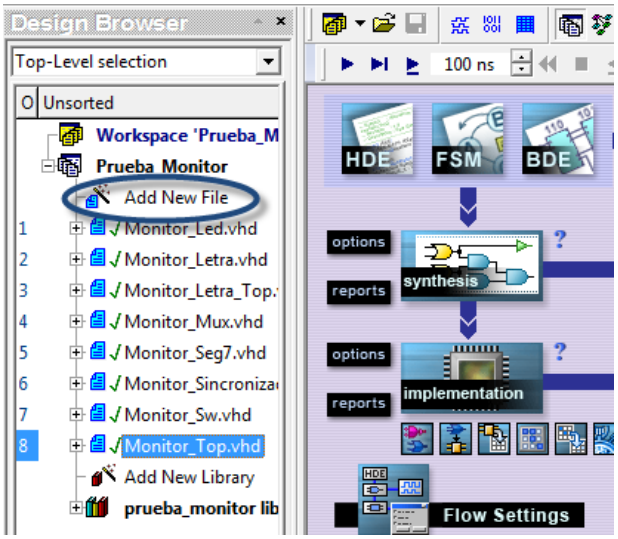
Conexión interna del Socket IC6 de la tarjeta Basys2

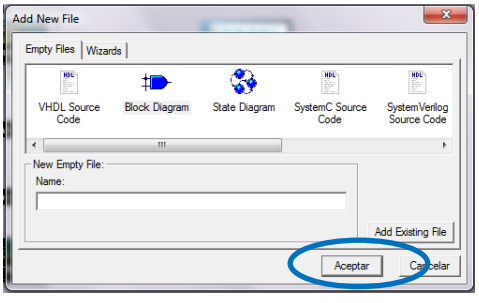
Fuente: Manual Esquemático de conexiones de la tarjeta Basys2



Descripción de los pines para agregar un oscilador externo de la tarjeta Basys2

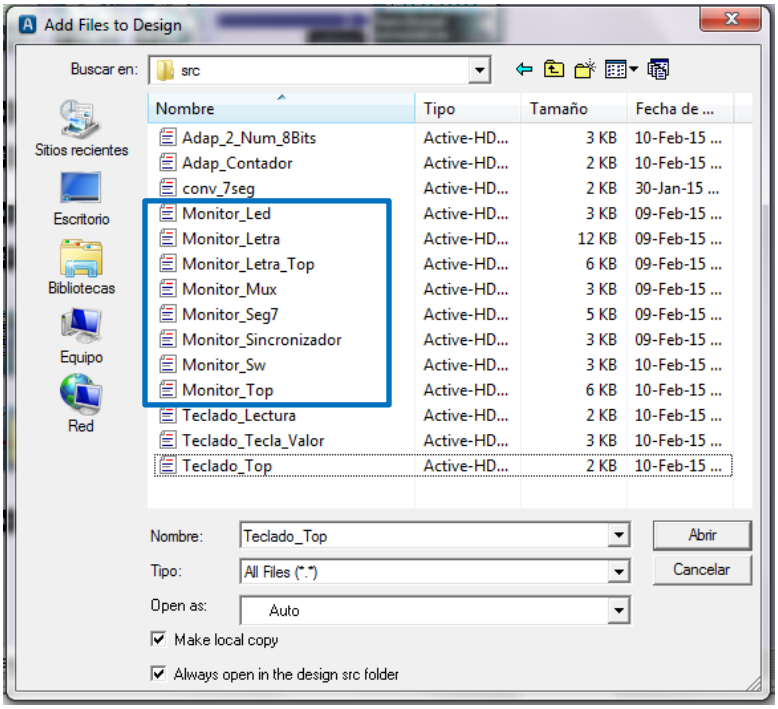
3. Agregar módulos para uso de un monitor con el puerto VGA





En la barra Lateral izquierda “Design Browser”, seleccionar “Add New File”. En la siguiente ventana, presionar “Add Existing File” y buscar los archivos a agregar.

Se deben que agregar los archivos nombrados a continuación:



-Monitor_Top

-Monitor_Sincronizador

-Monitor_Led

-Monitor_Sw

-Monitor_Seg7

-Monitor_Letra

-Monitor_Letra_Top

-Monitor_Mux

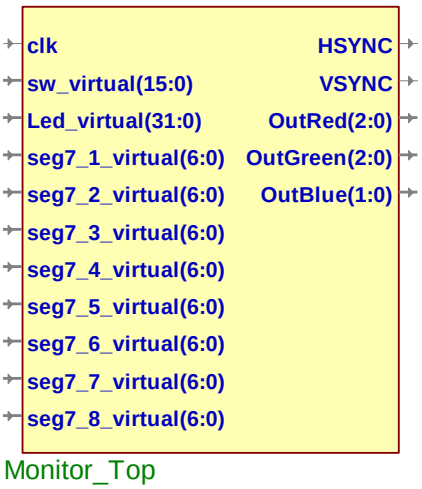
Módulo Monitor_Top

Función

El módulo *Monitor_Top* se encarga de realizar la comunicación de la tarjeta Basys2 con el puerto VGA, además crear una interfaz gráfica en donde se muestran Leds virtuales, Switches virtuales y Display 7 segmentos virtuales. Estos Leds, Switches y Display 7 segmentos virtuales, son salidas extras que se pueden usar para cualquier aplicación.

Los módulos internos que componen este módulo son: Monitor_Sincronizador, Monitor_Led, Monitor_Sw, Monitor_Seg7, Monitor_Letra, Monitor_Letra_Top, Monitor_Mux. Se tienen que agregar todos los módulos internos para su correcto funcionamiento.

Entidad



Variables de entrada

Nombre	Tipo	Descripción
Clk	Std_logic	Reloj a 50MHz
Sw_Virtual(15:0)	Std_Logic_Vector (15 downto 0)	Switches virtuales
Led_virtual(31:0)	Std_Logic_Vector (31 downto 0)	Leds virtuales
Seg7_1_virtual(6:0)	Std_Logic_Vector (6 downto 0)	7 segmento virtual 1
Seg7_2_virtual(6:0)	Std_Logic_Vector (6 downto 0)	7 segmento virtual 2
Seg7_3_virtual(6:0)	Std_Logic_Vector (6 downto 0)	7 segmento virtual 3
Seg7_4_virtual(6:0)	Std_Logic_Vector (6 downto 0)	7 segmento virtual 4
Seg7_5_virtual(6:0)	Std_Logic_Vector (6 downto 0)	7 segmento virtual 5
Seg7_6_virtual(6:0)	Std_Logic_Vector (6 downto 0)	7 segmento virtual 6
Seg7_7_virtual(6:0)	Std_Logic_Vector (6 downto 0)	7 segmento virtual 7
Seg7_8_virtual(6:0)	Std_Logic_Vector (6 downto 0)	7 segmento virtual 8

Variables de salida

Nombre	Tipo	Descripción
HSYNC	Std_logic	Señal de Sincronización Horizontal

		para el puerto VGA
VSYNC	Std_logic	Señal de Sincronización Vertical para el puerto VGA
OutRed(2:0)	Std_logic_Vector (2 downto 0)	Bits para la pigmentación de los pixeles en Rojo
OutGreen(2:0)	Std_logic_Vector (2 downto 0)	Bits para la pigmentación de los pixeles en Verde
OutBlue(1:0)	Std_logic_Vector (1 downto 0)	Bits para la pigmentación de los pixeles en Azul

Ejemplo 1. Control_Led

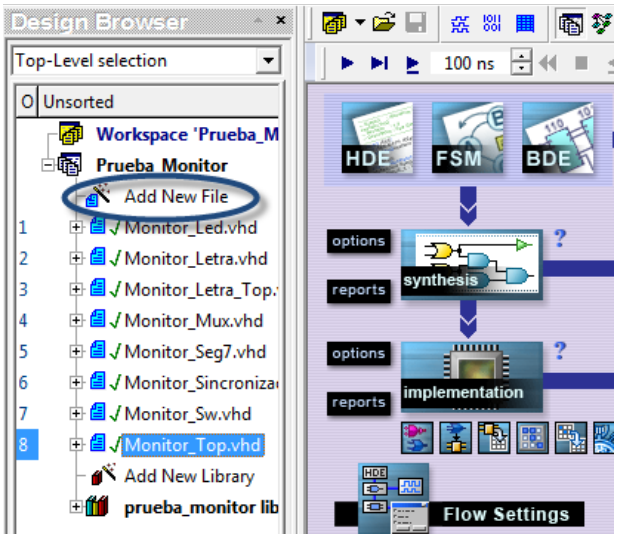
Se quiere controlar un Led virtual en el monitor, mediante un pulsador de la tarjeta Basys2.

Solución

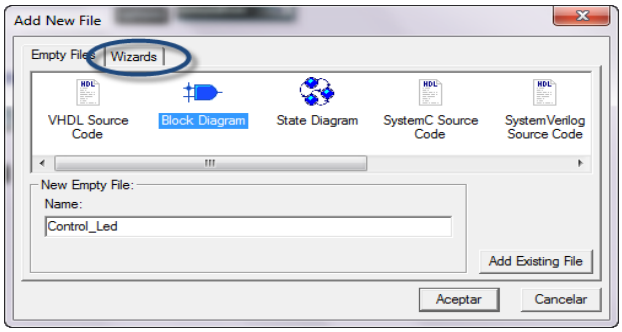
En este ejemplo, se realizará partiendo de haber creado un proyecto y diseño nuevo, y al mismo tiempo agregado los subprogramas para el uso del puerto VGA conectado a un monitor.

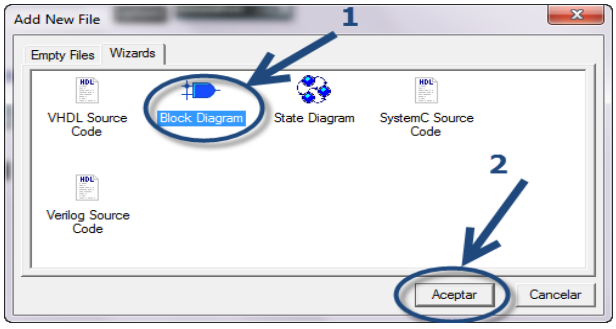
Se necesitará crear dos subprogramas, uno (Subprograma Control_Led) en donde llevará a cabo el control del led virtual mediante un pulsador, y otro (Subprograma Top_Control_Led) en donde se realizará el conexiondado de los subprogramas a los pines de la tarjeta. Ambos subprogramas se realizaran por medio de diagrama de bloques.

Subprograma Control_Led

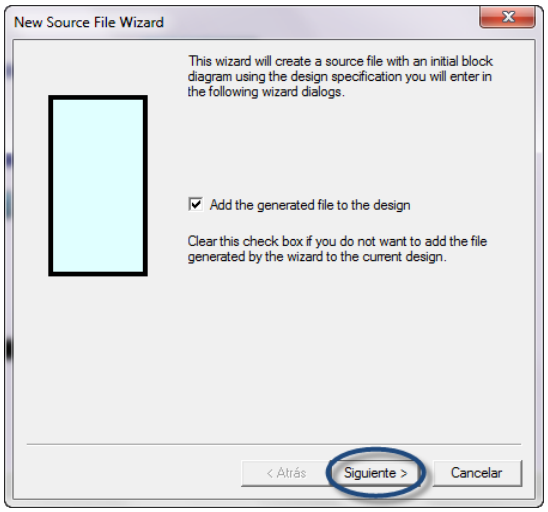


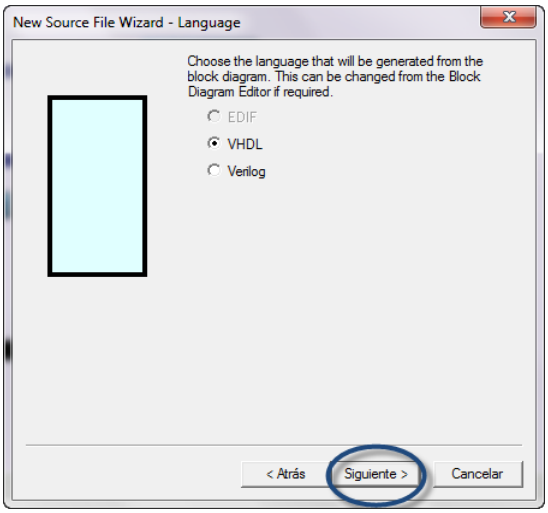
En la barra lateral izquierda, hacer doble click en “Add New File”, para crear un nuevo archivo.



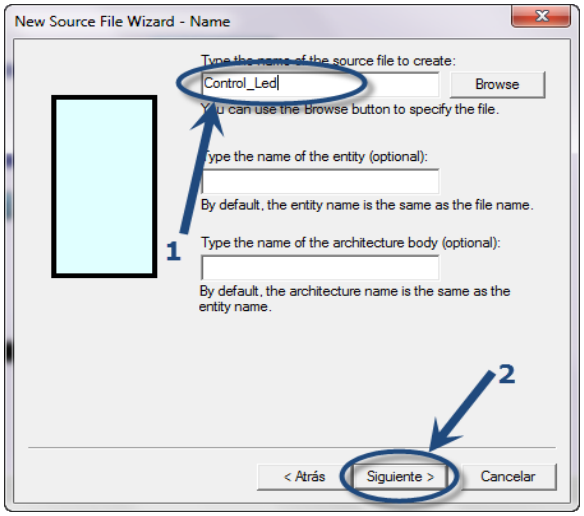


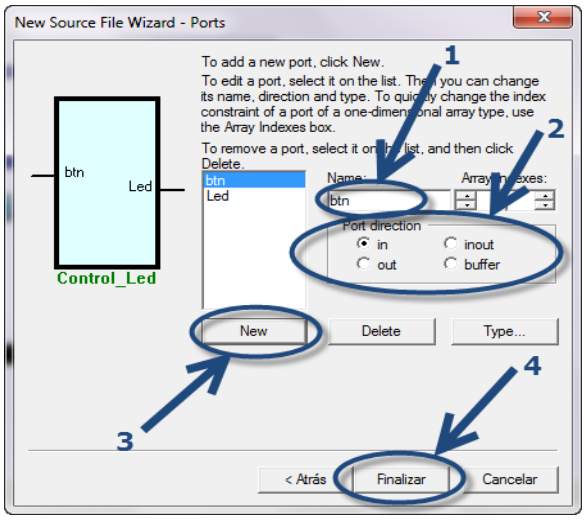
Se clickea “Wizards”, selecciona “Block Diagram” y luego clickear “Aceptar”.



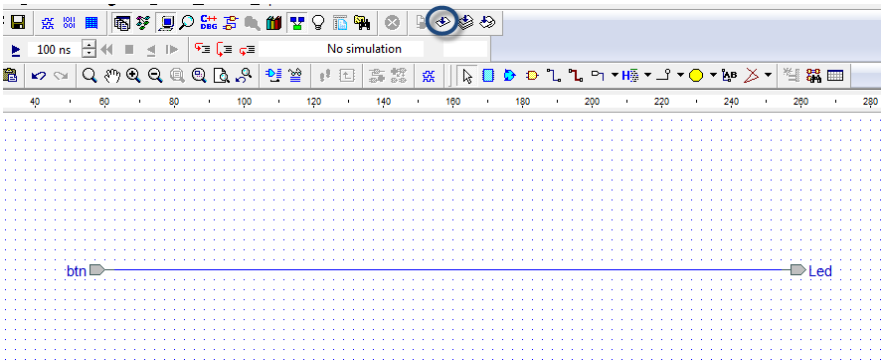


Se clickea “Siguiente” en ambas pantallas.



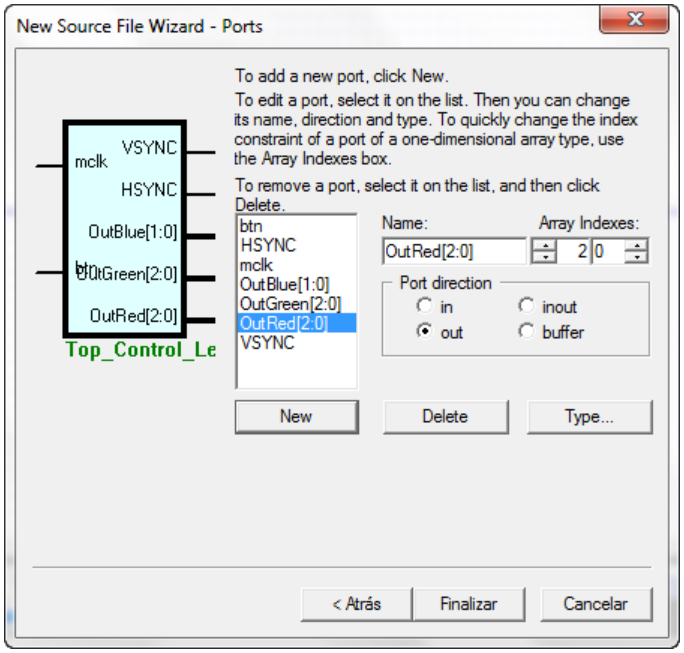


Se coloca el nombre que desee, en nuestro caso usamos “Control_Led”, luego se clickea “Siguiente”. En la proxima ventana añadiremos las variables de entrada y salida que tiene el programa, seleccionaremos de entrada a “btn” y de salida a “Led”.

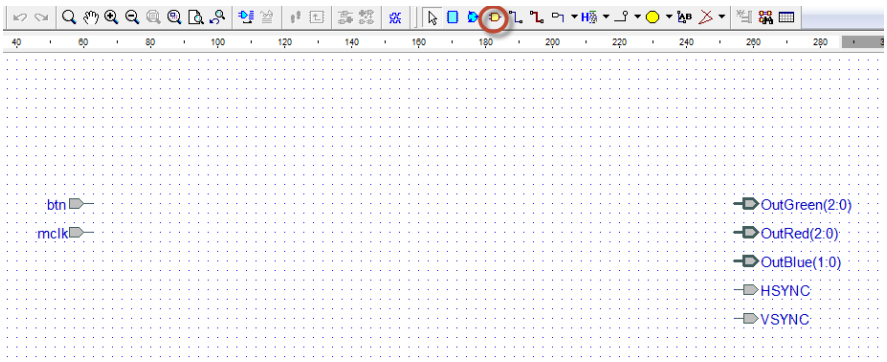


Lo que se quiere es que al presionar el pulsador se active un led virtual en el monitor, entonces lo que se necesita es conectar el pulsador directamente al led, como se muestra en la imagen.

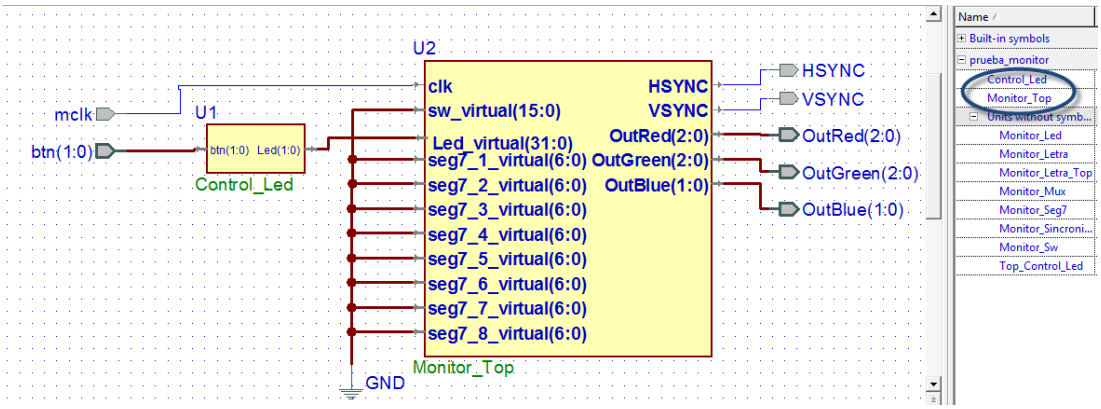
Subprograma Top_Control_Led



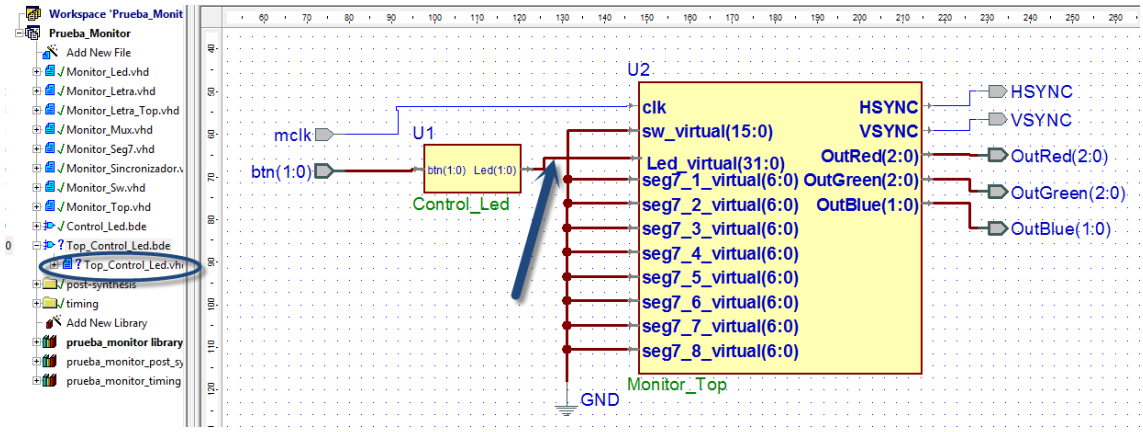
Ahora se procede a crear otro programa, el cual será la conexión de todos los bloques en el diseño, generalemente lo referimos como “Top”. Se deben agregar las variables de entradas y salidas del bloque Top_Control_Led.



En la imagen se visualiza como se mostraran en pantalla los elementos seleccionados, solo los pines de entradas y salidas. Se da click en  para poder acceder a los bloques de los programas ya creados.



Aparecerá “Symbols Toolbox”, una barra lateral a la derecha, en donde se seleccionará el programa creado anteriormente “Control_Led” y el programa “Monitor_Top” que es para el uso de puerto VGA conectado a un monitor. Realizamos las conexiones respectivamente.



Luego de realizado el conexionado del circuito, el cable bus señalado, si se colocó directamente, por defecto asignará el Led(1) y Led (0) al puerto Led_virtual(31) y Led_virtual(30), porque Led_Virtual se declaró como *(31 downto 0)* es decir, las asignaciones se haran a partir del bit mas significativo. Si se quiere modificar a que Led_virtual se quieren mostrar los Leds específicamente, entonces se da doble click al Top_Control_Led.vhd.

```
77   signal BUS460 : STD_LOGIC_VECTOR (15 downto 0);
78   signal BUS469 : STD_LOGIC_VECTOR (31 downto 0);
79
80   begin
81
82   ----  Component instantiations  ----
83
84   U1 : Control_Led
85     port map(
86          Led(0) => BUS469(0),
87          Led(1) => BUS469(1),
88          btn => btn
89     );
90
91   U2 : Monitor_Top
92     port map(
93          HSYNC => HSYNC,
94          Led_virtual => BUS469,
95          OutBlue => OutBlue,
96          OutGreen => OutGreen,
97          OutRed => OutRed,
98          seg7_1_virtual(0) => BUS460(9),
99          seg7_1_virtual(1) => BUS460(10),
100         seg7_1_virtual(2) => BUS460(11),
101         seg7_1_virtual(3) => BUS460(12),
102         seg7_1_virtual(4) => BUS460(13),
```

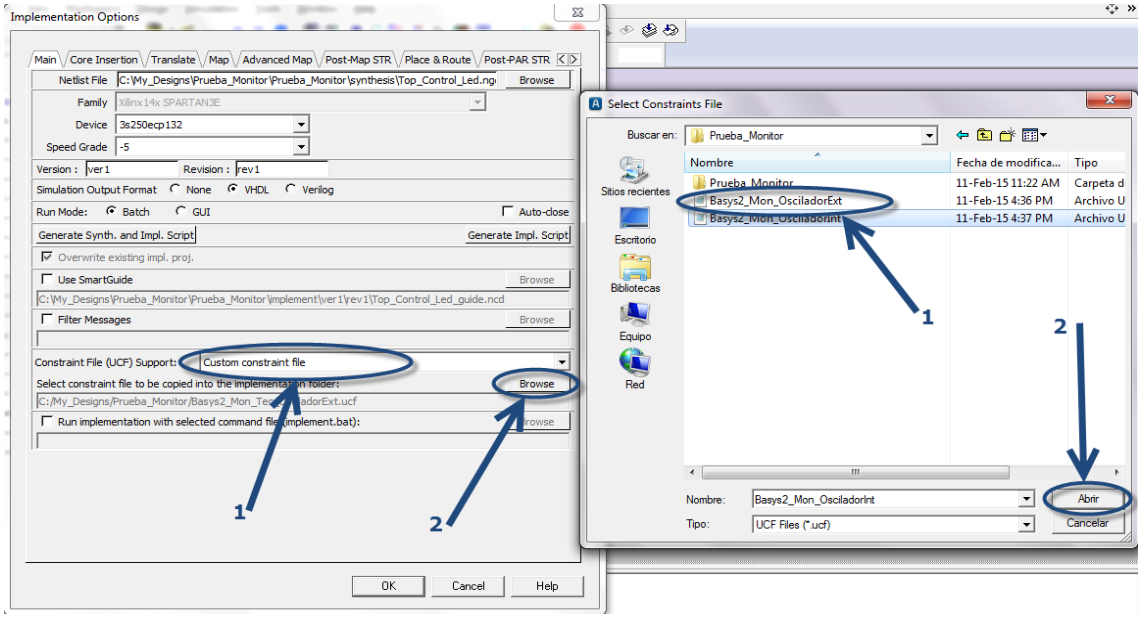
Se cambia la asignación de los Leds a los bits 0 y 1, los menos significativos o por el bit del Ledvirtual que desee mostrar.

Se compilan todos los archivos, se aseguran que no haya ningun error para luego continuar a la síntesis e implementación, en la *Guía rápida de Síntesis e Implementación*

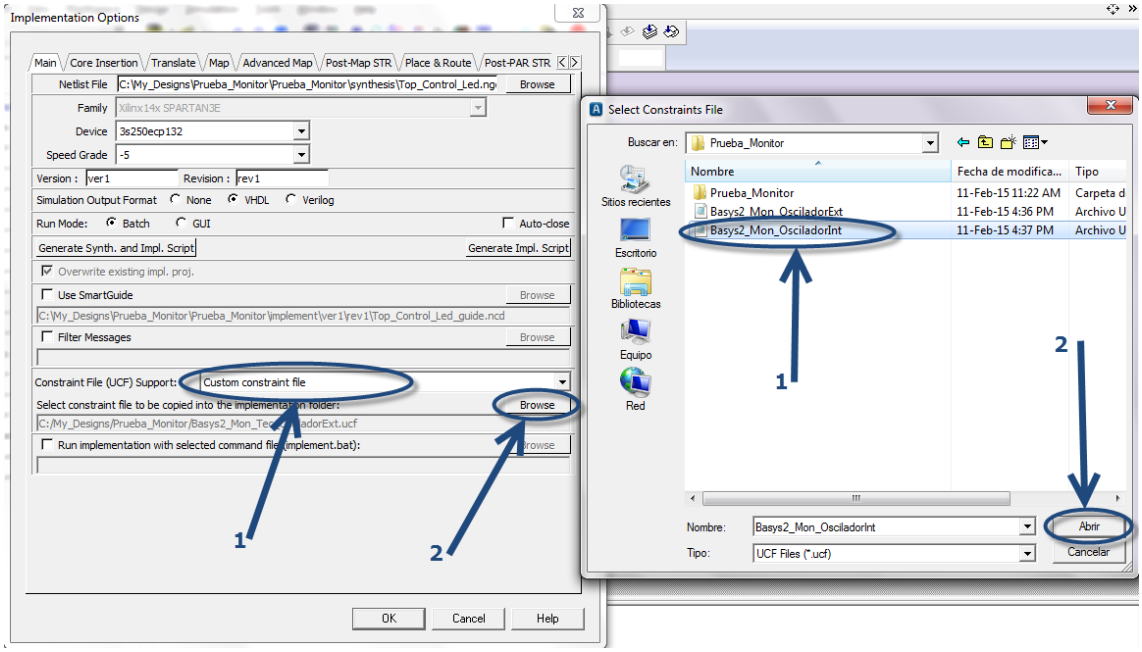
(con VHDL), del Profesor Demetrio Rey Lago [14], explica paso a paso como realizar la síntesis e implementación.

Al momento de configurar la implementación, el archivo *.ucf* a usar será:

-*Basys2_Mon_OsciladorExt.ucf*. Si le agregó un oscilador externo a la tarjeta Basys2.

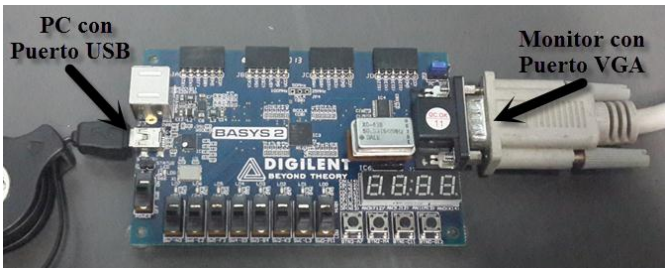


- *Basys2_Mon_OsciladorInt.ucf*. Si no le agregó un oscilador externo a la tarjeta Basys2.



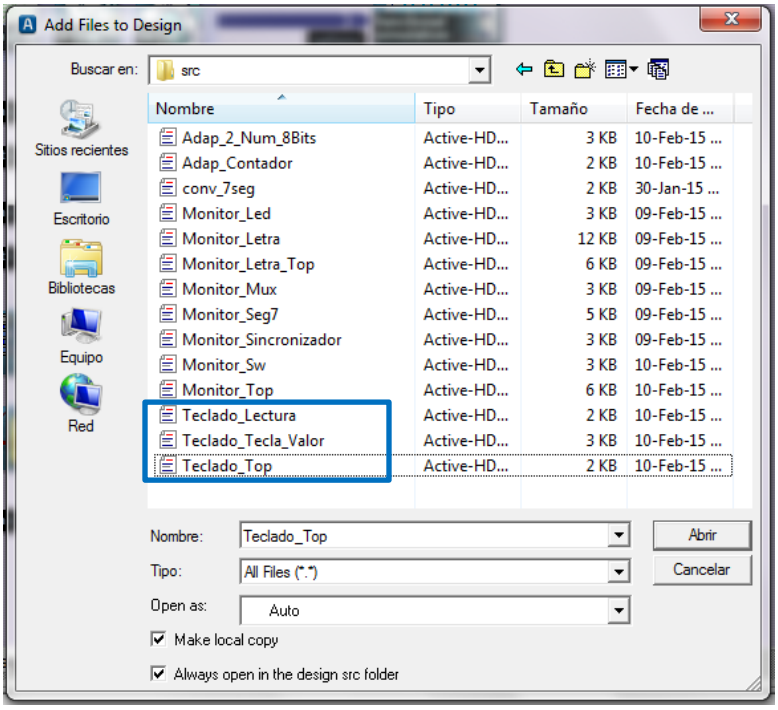
Después de realizar la implementación sin ningún error, procedemos a programar la tarjeta, transfiriendole el archivo de conexiones generados (extensión *.bit*), en la *Guía rápida de Síntesis e Implementación (con VHDL)*, del Profesor Demetrio Rey Lago [14], explica paso a paso de la programación del dispositivo.

Conexión de la tarjeta Basys2



4. Agregar módulos para uso del teclado con el puerto PS/2.

Se deben agregar los siguientes archivos para el uso del teclado con el puerto PS/2:



-Teclado_Top

-Teclado_Lectura

-Teclado_Tecla_Valor

Módulo Teclado_Top.

Función.

El módulo *Teclado_Top* se encarga de realizar la comunicación de la tarjeta Basys2 con el puerto PS/2 conectado a un teclado, posee una salida(Xkey) en donde se envía el código scan de la tecla presionada y otra salida (T_Valor) en donde se envía un hexadecimal donde el valor de este hexadecimal depende de la ultima tecla que se haya presionado, este valor esta reflejado en la siguiente tabla:

Tecla	Valor en binario	Tecla	Valor en binario	Tecla	Valor en binario
0	000000	G	010000	Y	100010
1	000001	H	010001	Z	100011
2	000010	I	010010	F1	100100
3	000011	K	010100	F2	100101

4	000100	L	010101	F3	100110
5	000101	M	010110	F4	100111
6	000110	N	010111	F5	101000
7	000111	O	011000	F6	101001
8	001000	P	011001	F7	101010
9	001001	Q	011010	F8	101011
A	001010	R	011011	F9	101100
B	001011	S	011100	F10	101101
C	001100	T	011101	F11	101110
D	001101	U	011110	F12	101111
E	001110	V	011111	ENTER	110000
F	001111	W	100000		
J	010011	X	100001		

Los módulos internos que componen este módulo son: Teclado_Lectura y Teclado_Tecla_Valor. Se tienen que agregar todos los módulos internos para su correcto funcioanmiento.

Entidad

Variables de entrada.

Nombre	Tipo	Descripción
PS2C	Std_Logic	Señal del reloj del teclado
PS2D	Std_Logic	Señal de la data del teclado
Clk	Std_Logic	Reloj a 50MHz
Clear	Std_Logic	Variable para reiniciar el sistema

Variables de salida.

Nombre	Tipo	Descripción
T_Valor(5:0)	Std_logic_Vector (5 downto 0)	Valor asignado del código scan
X_Key(15:0)	Std_logic_Vector (15 downto 0)	Código de scan

5. Usando ambos subprogramas como del monitor y el teclado con el puerto VGA y PS/2 respectivamente.

Se deben agregar los subprogramas dichos en el numeral 2 y 3, correspondientes al puerto VGA y PS/2.

Ejemplo 2. Sumador de 2 números de 8bits.

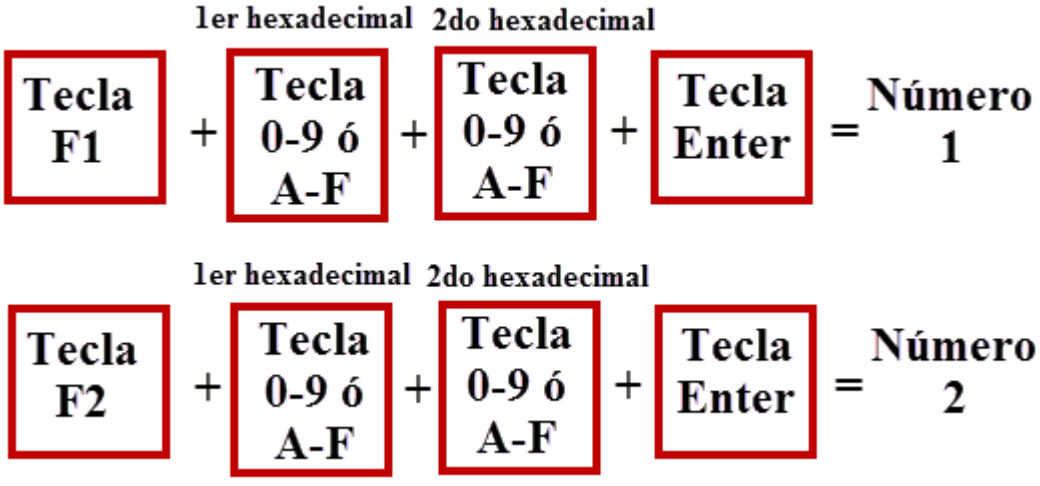
Realizar en VHDL un programa que sume dos números de 8 bits, además, mostrar el resultado de la suma en dos display 7 segmentos y el acarreo en un led.

Para este ejemplo, se incluirá el módulo Adap_2_Num_8Bits y el módulo Conv_7Seg, el cual se muestran a continuación:

Subprograma Adap_2_Num_8Bits

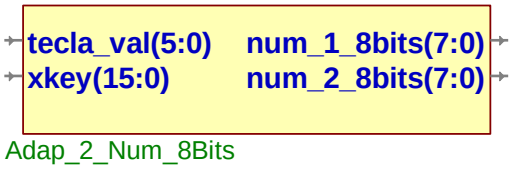
Función.

Este módulo se utiliza para adaptar las salidas del módulo Teclado_Top por una salida de dos números de 8bits cada uno. La cual funciona de la siguiente manera:



Se presiona la tecla “f1” o “f2” para especificar que número añadir. Cada número tendrá un valor de dos hexadecimales, el primer hexadecimal es el más significativo y el 2do hexadecimal el menos significativo. La tecla “enter” se utiliza para finalizar la toma de datos. En el momento de seleccionar los hexadecimales, si se presiona una tecla que no sea numérica o de la “a-f”, entonces el subprograma no lo toma en cuenta y sigue esperando hasta que se presione una tecla numérica o de la “a-f”. Si se presiona la tecla “f1” o “f2” y luego se presiona una tecla numérica o de la “a-f” y por último se presiona la tecla “enter”, entonces el hexadecimal será el más significativo y el menos significativo será cero.

Entidad.



Variables de entrada.

Nombre	Tipo	Descripción
Tecla_Val(5:0)	Std_Logic_Vector(5 downto 0)	Valor asignado del código scan
Xkey(15:0)	Std_Logic_Vector(15 downto 0)	Código scan del teclado

Variables de salida.

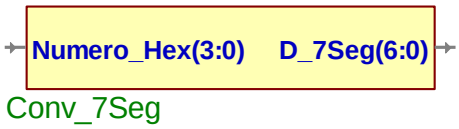
Nombre	Tipo	Descripción
num_1_8bits(7:0)	Std_logic_Vector (7 downto 0)	Número 1 de 8bits
num_2_8bits (7:0)	Std_logic_Vector (7 downto 0)	Número 2 de 8bits

Subprograma Conv_7Seg

Función.

Este módulo convierte de acuerdo al número de 4bits en un número de 8bits para el display 7 segmento.

Entidad.



Variables de entrada.

Nombre	Tipo	Descripción
Numero_Hex(3:0)	Std_Logic_Vector(3 downto 0)	Número de 4 bits

Variables de salida

Nombre	Tipo	Descripción
D_7Seg(6:0)	Std_logic_Vector (6 downto 0)	Dígito de un 7 segmento

```
architecture Conv_7Seg of Conv_7Seg is
begin

   -- enter your statements here --
   process (Numero_Hex)
   begin
      case Numero_Hex is
                  when "0000" => D_7Seg <= "1111110"; -- 0
                  when "0001" => D_7Seg <= "0110000"; -- 1
                  when "0010" => D_7Seg <= "1101101"; -- 2
                  when "0011" => D_7Seg <= "1111001"; -- 3
                  when "0100" => D_7Seg <= "0110011"; -- 4
                  when "0101" => D_7Seg <= "1011011"; -- 5
                  when "0110" => D_7Seg <= "1011111"; -- 6
                  when "0111" => D_7Seg <= "1110010"; -- 7
                  when "1000" => D_7Seg <= "1111111"; -- 8
                  when "1001" => D_7Seg <= "1111011"; -- 9
                  when "1010" => D_7Seg <= "1110111"; -- A
                  when "1011" => D_7Seg <= "0011111"; -- B
                  when "1100" => D_7Seg <= "1001110"; -- C
                  when "1101" => D_7Seg <= "0111101"; -- D
                  when "1110" => D_7Seg <= "1001111"; -- E
                  when others => D_7Seg <= "1000111"; -- F
      end case;
   end process;

end Conv_7Seg;
```

Agregar el código en el cuadro marcado. El código del conversor se encarga se preguntar por el valor del número hexadecimal, de acuerdo a su valor se le asignará el dígito en 7 segmento.

Solución.

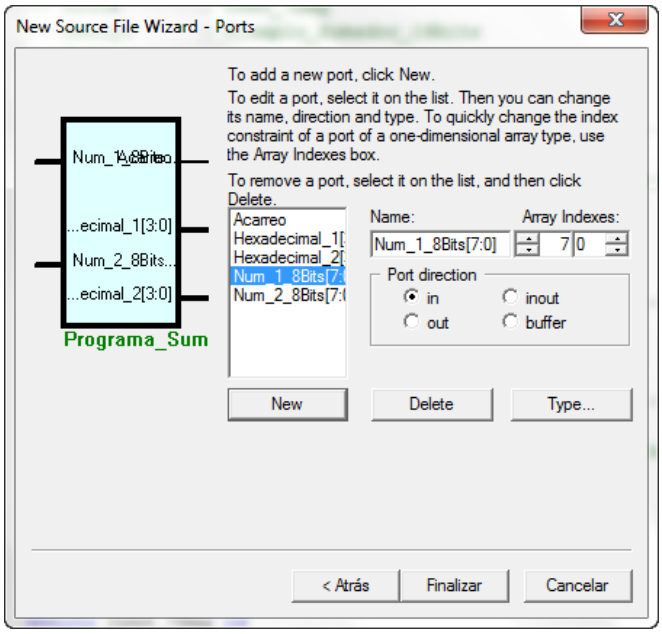
Se inicia partiendo de un proyecto y diseño nuevo, en el cual se agregaron los subprogramas mencionados en el segmento 2 y 3 de este manual de usuario, para el uso del

puerto VGA y PS/2 conectado a un monitor y un teclado respectivamente, además del subprograma Adap_2_Num_8Bits.

Se necesitará crear tres subprogramas, el Subprograma Programa_Sumador, en donde se llevará a cabo la suma de dos números de 8 bits y el acarreo se mostrará en un led virtual en la pantalla del monitor, el Subprograma Conv_7Seg, en donde convierta un número hexadecimal en un display 7 segmento y el Subprograma Top_Programa_Sumador, en donde se realizará el conexiondado de los subprogramas a los pines de la tarjeta.

Subprograma Programa_Sumador.

Este subprograma se realizará en código vhdl.



Variables de entrada.

Nombre	Tipo	Descripción
Num_1_8Bits(7:0)	Std_Logic_Vector(7 downto 0)	Número 1 de 8 bits
Num_2_8Bits(7:0)	Std_Logic_Vector(7 downto 0)	Número 2 de 8 bits

Variables de salida.

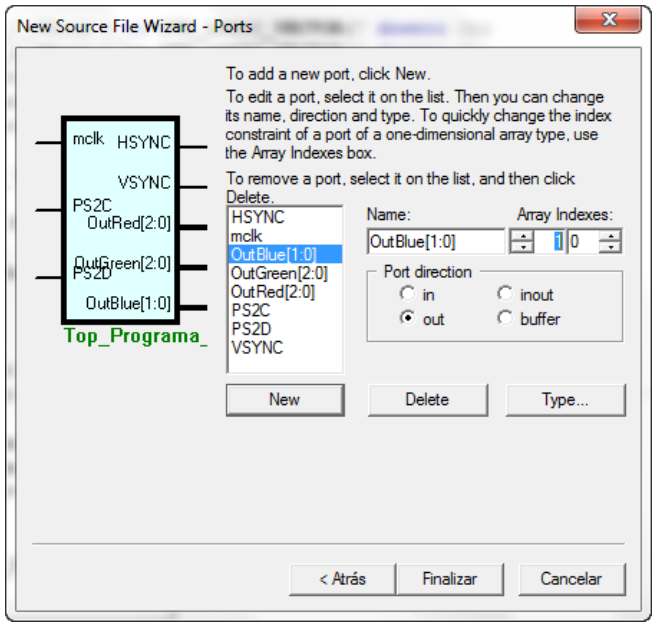
Nombre	Tipo	Descripción
Hexadecimal_1(3:0)	Std_logic_Vector (3 downto 0)	Hexadecimal 1
Hexadecimal_2(3:0)	Std_logic_Vector (3 downto 0)	Hexadecimal 2
Acarreo	Std_Logic	Acarreo de la suma

```
25  library IEEE;
26  use IEEE.STD_LOGIC_1164.all;
27  use IEEE.STD_LOGIC_UNSIGNED.ALL;
28
29  entity Programa_Sumador is
30      port(
31          Num_1_8Bits : in STD_LOGIC_VECTOR(7 downto 0);
32          Num_2_8Bits : in STD_LOGIC_VECTOR(7 downto 0);
33          Acarreo : out STD_LOGIC;
34          Hexadecimal_1 : out STD_LOGIC_VECTOR(3 downto 0);
35          Hexadecimal_2 : out STD_LOGIC_VECTOR(3 downto 0)
36          );
37  end Programa_Sumador;
38
39  architecture Programa_Sumador of Programa_Sumador is
40
41  signal resultado :  std_logic_vector(8 downto 0);
42
43  begin
44
45      resultado(7 downto 0) <= Num_1_8Bits + Num_2_8Bits;
46      resultado(8) <= '1' when Num_1_8Bits(7) = '1' and Num_2_8Bits(7) = '1' else '0';
47
48      Hexadecimal_2 <= resultado (7 downto 4);
49      Hexadecimal_1 <= resultado (3 downto 0);
50      Acarreo <= resultado(8);
51
52
53  end Programa_Sumador;
```

Se debe agregar la librería IEEE.STD_LOGIC_UNSIGNED.ALL para poder realizar operaciones matemáticas mediante lo símbolos de los operadores, en este caso el operador “+”.

Subprograma Top_Programa_Sumador.

Este subprograma se realizará en diagrama de bloques.



Variables de entrada.

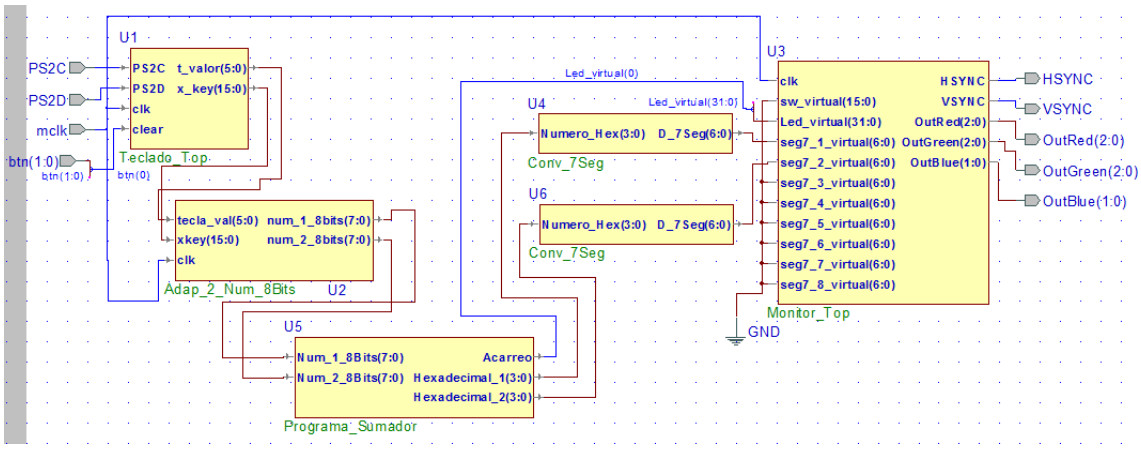
Nombre	Tipo	Descripción
Mclk	Std_Logic	Señal del reloj a 50MHz
PS2C	Std_Logic	Señal del reloj del teclado
PS2D	Std_Logic	Señal de data del teclado

Variables de salida.

Nombre	Tipo	Descripción
HSYNC	Std_logic	Señal de Sincronización Horizontal para el puerto VGA
VSYNC	Std_logic	Señal de Sincronización Vertical para el puerto VGA
OutRed(2:0)	Std_logic_Vector (2 downto 0)	Bits para la pigmentación de los pixeles en Rojo
OutGreen(2:0)	Std_logic_Vector (2 downto 0)	Bits para la pigmentación de los pixeles en Verde
OutBlue(1:0)	Std_logic_Vector (1 downto 0)	Bits para la pigmentación de los

pixeles en Azul

Esquema Top_Programa_Sumador.bde.



Se compilan todos los archivos, se aseguran que no haya ningun error para luego continuar a la síntesis e implementación, en la *Guía rápida de Síntesis e Implementación (con VHDL)*, del Profesor Demetrio Rey Lago [14], explica paso a paso como realizar la síntesis e implementación.

Al momento de configurar la implementación, el archivo *.ucf* a usar será:

-Basys2_Mon_Tec_OsciladorExt.ucf. Si le agregó un oscilador externo a la tarjeta Basys2.

- *Basys2_Mon_Tec_OsciladorInt.ucf*. Si no le agregó un oscilador externo a la tarjeta Basys2.

Después de realizar la implementación sin ningún error, procedemos a programar la tarjeta, transfiriendole el archivo de conexiones generados (extensión *.bit*), en la *Guía rápida de Síntesis e Implementación (con VHDL)*, del Profesor Demetrio Rey Lago [14], explica paso a paso de la programación del dispositivo.

Ejemplo 3. Contador ascendente y descendente.

Realizar en VHDL un programa que al presionar la tecla H suma una unidad, la tecla resta una unidad y la tecla R, para resetear el resultado, mostrando dicho el resultado en dos display 7 segmento en el monitor.

Para este ejemplo, se incluirá el subprograma Adap_Contador, el cual se describirá de la siguiente manera:

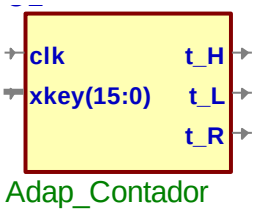
Subprograma Adap_Contador.

Función.

Funciona de la siguiente manera:

Si se presiona la tecla H, L o R, la salida respectiva T_H, T_L y T_R, será tratado como un pulsador, es un 1 binario cuando la tecla es presionada, sino es un 0 binario. Las 3 son independientes.

Entidad.



Variables de entrada.

Nombre	Tipo	Descripción
Clk	Std_Logic	Señal de reloj a 50Mhz
Xkey(15:0)	Std_Logic_Vector(15 downto 0)	Código scan del teclado

Variables de salida.

Nombre	Tipo	Descripción
T_H	Std_Logic	Cuenta ascendente en 1
T_L	Std_Logic	Cuenta descendente en 1
T_R	Std_Logic	Reinicia la cuenta

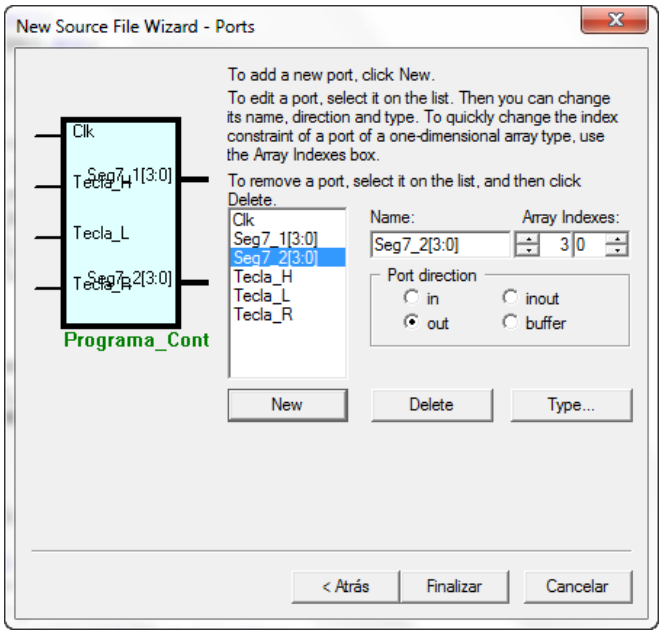
Solución.

Se inicia partiendo de un proyecto y diseño nuevo, en el cual se agregaron los subprogramas mencionados en el segmento 2 y 3 de este manual de usuario, para el uso del puerto VGA y PS/2 conectado a un monitor y un teclado respectivamente,además del subprograma Adap_Contador.

Se necesitará crear tres subprogramas, el Subprograma Programa_Contador, en donde se llevará a cabo la cuenta ascendente, descendente y el reinicio de la cuenta, además tendrá como salida en hexadecimal el resultado de la cuenta, el Subprograma Conv_7Seg (Ejemplo 2), en donde convierte un número hexadecimal en un display 7 segmento y el Subprograma Top_Programa_Sumador, en donde se realizará el conexiondado de los subprogramas a los pines de la tarjeta.

Subprograma Programa_Contador

Este subprograma se realizará en código vhdl.



Variables de entrada

Nombre	Tipo	Descripción
Clk	Std_Logic	Señal de reloj a 50Mhz
Tecla_H	Std_Logic	Cuenta ascendente en 1

Tecla_L	Std_Logic	Cuenta descendente en 1
Tecla_R	Std_Logic	Reinicia la cuenta

Variables de salida

Nombre	Tipo	Descripción
Seg7_1	Std_logic_Vector (3 downto 0)	Hexadecimal 1
Seg7_2	Std_logic_Vector (3 downto 0)	Hexadecimal 2

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity Programa_Contador is
     port(
          Clk : in STD_LOGIC;
          Tecla_H : in STD_LOGIC;
          Tecla_L : in STD_LOGIC;
          Tecla_R : in STD_LOGIC;
          Seg7_1 : out STD_LOGIC_VECTOR(3 downto 0);
          Seg7_2 : out STD_LOGIC_VECTOR(3 downto 0)
          );
end Programa_Contador;

architecture Programa_Contador of Programa_Contador is
signal cuenta: std_logic_vector (7 downto 0):= (others => '0');
signal b1,b2,b3 : std_logic := '0';

begin

    process (Clk,Tecla_H,Tecla_R,Tecla_L)
    begin
        if clk'event and clk = '1' then
            if Tecla_H ='1' then
                b1 <= '1';
            elsif Tecla_H ='0' and b1 = '1' then
                cuenta <= cuenta + 1;
                b1 <= '0';
            elsif Tecla_L ='1'then
                b2 <= '1';
            elsif Tecla_L ='0' and  b2 = '1' then
                cuenta <= cuenta - 1;
                b2 <= '0';
            elsif Tecla_R ='1'then
                b3 <= '1';
            elsif Tecla_R ='0' and  b3 = '1' then
                cuenta <= x"00";
                b3 <= '0';
            else
                cuenta <= cuenta;
            end if;
        end if;
    end process;
    Seg7_1 <= cuenta (7 downto 4);
    Seg7_2 <= cuenta (3 downto 0);

end Programa_Contador;
```

Se debe agregar la librería IEEE.STD_LOGIC_UNSIGNED.ALL para poder realizar operaciones matemáticas mediante lo símbolos de los operadores, en este caso los operadores “+” y ”-”.

Si se presionan más de una tecla, se toman en cuenta ambas, ya que son independientes, es decir, si se presiona a la vez la tecla “l” y la tecla “h”, entonces se incrementa en uno y a la vez decrementa uno.

Subprograma Top_Programa_Contador

Este subprograma se realizará en diagrama de bloques.

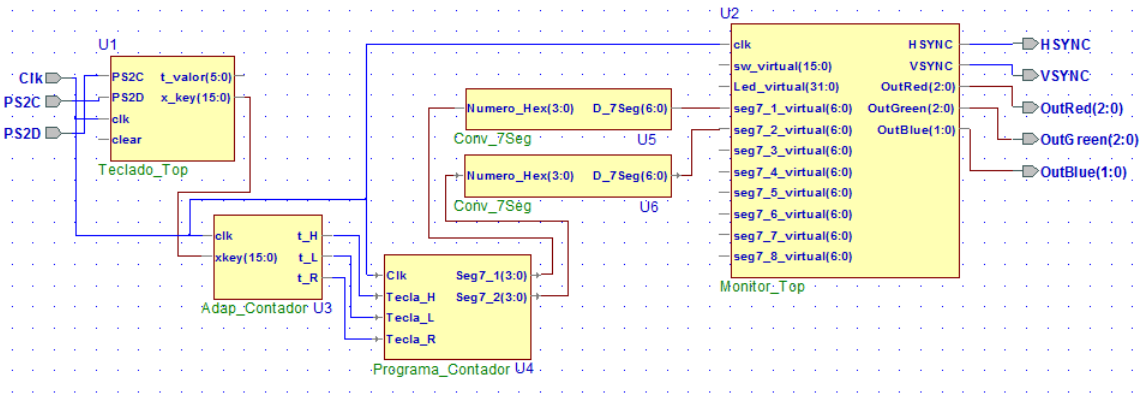
Variables de entrada

Nombre	Tipo	Descripción
Clk	Std_Logic	Señal del reloj a 50MHz
PS2C	Std_Logic	Señal del reloj del teclado
PS2D	Std_Logic	Señal de data del teclado

Variables de salida

Nombre	Tipo	Descripción
HSYNC	Std_logic	Señal de Sincronización Horizontal para el puerto VGA
VSYNC	Std_logic	Señal de Sincronización Vertical para el puerto VGA
OutRed(2:0)	Std_logic_Vector (2 downto 0)	Bits para la pigmentación de los pixeles en Rojo
OutGreen(2:0)	Std_logic_Vector (2 downto 0)	Bits para la pigmentación de los pixeles en Verde
OutBlue(1:0)	Std_logic_Vector (1 downto 0)	Bits para la pigmentación de los pixeles en Azul

Esquema Top_Programa_Contador.bde



Se compilan todos los archivos, se aseguran que no haya ningun error para luego continuar a la síntesis e implementación, en la *Guía rápida de Síntesis e Implementación (con VHDL)*, del Profesor Demetrio Rey Lago [14], explica paso a paso como realizar la síntesis e implementación.

Al momento de configurar la implementación, el archivo *.ucf* a usar será:

-Basys2_Mon_Tec_OsciladorExt.ucf. Si le agregó un oscilador externo a la tarjeta Basys2.

- *Basys2_Mon_Tec_OsciladorInt.ucf*. Si no le agregó un oscilador externo a la tarjeta Basys2.

Después de realizar la implementación sin ningún error, procedemos a programar la tarjeta, transfiriendole el archivo de conexiones generados (extensión *.bit*), en la *Guía rápida de Síntesis e Implementación (con VHDL)*, del Profesor Demetrio Rey Lago [14], explica paso a paso de la programación del dispositivo.

APENDICE D

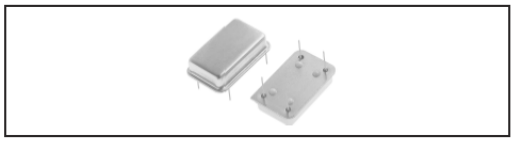
Datasheet del oscilador de cristal de 50MHz Modelo: X0-43

Model XO-43

Vishay Dale

Clock Oscillators

Hybrid Crystal 4.0MHz to 60.0MHz



FEATURES

- TTL compatible.
- Industrial temperature optional.
- Hermetically sealed package.

ELECTRICAL SPECIFICATIONS

Operating Temperature: 0°C to + 70°C (- 40 to 85 optional).
Frequency Stability: .01% Standard (.0025% + .005% optional).
Input Voltage: + 5.0VDC ± 0.5V.
Output Load: 1 to 10 TTL loads.

MECHANICAL SPECIFICATIONS

Marking Ink: Epoxy, solvent resistant.
Hermetically Sealed Package: Leak rate less than 2 x 10^{-8} atmosphere cc/sec. of helium.
Terminal Solderability: A minimum of 95% coverage after solder dip.

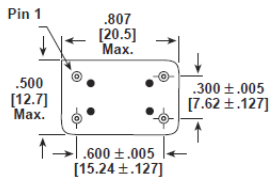
ENVIRONMENTAL SPECIFICATIONS

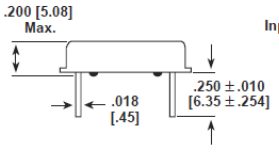
Temperature Cycle: - 55°C to + 85°C, 3 cycles.
Shock: 1000g, 0.35 millisecond, 1/2 sine wave, 3 shocks each plane.
Vibration: .06 D.A., 10 - 55Hz, 20g, 55 - 200Hz.

STANDARD ELECTRICAL SPECIFICATIONS

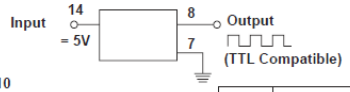
FREQUENCY RANGE (MHz)	INPUT CURRENT (mA) (Max.)	WAVEFORM SYMMETERY @ 1.4VDC	TTL OUTPUT RISE AND FALL TIME (nS) (Max.)	"ZERO" LEVEL SINKING 16mA (Typ. Max.)	"ONE" LEVEL SOURCING 0.4mA (Typ. Min.)
4.0 to 9.999	40	40/60	10	0.5/0.25	3.5/2.4
10.0 to 23.999	40	40/60	8	0.5/0.25	3.5/2.4
24.0 to 60.0	40	40/60	4	0.5/0.25	3.5/2.4

DIMENSIONAL CONFIGURATIONS [Numbers in brackets indicate millimeters]





Schematic



PIN	CONNECTION
1	N.C. or E/D
7	Ground
8	Output
14	+ 5VDC

Pin 1 is identified by square corner. Design subject to change notice.

PART MARKING

- Model
- Frequency
- Pin identifier
- Vishay Dale

HOW TO ORDER

XO-43 / MODEL

B / FREQUENCY STABILITY
- AA = .0025% (25PPM)
- A = .005% (50PPM)
- B = .01% (100PPM)
- Standard

R / OTR
- Blank = 0°C to 70°C
- R = - 40°C to 85°C

40M / FREQUENCY/MHz

NOTE: Contact factory for other models, frequencies, stabilities and temperature ranges.

APENDICE C

Código de programación realizado

Códigos del módulo Teclado_Top y sus módulos internos

Teclado_Top

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use ieee.std_logic_arith.all;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity Teclado_Top is
7        port(
8             PS2C : in STD_LOGIC;
9             PS2D : in STD_LOGIC;
10            clk : in STD_LOGIC;
11            clear : in STD_LOGIC;
12            t_pulsada : out STD_LOGIC;
13            t_valor : out std_logic_vector(5 downto 0);
14            x_key : out std_logic_vector(15 downto 0)
15            );
16  end Teclado_Top;
17
18  architecture Teclado_Top of Teclado_Top is
19  signal xkey_cable: std_logic_vector (15 downto 0);
20  signal t_pulsa : std_logic;
21
22      component Teclado_Lectura
23      port(
24           clk : in STD_LOGIC;
25           clr : in STD_LOGIC;
26           PS2C : in STD_LOGIC;
27           PS2D : in STD_LOGIC;
28           xkey : out STD_LOGIC_VECTOR(15 downto 0));
29      end component;
30      for all: Teclado_Lectura use entity work.Teclado_Lectura(
    Teclado_Lectura);
31
32      component Teclado_Tecla_Valor
33      port(
34           clk : in STD_LOGIC;
35           xkey : in STD_LOGIC_VECTOR(15 downto 0);
36           tecla_pul : out STD_LOGIC;
37           tecla_val : out STD_LOGIC_VECTOR(5 downto 0));
38      end component;
39      for all: Teclado_Tecla_Valor use entity work.Teclado_Tecla_Valor
    Teclado_Tecla_Valor);
40
41  begin
42      t_pulsada <= t_pulsa;
43      x_key <= xkey_cable;
44
45      U1: Teclado_Lectura
46      port map (clk => clk, clr => clear, PS2C => PS2C, PS2D => PS2D,
    xkey => xkey_cable
47      );
48
49      U3: Teclado_Tecla_Valor
50      port map (clk => clk, xkey => xkey_cable, tecla_pul => t_pulsa,
    tecla_val => t_valor
51      );
52
53  end Teclado_Top;
```

Teclado_Valor

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use ieee.std_logic_arith.all;
4
5   entity Teclado_Tecla_Valor is
6        port(
7            clk : in STD_LOGIC;
8            xkey : in STD_LOGIC_VECTOR(15 downto 0);
9            tecla_pul : out STD_LOGIC;
10           tecla_val : out std_logic_vector(5 downto 0)
11           );
12  end Teclado_Tecla_Valor;
13  architecture Teclado_Tecla_Valor of Teclado_Tecla_Valor is
14  signal te_pul : std_logic;
15  signal te_val : std_logic_vector(5 downto 0);
16
17  begin
18      process(xkey)
19      begin
20              if xkey(15 downto 8)= x"F0" or xkey(15 downto 8)= x"00"
    then    --despulsado
21                  te_pul<='1';
22              else
23                  te_pul<='0';    --pulsado
24              end if;
25      end process;
26
27      process (xkey,clk)
28      begin
29          if clk 'event and clk ='1' then
30              case xkey is
31                  when x"F045" => te_val<="000000";   --0
32                  when x"F016" => te_val<="000001";   --1
33                  when x"F01E" => te_val<="000010";   --2
34                  when x"F026" => te_val<="000011";   --3
35                  when x"F025" => te_val<="000100";   --4
36                  when x"F02E" => te_val<="000101";   --5
37                  when x"F036" => te_val<="000110";   --6
38                  when x"F03D" => te_val<="000111";   --7
39                  when x"F03E" => te_val<="001000";   --8
40                  when x"F046" => te_val<="001001";   --9
41                  when x"F01C" => te_val<="001010";   --A
42                  when x"F032" => te_val<="001011";   --B
43                  when x"F021" => te_val<="001100";   --C
44                  when x"F023" => te_val<="001101";   --D
45                  when x"F024" => te_val<="001110";   --E
46                  when x"F02B" => te_val<="001111";   --F
47                  when x"F034" => te_val<="010000";   --G
48                  when x"F033" => te_val<="010001";   --H
49                  when x"F043" => te_val<="010010";   --I
50                  when x"F03B" => te_val<="010011";   --J
```

```
51                  when x"F042" => te_val<="010100";   --K
52                  when x"F04B" => te_val<="010101";   --L
53                  when x"F03A" => te_val<="010110";   --M
54                  when x"F031" => te_val<="010111";   --N
55                  when x"F044" => te_val<="011000";   --O
56                  when x"F04D" => te_val<="011001";   --P
```

```
57                  when x"F015" => te_val<="011010";    --Q
58                  when x"F02D" => te_val<="011011";    --R
59                  when x"F01B" => te_val<="011100";    --S
60                  when x"F02C" => te_val<="011101";    --T
61                  when x"F03C" => te_val<="011110";    --U
62                  when x"F02A" => te_val<="011111";    --V
63                  when x"F01D" => te_val<="100000";    --W
64                  when x"F022" => te_val<="100001";    --X
65                  when x"F035" => te_val<="100010";    --Y
66                  when x"F01A" => te_val<="100011";    --Z
67                  when x"F005" => te_val<="100100";    --F1
68                  when x"F006" => te_val<="100101";    --F2
69                  when x"F004" => te_val<="100110";    --F3
70                  when x"F00C" => te_val<="100111";    --F4
71                  when x"F003" => te_val<="101000";    --F5
72                  when x"F00B" => te_val<="101001";    --F6
73                  when x"F083" => te_val<="101010";    --F7
74                  when x"F00A" => te_val<="101011";    --F8
75                  when x"F001" => te_val<="101100";    --F9
76                  when x"F009" => te_val<="101101";    --F10
77                  when x"F078" => te_val<="101110";    --F11
78                  when x"F007" => te_val<="101111";    --F12
79                  when x"F05A" => te_val<="110000";    --ENTER (48)
80                  when others =>te_val<="110010";    -- 50
81              end case;
82          end if;
83      end process;
84      tecla_val <= te_val;
85      tecla_pul <= te_pul;
86  end Teclado_Tecla_Valor;
87
```

Teclado_Lectura

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3
4   entity Teclado_Lectura is
5        port(
6             clk    : in  STD_LOGIC;
7             clr    : in  STD_LOGIC;
8             PS2C   : in  STD_LOGIC;
9             PS2D   : in  STD_LOGIC;
10            xkey   : out STD_LOGIC_VECTOR(15 downto 0)
11            );
12  end Teclado_Lectura;
13
14  architecture Teclado_Lectura of Teclado_Lectura is
15  signal ps2cf, ps2df : std_logic;
16  signal ps2c_filtro, ps2d_filtro : std_logic_vector(9 downto 0);
17  signal shift1, shift2 : std_logic_vector(10 downto 0);
18  begin
19
20  xkey <= shift2(8 downto 1) & shift1(8 downto 1);
21
22  --Filtro para el reloj y la data de puerto PS2
23
24  filtro: process(clk,clr)
25  begin
26      if clr = '1' then
27          ps2c_filtro <= (others => '0');
28          ps2d_filtro <= (others => '0');
29          ps2cf <= '1';
30          ps2df <= '1';
31      elsif clk   'event and clk = '1' then
32          ps2c_filtro(9) <= PS2C;
33          ps2c_filtro(8 downto 0) <= ps2c_filtro(9 downto 1);
34          ps2d_filtro(9) <= PS2D;
35          ps2d_filtro(8 downto 0) <= ps2d_filtro(9 downto 1);
36          if ps2c_filtro = "11" & X"FF" then
37              ps2cf <= '1';
38          elsif ps2c_filtro ="00" & X"00" then
39              ps2cf <= '0';
40          end if;
41          if ps2d_filtro ="11" & X"FF" then
42              ps2df <= '1';
43          elsif ps2d_filtro ="00" & X"00" then
44              ps2df <= '0';
45          end if;
46      end if;
47  end process filtro;
48
49  --Registro Shift usados para escanear el dato mediante el reloj del
    PS2
50  shift: process(ps2cf, clr)
51  begin
52      if (clr = '1') then
53          shift1 <= (others => '0');
54          shift2 <= (others => '0');
55      elsif (ps2cf 'event and ps2cf =  '0') then
56          shift1 <= ps2df & shift1(10 downto 1);
57          shift2 <= shift1(0) & shift2(10 downto 1);
58      end if;
59  end process shift;
60
```

Códigos del módulo Monitor_Top y sus módulos internos

Monitor_Top

```
1    library IEEE;
2    use IEEE.STD_LOGIC_1164.ALL;
3    use IEEE.STD_LOGIC_ARITH.ALL;
4    use IEEE.STD_LOGIC_UNSIGNED.ALL;
5    entity Monitor_Top is
6        port(
7        clk              :   in  STD_LOGIC;
8        sw_virtual       :   in  std_logic_vector(15 downto 0);
9        Led_virtual      :   in  std_logic_vector(31 downto 0);
10       seg7_1_virtual   :   in  std_logic_vector(6 downto 0);
11       seg7_2_virtual   :   in  std_logic_vector(6 downto 0);
12       seg7_3_virtual   :   in  std_logic_vector(6 downto 0);
13       seg7_4_virtual   :   in  std_logic_vector(6 downto 0);
14       seg7_5_virtual   :   in  std_logic_vector(6 downto 0);
15       seg7_6_virtual   :   in  std_logic_vector(6 downto 0);
16       seg7_7_virtual   :   in  std_logic_vector(6 downto 0);
17       seg7_8_virtual   :   in  std_logic_vector(6 downto 0);
18       HSYNC            :   out STD_LOGIC;
19       VSYNC            :   out STD_LOGIC;
20       OutRed           :   out std_logic_vector(2 downto 0);
21       OutGreen         :   out std_logic_vector(2 downto 0);
22       OutBlue          :   out std_logic_vector(1 downto 0)
23       );
24   end Monitor_Top;
25
26   architecture Monitor_Top of Monitor_Top is
27
28   signal clk_25_cable :   std_logic;
29   signal hout,vout    :   STD_LOGIC_VECTOR(9 downto 0):="0000000000"
30   signal mux          :   STD_LOGIC_VECTOR(3 downto 0):="0000";
31   signal a_video      :   std_logic:='0';
32   signal rgb_sw       :   std_logic_vector(7 downto 0);
33   signal rgb_letra    :   std_logic_vector(7 downto 0);
34   signal rgb_led      :   std_logic_vector(7 downto 0);
35   signal rgb_seg7     :   std_logic_vector(7 downto 0);
36   signal g_let        :   integer;
37   signal d_v          :   std_logic_vector(3 downto 0);
38   signal d_h          :   std_logic_vector(0 to 9);
39
40       component Monitor_Sincronizador
41       port(
42           mclk : in STD_LOGIC;
43           clk25 : out STD_LOGIC;
44           HSYNC : out STD_LOGIC;
45           VSYNC : out STD_LOGIC;
46           act_video : out STD_LOGIC;
47           s_hout : out STD_LOGIC_VECTOR(9 downto 0);
48           s_vout : out STD_LOGIC_VECTOR(9 downto 0));
49       end component;
50       for all: Monitor_Sincronizador use entity work.
     Monitor_Sincronizador(monitor_sincronizador);
51
52       component Monitor_Sw
53       port(
54           clk_25 : in STD_LOGIC;
55           act_video : in STD_LOGIC;
56           s_h : in STD_LOGIC_VECTOR(9 downto 0);
```

```
46            act_video : out STD_LOGIC;
47            s_hout : out STD_LOGIC_VECTOR(9 downto 0);
48            s_vout : out STD_LOGIC_VECTOR(9 downto 0));
49        end component;
50        for all: Monitor_Sincronizador use entity work.
    Monitor_Sincronizador(monitor_sincronizador);
51
52        component Monitor_Sw
53        port(
54            clk_25 : in STD_LOGIC;
55            act_video : in STD_LOGIC;
56            s_h : in STD_LOGIC_VECTOR(9 downto 0);
57            s_v : in STD_LOGIC_VECTOR(9 downto 0);
58            mux_v : in STD_LOGIC_VECTOR(3 downto 0);
59            sw : in STD_LOGIC_VECTOR(15 downto 0);
60            rgb_out : out STD_LOGIC_VECTOR(7 downto 0));
61        end component;
62        for all: Monitor_Sw use entity work.Monitor_Sw(monitor_sw);
63
64        component Monitor_Led
65        port(
66            clk_25 : in STD_LOGIC;
67            act_video : in STD_LOGIC;
68            s_h : in STD_LOGIC_VECTOR(9 downto 0);
69            mux_v : in STD_LOGIC_VECTOR(3 downto 0);
70            Led : in STD_LOGIC_VECTOR(31 downto 0);
71            rgb_out : out STD_LOGIC_VECTOR(7 downto 0));
72        end component;
73        for all: Monitor_Led use entity work.Monitor_Led(monitor_led);
74
75        component Monitor_Seg7
76        port(
77            act_video : in STD_LOGIC;
78            mux_v : in STD_LOGIC_VECTOR(3 downto 0);
79            seg7_1 : in STD_LOGIC_VECTOR(6 downto 0);
80            seg7_2 : in STD_LOGIC_VECTOR(6 downto 0);
81            seg7_3 : in STD_LOGIC_VECTOR(6 downto 0);
82            seg7_4 : in STD_LOGIC_VECTOR(6 downto 0);
83            seg7_5 : in STD_LOGIC_VECTOR(6 downto 0);
84            seg7_6 : in STD_LOGIC_VECTOR(6 downto 0);
85            seg7_7 : in STD_LOGIC_VECTOR(6 downto 0);
86            seg7_8 : in STD_LOGIC_VECTOR(6 downto 0);
87            s_v : in STD_LOGIC_VECTOR(9 downto 0);
88            s_h : in STD_LOGIC_VECTOR(9 downto 0);
89            rgb_out : out STD_LOGIC_VECTOR(7 downto 0));
90        end component;
91        for all: Monitor_Seg7 use entity work.Monitor_Seg7(monitor_seg7
    );
92
93        component Monitor_Mux
94        port(
95            rgb_letra : in STD_LOGIC_VECTOR(7 downto 0);
96            rgb_led : in STD_LOGIC_VECTOR(7 downto 0);
97            rgb_sw : in STD_LOGIC_VECTOR(7 downto 0);
98            rgb_seg7 : in STD_LOGIC_VECTOR(7 downto 0);
99            s_v : in STD_LOGIC_VECTOR(9 downto 0);
100           s_h : in STD_LOGIC_VECTOR(9 downto 0);
101           mux_v : out STD_LOGIC_VECTOR(3 downto 0);
102           O_Red : out STD_LOGIC_VECTOR(2 downto 0);
103           O_Green : out STD_LOGIC_VECTOR(2 downto 0);
104           O_Blue : out STD_LOGIC_VECTOR(1 downto 0));
```

```
105      end component;
106      for all: Monitor_Mux use entity work.Monitor_Mux(monitor_mux);
107
108      component Monitor_Letra_Top
109      port(
110          act_video : in STD_LOGIC;
111          s_h : in STD_LOGIC_VECTOR(9 downto 0);
112          s_v : in STD_LOGIC_VECTOR(9 downto 0);
113          mux_v : in STD_LOGIC_VECTOR(3 downto 0);
114          dir_hor : in STD_LOGIC_VECTOR(0 to 9);
115          g_letra : out INTEGER;
116          dir_vert : out STD_LOGIC_VECTOR(3 downto 0);
117          rgb_out : out STD_LOGIC_VECTOR(7 downto 0));
118      end component;
119      for all: Monitor_Letra_Top use entity work.Monitor_Letra_Top(
     monitor_letra_top);
120
121      component Monitor_Letra
122      port(
123          g_letra : in INTEGER;
124          dir_vert : in STD_LOGIC_VECTOR(3 downto 0);
125          dir_hor : out STD_LOGIC_VECTOR(0 to 9));
126      end component;
127      for all: Monitor_Letra use entity work.Monitor_Letra(
     monitor_letra);
128
129
130  begin
131
132  u1 : Monitor_Sincronizador
133  port map (
134  mclk         => clk,
135  clk25        => clk_25_cable,
136  HSYNC        => HSYNC,
137  VSYNC        => VSYNC,
138  act_video    => a_video,
139  s_hout       =>hout,
140  s_vout       =>vout
141  );
142
143  u2 : Monitor_Sw
144  port map (
145  clk_25       =>  clk_25_cable,
146  act_video    =>  a_video,
147  s_v          =>  vout,
148  s_h          =>  hout,
149  mux_v        =>  mux,
150  sw           =>  sw_virtual,
151  rgb_out      =>  rgb_sw
152  );
153
154  u3 : Monitor_Led
155  port map (
156  clk_25       =>  clk_25_cable,
157  act_video    =>  a_video,
158  s_h          =>  hout,
159  mux_v        =>  mux,
160  Led          =>  Led_virtual,
161  rgb_out      =>  rgb_led
162  );
163
```

```
169  rgb_seg7    =>  rgb_seg7,
170  mux_v       =>  mux,
171  s_v         =>  vout,
172  s_h         =>  hout,
173  O_Red       =>  OutRed,
174  O_Green     =>  OutGreen,
175  O_Blue      =>  OutBlue
176  );
177
178  u6 : Monitor_Letra
179  port map (
180  g_letra     =>  g_let,
181  dir_vert    =>  d_v,
182  dir_hor     =>  d_h
183  );
184
185  u7 : Monitor_Letra_Top
186  port map (
187  act_video   =>  a_video,
188  s_h         =>  hout,
189  s_v         =>  vout,
190  mux_v       =>  mux,
191  dir_hor     =>  d_h,
192  g_letra     =>  g_let,
193  dir_vert    =>  d_v,
194  rgb_out     =>  rgb_letra
195  );
196
197  u8: Monitor_Seg7
198  port map(
199  act_video   =>  a_video,
200  mux_v       =>  mux,
201  seg7_1      =>  seg7_1_virtual,
202  seg7_2      =>  seg7_2_virtual,
203  seg7_3      =>  seg7_3_virtual,
204  seg7_4      =>  seg7_4_virtual,
205  seg7_5      =>  seg7_5_virtual,
206  seg7_6      =>  seg7_6_virtual,
207  seg7_7      =>  seg7_7_virtual,
208  seg7_8      =>  seg7_8_virtual,
209  s_h         =>  hout,
210  s_v         =>  vout,
211  rgb_out     =>  rgb_seg7
212  );
213
214  end Monitor_Top;
215
```

Monitor_Sincronizador

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity Monitor_Sincronizador is
7       port(
8       mclk        : in    STD_LOGIC;                          --Reloj de
    entrada 50Mhz
9       clk25       : out   STD_LOGIC;                          --Reloj de
    salida 25Mhz
10      HSYNC       : out   STD_LOGIC;                          --Señal de
    sincronización horizontal
11      VSYNC       : out   STD_LOGIC;                          --Señal de
    sincronización vertical
12      act_video   : out   STD_LOGIC;                          --Señal que
    define si hay en el monitor
13      s_hout      : out   STD_LOGIC_VECTOR(9 downto 0);       --Posición
    del pixel horizontal
14      s_vout      : out   STD_LOGIC_VECTOR(9 downto 0)        --Posición
    del pixel vertical
15      );
16
17  end Monitor_Sincronizador;
18
19  architecture Monitor_Sincronizador of Monitor_Sincronizador is
20  signal   s_v    : std_logic_vector(9 downto 0) := "0000000000";
    --Contador del pixel vertical
21  signal   s_h    : std_logic_vector(9 downto 0) := "0000000000";
    --Contador del pixel horizontal
22  signal   c_25   : std_logic := '0';
    --Variable auxiliar, reloj a 25Mhz
23  begin
24
25  -- - - - - -  - - - - - - - - - - - - - - - - - - - - - - - - - -
26
27  --PARA OSCILADOR DE 50MHZ
28
29  P1: process(mclk,c_25)
30      begin
31          if(mclk'event and mclk = '1') then
32              c_25 <= not c_25;
33          end if;
34          clk25<=c_25;
35      end process;
36
37  -- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
38
39  --PARA OSCILADOR DE 25MHZ
40  --  c_25<=mclk;
41  --  clk25<=c_25;
42
43  -- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
44
45  --DIRECCIONAMIENTO DEL PIXEL HORIZONTAL Y VERTICAL
46      s_hout<=s_h;
47      s_vout<=s_v;
48
49  --SINCRONIZACION  HORIZONTAL
50  P2: process (c_25,s_h,s_v)
51      begin
```

```
52      if (c_25='1'and c_25'event) then
53          if s_h< 752 and s_h>= 656 then
54              HSYNC<='0';
55              s_h<=s_h+1;
56          else
57              HSYNC<='1';
58              if s_h=800 then
59                  s_h<="0000000000";
60                  if s_v < 521 then
61                      s_v<=s_v+1;
62                  elsif s_v=521 then
63                      s_v<="0000000000";
64                  end if;
65              else
66                  s_h<=s_h+1;
67              end if;
68          end if;
69      end if;
70  end process;
71
72  --SINCRONIZACION  VERTICAL
73  P3: process (c_25,s_v)
74      begin
75      if (c_25='1')and (c_25'event)  then
76          if s_v< 492 and s_v>= 490  then
77              VSYNC<='0';
78          else
79              VSYNC<='1';
80          end if;
81      end if;
82  end process;
83
84  --  ACTIVA LA SEÑAL DE VIDEO
85  act_video <= '1' when ((s_h < 640) and (s_v< 480)) else '0';
86
87  end Monitor_Sincronizador;
88
```

Monitor_Mux

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5   entity Monitor_Mux is
6       port(
7       rgb_letra:  in  std_logic_vector(7 downto 0);       --Señal RGB de
    las letras
8       rgb_led:    in  std_logic_vector(7 downto 0);       --Señal RGB de
    los Leds
9       rgb_sw:     in  std_logic_vector(7 downto 0);       --Señal RGB de
    los switches
10      rgb_seg7:   in  std_logic_vector(7 downto 0);       --Señal RGB de
    los 7 segmentos
11      s_v:        in  std_logic_vector(9 downto 0);       --Posición del
    pixel vertical
12      s_h:        in  std_logic_vector(9 downto 0);       --Posición del
    pixel horizontal
13      mux_v:      out std_logic_vector(3 downto 0);       --Señal de
    salida del multiplexor
14      O_Red:      out std_logic_vector(2 downto 0);       --Señal de
    salida, bits de color rojo
15      O_Green:    out std_logic_vector(2 downto 0);       --Señal de
    salida, bits de color verde
16      O_Blue: out std_logic_vector(1 downto 0)            --Señal de
    salida, bits de color azul
17      );
18  end Monitor_Mux;
19
20  architecture Monitor_Mux of Monitor_Mux is
21  signal rgb_out  : std_logic_vector(7 downto 0);         --Señal RGB
    auxiliar
22  signal Mux_s    : std_logic_vector(3 downto 0);         --Multiplexor
    auxiliar
23
24  begin
25
26  Mux_v <= Mux_s;
27
28  --MULTIPLEXOR MUX_S: SEPARA LA MATRIZ DE PIXELES EN SECCIONES
29
30  Mux_s <=
31  "0000" WHEN ((s_v > 30)and(s_v <= 40)) and ((s_h > 265)and(s_h <= 305
    )) ELSE  --TITULO "LEDS"
32  "0001" WHEN ((s_v > 50)and(s_v <= 70)) and ((s_h >  50)and(s_h <= 595
    )) ELSE  --1RA HILERA DE LED
33  "0010" WHEN ((s_v > 75)and(s_v <= 85)) and ((s_h >  40)and(s_h <= 600
    )) ELSE  --1RA DESCRIPCION HILERA DE LED
34  "0011" WHEN ((s_v > 95)and(s_v <=115)) and ((s_h >  50)and(s_h <= 595
    )) ELSE  --2DA HILERA DE LED
35  "0100" WHEN ((s_v >120)and(s_v <=130)) and ((s_h >  40)and(s_h <= 605
    )) ELSE  --2DA DESCRIPCION HILERA DE LED
36  "0101" WHEN ((s_v >150)and(s_v <=160)) and ((s_h > 250)and(s_h <= 420
    )) ELSE  --TITULO INTERRUPTORES "SWITCHES"
37  "0110" WHEN ((s_v >170)and(s_v <=210)) and ((s_h >  50)and(s_h <= 595
    )) ELSE  --HILERA DE INTERRUPTOR
38  "0111" WHEN ((s_v >215)and(s_v <=225)) and ((s_h >  40)and(s_h <= 600
    )) ELSE  --DESCRIPCION HILERA DE INTERRUPTOR
```

```
39  "1000" WHEN ((s_v >245)and(s_v <=255)) and ((s_h > 250)and(s_h <= 420
    )) ELSE  --TITULO "7DISPLAY"
40  "1001" WHEN ((s_v >263)and(s_v <=310)) and ((s_h >  45)and(s_h <= 505
    )) ELSE  --HILERA DE 7SEGMENTO
41  "1111";
42
43  --MULTIPLEXOR RGB: INDICA QUE SEÑAL RGB SE USA EN CADA SECCIÓN
44  rgb_out <=
45      rgb_letra   when mux_s="0000" or mux_s="0010" or mux_s="0100"
46  or mux_s="0101" or mux_s="0111" or mux_s="1000" else
47      rgb_led     when mux_s="0001" or mux_s="0011" else
48      rgb_sw      when mux_s="0110" else
49      rgb_seg7    when mux_s="1001" else
50      x"00";
51
52  O_Red   <= rgb_out(7 downto 5);
53  O_Green <= rgb_out(4 downto 2);
54  O_Blue  <= rgb_out(1 downto 0);
55
56  end Monitor_Mux;
```

Monitor_Led

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5   entity Monitor_Led is
6       port(
7       clk_25      : in  STD_LOGIC;                                --Reloj a
    25Mhz
8       act_video   : in  std_logic;                                --Señal
    que indica si se encuentra en la matriz visual
9       s_h         : in  std_logic_vector(9 downto 0);             --Posición
    del pixel horizontal
10      mux_v       : in  std_logic_vector(3 downto 0);             --Señal de
    multiplexor
11      Led         : in  std_logic_vector(31 downto 0);            --leds
12      rgb_out     : out std_logic_vector(7 downto 0)              --Señal de
    salida RGB de los Leds
13      );
14  end Monitor_Led;
15  architecture Monitor_Led of Monitor_Led is
16
17  signal rgb: std_logic_vector(7 downto 0):= x"00";               --Bits
    auxiliar,indicador de color
18
19  --PROCEDIMIENTO_LED: INDICA EL ENCENDIDO Y APAGADO DEL LED
20  procedure proceso_led(led : in std_logic;signal rgb : out
    std_logic_vector(7 downto 0)) is
21      begin
22          case led is
23              when '0'    => rgb <= "10010010";
24              when '1'    => rgb <= "00011100";
25              when others => rgb <= "01101100";
26          end case;
27      end proceso_led;
28
29  --PROCEDIMIENTO_HILERA_LED: INDICA LA POSICIÓN DE LOS LED
30  procedure proceso_hilera_led(s_h : in std_logic_vector(9 downto 0);
    led : in std_logic_vector(15 downto 0);signal rgb : out
    std_logic_vector(7 downto 0)) is
31      begin
32          if     ((s_h >  50) and (s_h  <=  70)) then
33              proceso_led(led(15),rgb);
34          elsif ((s_h >  85) and (s_h  <= 105)) then
35              proceso_led(led(14),rgb);
36          elsif ((s_h > 120) and (s_h  <= 140)) then
37              proceso_led(led(13),rgb);
38          elsif ((s_h > 155) and (s_h  <= 175)) then
39              proceso_led(led(12),rgb);
40          elsif ((s_h > 190) and (s_h  <= 210)) then
41              proceso_led(led(11),rgb);
42          elsif ((s_h > 225) and (s_h  <= 245)) then
43              proceso_led(led(10),rgb);
44          elsif ((s_h > 260) and (s_h  <= 280)) then
45              proceso_led(led(9),rgb);
46          elsif ((s_h > 295) and (s_h  <= 315)) then
47              proceso_led(led(8),rgb);
48          elsif ((s_h > 330) and (s_h  <= 350)) then
49              proceso_led(led(7),rgb);
50          elsif ((s_h > 365) and (s_h  <= 385)) then
51              proceso_led(led(6),rgb);
52          elsif ((s_h > 400) and (s_h  <= 420)) then
```

```
53              proceso_led(led(5),rgb);
54          elsif ((s_h > 435) and (s_h  <= 455)) then
55              proceso_led(led(4),rgb);
56          elsif ((s_h > 470) and (s_h  <= 490)) then
57              proceso_led(led(3),rgb);
58          elsif ((s_h > 505) and (s_h  <= 525)) then
59              proceso_led(led(2),rgb);
60          elsif ((s_h > 540) and (s_h  <= 560)) then
61              proceso_led(led(1),rgb);
62          elsif ((s_h > 575) and (s_h  <= 595)) then
63              proceso_led(led(0),rgb);
64          else
65              rgb<="00000000";
66          end if;
67      end proceso_hilera_led;
68
69  begin
70
71  --PROCESO: HILERA DE LEDS
72  process (clk_25,act_video,Led,s_h,mux_v,rgb)
73  begin
74      if (clk_25='1') and (clk_25'event) then
75          if act_video='1' then
76              if    mux_v="0001" then
77                  proceso_hilera_led(s_h,Led(15 downto 0),rgb);
78              elsif mux_v="0011" then
79                  proceso_hilera_led(s_h,Led(31 downto 16),rgb);
80              end if;
81          end if;
82      end if;
83  rgb_out <=  rgb;
84  end process;
85
86  end Monitor_Led;
87
```

Monitor_Sw

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_UNSIGNED.ALL;
4   entity Monitor_Sw is
5       port(
6       clk_25      : in STD_LOGIC;                         --Reloj a
    25Mhz
7       act_video   : in std_logic;                         --Señal que
    indica si se encuentra en la matriz visual
8       s_h         : in std_logic_vector(9 downto 0);      --Posición
    del pixel horizontal
9       s_v         : in std_logic_vector(9 downto 0);      --Posición
    del pixel vertical
10      mux_v       : in std_logic_vector(3 downto 0);      --Señal de
    multiplexor
11      sw          : in std_logic_vector(15 downto 0);     --Switches
12      rgb_out     : out std_logic_vector(7 downto 0)      --Señal de
    salida RGB de los switches
13      );
14  end Monitor_Sw;
15  architecture Monitor_Sw of Monitor_Sw is
16
17  signal rgb: std_logic_vector(7 downto 0):= x"00";       --Bits
    auxiliar,indicador de color
18
19  --PROCEDIMIENTC CICLO: INDICA LA ACTIVACIÓN Y DESACTIVACÍON DEL
    SWITCH
20
21  procedure ciclo(s_v : in std_logic_vector(9 downto 0);sw : in
    std_logic;signal rgb : out std_logic_vector(7 downto 0)) is
22      begin
23          if s_v < 190 then
24              case sw is
25                  when '1'    => rgb<= "00011100";
26                  when others => rgb<= "10010010";
27              end case;
28          else
29              case sw is
30                  when '0'    => rgb<= "11100000";
31                  when others => rgb<= "10010010";
32              end case;
33          end if;
34  end ciclo;
35
36  begin
37
38  --PROCESO: INDICA LA POSICION DE CADA SWITCH EN LA HILERA
39
40      process (clk_25,sw,s_h,s_v,mux_v,rgb)
41      begin
42          if (clk_25='1') and (clk_25'event) then
43              if act_video = '1' and mux_v="0110"  then
44                  if    ((s_h > 50) and (s_h  <= 70))then
45                      ciclo(s_v,sw(15),rgb);
46                  elsif ((s_h > 85) and (s_h  <= 105)) then
47                      ciclo(s_v,sw(14),rgb);
48                  elsif ((s_h > 120) and (s_h  <= 140)) then
49                      ciclo(s_v,sw(13),rgb);
50                  elsif ((s_h > 155) and (s_h  <= 175)) then
51                      ciclo(s_v,sw(12),rgb);
52                  elsif ((s_h > 190) and (s_h  <= 210)) then
```

```
53                      ciclo(s_v,sw(11),rgb);
54                  elsif ((s_h > 225) and (s_h  <= 245)) then
55                      ciclo(s_v,sw(10),rgb);
56                  elsif ((s_h > 260) and (s_h  <= 280)) then
57                      ciclo(s_v,sw(9),rgb);
58                  elsif ((s_h > 295) and (s_h  <= 315)) then
59                      ciclo(s_v,sw(8),rgb);
60                  elsif ((s_h > 330) and (s_h  <= 350))then
61                      ciclo(s_v,sw(7),rgb);
62                  elsif ((s_h > 365) and (s_h  <= 385)) then
63                      ciclo(s_v,sw(6),rgb);
64                  elsif ((s_h > 400) and (s_h  <= 420)) then
65                      ciclo(s_v,sw(5),rgb);
66                  elsif ((s_h > 435) and (s_h  <= 455)) then
67                      ciclo(s_v,sw(4),rgb);
68                  elsif ((s_h > 470) and (s_h  <= 490)) then
69                      ciclo(s_v,sw(3),rgb);
70                  elsif ((s_h > 505) and (s_h  <= 525)) then
71                      ciclo(s_v,sw(2),rgb);
72                  elsif ((s_h > 540) and (s_h  <= 560)) then
73                      ciclo(s_v,sw(1),rgb);
74                  elsif ((s_h > 575) and (s_h  <= 595)) then
75                      ciclo(s_v,sw(0),rgb);
76                  else
77                      rgb<="00000000";
78                  end if;
79              end if;
80          end if;
81          rgb_out<=rgb;
82      end process;
83
84  end Monitor_Sw;
85
```

Monitor_Seg7

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5   entity Monitor_Seg7 is
6        port(
7             act_video  : in  STD_LOGIC;                          --Señal
    que indica si se encuentra en la matriz visual
8             mux_v      : in  STD_LOGIC_vector(3 downto 0);       --Señal de
    multiplexor
9             seg7_1     : in  STD_LOGIC_VECTOR(6 downto 0);       --Señal 7
    segmento 1
10            seg7_2     : in  STD_LOGIC_VECTOR(6 downto 0);       --Señal 7
    segmento 2
11            seg7_3     : in  STD_LOGIC_VECTOR(6 downto 0);       --Señal 7
    segmento 3
12            seg7_4     : in  STD_LOGIC_VECTOR(6 downto 0);       --Señal 7
    segmento 4
13            seg7_5     : in  STD_LOGIC_VECTOR(6 downto 0);       --Señal 7
    segmento 5
14            seg7_6     : in  STD_LOGIC_VECTOR(6 downto 0);       --Señal 7
    segmento 6
15            seg7_7     : in  STD_LOGIC_VECTOR(6 downto 0);       --Señal 7
    segmento 7
16            seg7_8     : in  STD_LOGIC_VECTOR(6 downto 0);       --Señal 7
    segmento 8
17            s_v        : in  STD_LOGIC_VECTOR(9 downto 0);       --Posición
    del pixel vertical
18            s_h        : in  STD_LOGIC_VECTOR(9 downto 0);       --Posición
    del pixel horizontal
19            rgb_out    : out STD_LOGIC_VECTOR(7 downto 0)        --Señal de
    salida RGB de los 7 segmentos
20            );
21  end Monitor_Seg7;
22
23  architecture Monitor_Seg7 of Monitor_Seg7 is
24
25  constant pix_la: std_logic_vector (9 downto 0)  := "0000110010" ; --
    50
26  constant pix_ra: std_logic_vector (9 downto 0)  := "0001010000" ; --
    80
27  constant lin_uf: std_logic_vector (9 downto 0)  := "0100001001" ; --
    265
28  constant lin_df: std_logic_vector (9 downto 0)  := "0100011011" ; --
    283
29
30  --PROCEDIMIENTO SEGMENTOS7: INDICA QUE SEGMENTO DEL 7 SEGMENTO SE
    ACTIVA
31
32  procedure segmentos7 (
33          numero : in STD_LOGIC_vector (6 downto 0);
34          A1, A2, B1, B2 : in STD_LOGIC_vector (9 downto 0);
35          signal C: out STD_LOGIC_vector (7 downto 0)) is
36      begin
37          -- PARA DIBUJAR EL "SEGMENTO A"
38          if numero(6)= '0' and (s_h>=A1) and (s_h<=A2) and (s_v>=B1-1)
    and (s_v<=B1)then
```

```
39              C <= "00011100";
40          -- PARA DIBUJAR EL "SEGMENTO G"
41          elsif numero(0)= '0' and (s_h>=A1) and (s_h<=A2) and (s_v>=B2)
    and (s_v<=B2+1)  then
42              C <= "00011100";
43          -- PARA DIBUJAR EL  "SEGMENTO D"
44          elsif numero(3)= '0' and (s_h>=A1) and (s_h<=A2) and (s_v>=B2+
    B2-B1+1) and (s_v<=B2+B2-B1+2) then
45              C <= "00011100";
46          -- PARA DIBUJAR EL  "SEGMENTO F"
47          elsif numero(1)= '0' and (s_h>=A1-1) and (s_h<=A1) and (s_v>=
    B1) and (s_v<=B2) then
48              C <= "00011100";
49          -- PARA DIBUJAR EL  "SEGMENTO E"
50          elsif numero(2)= '0' and (s_h>=A1-1) and (s_h<=A1) and (s_v>=
    B2+1) and (s_v<=B2+B2-B1+1) then
51              C <= "00011100";
52          -- PARA DIBUJAR EL  "SEGMENTO B"
53          elsif numero(5)= '0' and (s_h>=A2) and (s_h<=A2+1) and (s_v>=
    B1) and (s_v<=B2) then
54              C <= "00011100";
55          -- PARA DIBUJAR EL  "SEGMENTO C"
56          elsif numero(4)= '0' and (s_h>=A2) and (s_h<=A2+1)  and (s_v>=
    B2+1) and (s_v<=B2+B2-B1+1) then
57              C <= "00011100";
58          else
59              C <= "00000000";
60          end if;
61  end;
62
63  begin
64
65      -- PROCESO: INDICA LA POSICION DE LOS 7 SEGMENTOS
66
67      process (act_video,s_h,s_v,seg7_1,seg7_2,seg7_3,seg7_4,seg7_5,
    seg7_6,seg7_7, seg7_8, mux_v)
68          begin
69              if (act_video = '1') then
70                  if  (mux_v = "1001" ) then
71                      if s_h >= pix_la-2 and s_h <= pix_ra+2 and s_v >=
    lin_uf-2 and s_v <= lin_df+lin_df-lin_uf+2 then
72                          segmentos7 (seg7_1 ,pix_la ,pix_ra, lin_uf,
    lin_df, rgb_out);
73                      elsif s_h >= pix_la-2+60 and s_h <= pix_ra+2+60
    and s_v >= lin_uf-2 and s_v <= lin_df+lin_df-lin_uf+2 then
74                          segmentos7 (seg7_2 ,pix_la+60 ,pix_ra+60,
    lin_uf, lin_df, rgb_out);
75                      elsif s_h >= pix_la-2+120 and s_h <= pix_ra+2+120
    and s_v >= lin_uf-2 and s_v <= lin_df+lin_df-lin_uf+2 then
76                          segmentos7 (seg7_3 ,pix_la +120,pix_ra+120,
    lin_uf, lin_df, rgb_out);
77                      elsif s_h >= pix_la-2+180 and s_h <= pix_ra+2+180
    and s_v >= lin_uf-2 and s_v <= lin_df+lin_df-lin_uf+2then
78                          segmentos7 (seg7_4 ,pix_la+180 ,pix_ra+180,
    lin_uf, lin_df, rgb_out);
79                      elsif s_h >= pix_la-2+240 and s_h <= pix_ra+2+240
    and s_v >= lin_uf-2 and s_v <= lin_df+lin_df-lin_uf+2 then
```

```
80                          segmentos7 (seg7_5 ,pix_la+240 ,pix_ra+240,
    lin_uf, lin_df, rgb_out);
81                      elsif s_h >= pix_la-2+300 and s_h <= pix_ra+2+30
    and s_v >= lin_uf-2 and s_v <= lin_df+lin_df-lin_uf+2 then
82                          segmentos7 (seg7_6 ,pix_la+300 ,pix_ra+300,
    lin_uf, lin_df, rgb_out);
83                      elsif s_h >= pix_la-2+360 and s_h <= pix_ra+2+36
    and s_v >= lin_uf-2 and s_v <= lin_df+lin_df-lin_uf+2 then
84                          segmentos7 (seg7_7 ,pix_la +360,pix_ra+360,
    lin_uf, lin_df, rgb_out);
85                      elsif s_h >= pix_la-2+420 and s_h <= pix_ra+2+42
    and s_v >= lin_uf-2 and s_v <= lin_df+lin_df-lin_uf+2 then
86                          segmentos7 (seg7_8 ,pix_la+420 ,pix_ra+420,
    lin_uf, lin_df, rgb_out);
87                      else
88                          rgb_out <= "00000000";
89                      end if;
90                  else
91                      rgb_out <= "00000000";
92                  end if;
93              end if;
94          end process;
95
96  end Monitor_Seg7;
```

Monitor_Letra

```
1    library IEEE;
2    use IEEE.STD_LOGIC_1164.ALL;
3    use IEEE.STD_LOGIC_ARITH.ALL;
4    use IEEE.STD_LOGIC_UNSIGNED.ALL;
5    entity Monitor_Letra is
6        port(
7        g_letra: in integer;                          --Asignación de
     letra
8        dir_vert: in std_logic_vector(3 downto 0);    --Señal de
     posición de la fila vertical de pixel en la letra
9        dir_hor: out std_logic_vector(0 to 9)         --Señal de
     posición del pixel horizontal en la matriz de pixel de la letra
10       );
11   end Monitor_Letra;
12
13   architecture Monitor_Letra of Monitor_Letra is
14
15   type letra is array (0 to 9) of std_logic_vector(0 to 9);
     --Matriz letra 10x10
16
17   --Matriz de pixeles de las letras
18
19   constant letra_0: letra :=(
20   "0000000000",            --0
21   "0011111100",            --1
22   "0100000010",            --2
23   "0100000010",            --3
24   "0100000010",            --4         --0
25   "0100000010",            --5
26   "0100000010",            --6
27   "0100000010",            --7
28   "0100000010",            --8
29   "0011111100"             --9
30   );
31   constant letra_1: letra :=(
32   "0000000000",            --0
33   "0000110000",            --1
34   "0001010000",            --2
35   "0110010000",            --3
36   "0000010000",            --4         --1
37   "0000010000",            --5
38   "0000010000",            --6
39   "0000010000",            --7
40   "0000010000",            --8
41   "0111111110"             --9
42   );
43   constant letra_2: letra :=(
44   "0000000000",            --0
45   "0001111000",            --1
46   "0010000100",            --2
47   "0100000010",            --3
48   "0100000100",            --4         --2
49   "0000011000",            --5
50   "0001100000",            --6
51   "0010000000",            --7
52   "0100000000",            --8
53   "0111111110"             --9
54   );
55   constant letra_3: letra :=(
56   "0000000000",            --0
57   "0011111100",            --1
```

```
58   "0100000010",            --2
59   "0000000010",            --3
60   "0000000010",            --4         --3
61   "0000111100",            --5
62   "0000000010",            --6
63   "0000000010",            --7
64   "0100000010",            --8
65   "0011111100"             --9
66   );
67   constant letra_4: letra :=(
68   "0000000000",            --0
69   "0000001100",            --1
70   "0000010100",            --2
71   "0000100100",            --3
72   "0001000100",            --4         --4
73   "0010000100",            --5
74   "0111111110",            --6
75   "0000000100",            --7
76   "0000000100",            --8
77   "0000000100"             --9
78   );
79   constant letra_5: letra :=(
80   "0000000000",            --0
81   "0011111110",            --1
82   "0100000000",            --2
83   "0100000000",            --3
84   "0100000000",            --4         --5
85   "0011111100",            --5
86   "0000000010",            --6
87   "0000000010",            --7
88   "0000000010",            --8
89   "0011111100"             --9
90   );
91   constant letra_6: letra :=(
92   "0000000000",            --0
93   "0011111110",            --1
94   "0100000000",            --2
95   "0100000000",            --3
96   "0100000000",            --4         --6
97   "0111111100",            --5
98   "0100000010",            --6
99   "0100000010",            --7
100  "0100000010",            --8
101  "0011111100"             --9
102  );
103  constant letra_7: letra :=(
104  "0000000000",            --0
105  "0111111110",            --1
106  "0000000010",            --2
107  "0000000010",            --3
108  "0000000010",            --4         --7
109  "0000000010",            --5
110  "0000000010",            --6
111  "0000000010",            --7
112  "0000000010",            --8
113  "0000000010"             --9
114  );
115  constant letra_8: letra :=(
116  "0000000000",            --0
117  "0011111100",            --1
118  "0100000010",            --2
```

```
119  "0100000010",            --3
120  "0100000010",            --4          --8
121  "0111111110",            --5
122  "0100000010",            --6
123  "0100000010",            --7
124  "0100000010",            --8
125  "0011111100"             --9
126  );
127  constant letra_9: letra :=(
128  "0000000000",            --0
129  "0011111100",            --1
130  "0100000010",            --2
131  "0100000010",            --3
132  "0100000010",            --4          --9
133  "0011111110",            --5
134  "0000000010",            --6
135  "0000000010",            --7
136  "0000000010",            --8
137  "0011111100"             --9
138  );
139  constant letra_a: letra :=(
140  "0000000000",            --0
141  "0011111100",            --1
142  "0100000010",            --2
143  "0100000010",            --3
144  "0100000010",            --4          --A
145  "0111111110",            --5
146  "0100000010",            --6
147  "0100000010",            --7
148  "0100000010",            --8
149  "0100000010"             --9
150  );
151  constant letra_b: letra :=(
152  "0000000000",            --0
153  "0111111000",            --1
154  "0100000100",            --2
155  "0010000010",            --3
156  "0100000100",            --4          --B
157  "0111111000",            --5
158  "0100000100",            --6
159  "0100000010",            --7
160  "0100000100",            --8
161  "0111111000"             --9
162  );
163  constant letra_c: letra :=(
164  "0000000000",            --0
165  "0000111100",            --1
166  "0001000010",            --2
167  "0010000000",            --3
168  "0100000000",            --4          --C
169  "0100000000",            --5
170  "0100000000",            --6
171  "0010000000",            --7
172  "0001000010",            --8
173  "0000111100"             --9
174  );
175  constant letra_d: letra :=(
176  "0000000000",            --0
177  "0111110000",            --1
178  "0100001000",            --2
179  "0100000100",            --3
```

```
180  "0100000010",            --4          --D
181  "0100000010",            --5
182  "0100000010",            --6
183  "0100000100",            --7
184  "0100001000",            --8
185  "0111110000"             --9
186  );
187  constant letra_e: letra :=(
188  "0000000000",            --0
189  "0111111110",            --1
190  "0100000000",            --2
191  "0100000000",            --3
192  "0100000000",            --4          --E
193  "0111111100",            --5
194  "0100000000",            --6
195  "0100000000",            --7
196  "0100000000",            --8
197  "0111111110"             --9
198  );
199  constant letra_f: letra :=(
200  "0000000000",            --0
201  "0111111110",            --1
202  "0100000000",            --2
203  "0100000000",            --3
204  "0100000000",            --4          --F
205  "0100000000",            --5
206  "0111111000",            --6
207  "0100000000",            --7
208  "0100000000",            --8
209  "0100000000"             --9
210  );
211  constant letra_g: letra :=(
212  "0000000000",            --0
213  "0001111100",            --1
214  "0010000010",            --2
215  "0100000000",            --3
216  "0100000000",            --4          --G
217  "0100011110",            --5
218  "0100000010",            --6
219  "0010000010",            --7
220  "0001000100",            --8
221  "0000111000"             --9
222  );
223  constant letra_h: letra :=(
224  "0000000000",            --0
225  "0100000010",            --1
226  "0100000010",            --2
227  "0100000010",            --3
228  "0100000010",            --4          --H
229  "0111111110",            --5
230  "0100000010",            --6
231  "0100000010",            --7
232  "0100000010",            --8
233  "0100000010"             --9
234  );
235  constant letra_i: letra :=(
236  "0000000000",            --0
237  "0111111110",            --1
238  "0000100000",            --2
239  "0000100000",            --3
240  "0000100000",            --4          --I
```

```
241  "0000100000",           --5
242  "0000100000",           --6
243  "0000100000",           --7
244  "0000100000",           --8
245  "0111111110"            --9
246  );
247  constant letra_j: letra :=(
248  "0000000000",           --0
249  "0000011110",           --1
250  "0000000100",           --2
251  "0000000100",           --3
252  "0000000100",           --4         --J
253  "0100000100",           --5
254  "0100000100",           --6
255  "0010000100",           --7
256  "0001001000",           --8
257  "0000110000"            --9
258  );
259  constant letra_k: letra :=(
260  "0000000000",           --0
261  "0100000010",           --1
262  "0100000100",           --2
263  "0100001000",           --3
264  "0100010000",           --4         --K
265  "0111100000",           --5
266  "0100010000",           --6
267  "0100001000",           --7
268  "0100000100",           --8
269  "0100000010"            --9
270  );
271  constant letra_l: letra :=(
272  "0000000000",           --0
273  "0100000000",           --1
274  "0100000000",           --2
275  "0100000000",           --3
276  "0100000000",           --4         --L
277  "0100000000",           --5
278  "0100000000",           --6
279  "0100000000",           --7
280  "0100000000",           --8
281  "0111111110"            --9
282  );
283  constant letra_m: letra :=(
284  "0000000000",           --0
285  "0100000010",           --1
286  "0110000110",           --2
287  "0101001010",           --3
288  "0100110010",           --4         --M
289  "0100000010",           --5
290  "0100000010",           --6
291  "0100000010",           --7
292  "0100000010",           --8
293  "0100000010"            --9
294  );
295  constant letra_n: letra :=(
296  "0000000000",           --0
297  "0100000010",           --1
298  "0110000010",           --2
299  "0101000010",           --3
300  "0100100010",           --4         --N
301  "0100010010",           --5
```

```
302  "0100010010",            --6
303  "0100001010",            --7
304  "0100000110",            --8
305  "0100000010"             --9
306  );
307  constant letra_o: letra :=(
308  "0000000000",            --0
309  "0001111000",            --1
310  "0010000100",            --2
311  "0100000010",            --3
312  "0100000010",            --4          --O
313  "0100000010",            --5
314  "1000000010",            --6
315  "1000000010",            --7
316  "0010000100",            --8
317  "0001111000"             --9
318  );
319  constant letra_p: letra :=(
320  "0000000000",            --0
321  "0111111100",            --1
322  "0100000010",            --2
323  "0100000010",            --3
324  "0100000010",            --4          --P
325  "0111111100",            --5
326  "0100000000",            --6
327  "0100000000",            --7
328  "0100000000",            --8
329  "0100000000"             --9
330  );
331
332  constant letra_q: letra :=(
333  "0000000000",            --0
334  "0011111100",            --1
335  "0100000010",            --2
336  "0100000010",            --3
337  "0100000010",            --4          --Q
338  "0100000010",            --5
339  "0100000010",            --6
340  "0100001010",            --7
341  "0100000110",            --8
342  "0011111110"             --9
343  );
344  constant letra_r: letra :=(
345  "0000000000",            --0
346  "0111111100",            --1
347  "0100000010",            --2
348  "0100000010",            --3
349  "0100000010",            --4          --R
350  "0111111100",            --5
351  "0100010000",            --6
352  "0100001000",            --7
353  "0100000100",            --8
354  "0100000010"             --9
355  );
356  constant letra_s: letra :=(
357  "0000000000",            --0
358  "0001111100",            --1
359  "0010000010",            --2
360  "0100000000",            --3
361  "0010000000",            --4          --S
362  "0001111000",            --5
```

```
363  "0000000100",            --6
364  "0000000010",            --7
365  "0100000100",            --8
366  "0011111000"             --9
367  );
368  constant letra_t: letra :=(
369  "0000000000",            --0
370  "0111111110",            --1
371  "0000100000",            --2
372  "0000100000",            --3
373  "0000100000",            --4          --T
374  "0000100000",            --5
375  "0000100000",            --6
376  "0000100000",            --7
377  "0000100000",            --8
378  "0000100000"             --9
379  );
380  constant letra_u: letra :=(
381  "0000000000",            --0
382  "0100000010",            --1
383  "0100000010",            --2
384  "0100000010",            --3
385  "0100000010",            --4          --U
386  "0100000010",            --5
387  "0100000010",            --6
388  "0100000010",            --7
389  "0100000010",            --8
390  "0011111100"             --9
391  );
392  constant letra_v: letra :=(
393  "0000000000",            --0
394  "0100000010",            --1
395  "0100000010",            --2
396  "0100000010",            --3
397  "0100000010",            --4          --V
398  "0100000010",            --5
399  "0100000010",            --6
400  "0010000100",            --7
401  "0001001000",            --8
402  "0000110000"             --9
403  );
404  constant letra_w: letra :=(
405  "0000000000",            --0
406  "0100000010",            --1
407  "0100000010",            --2
408  "0100000010",            --3
409  "0100000010",            --4          --W
410  "0100000010",            --5
411  "0100110010",            --6
412  "0101001010",            --7
413  "0110000110",            --8
414  "0100000010"             --9
415  );
416  constant letra_x: letra :=(
417  "0000000000",            --0
418  "0100000010",            --1
419  "0010000100",            --2
420  "0001001000",            --3
421  "0000110000",            --4          --X
422  "0000110000",            --5
423  "0000110000",            --6
```

```
424  "0001001000",            --7
425  "0010000100",            --8
426  "0100000010"             --9
427  );
428  constant letra_y: letra :=(
429  "0000000000",            --0
430  "0100000010",            --1
431  "0100000010",            --2
432  "0010000100",            --3
433  "0001001000",            --4         --Y
434  "0000110000",            --5
435  "0000110000",            --6
436  "0000110000",            --7
437  "0000110000",            --8
438  "0000110000"             --9
439  );
440  constant letra_z: letra :=(
441  "0000000000",            --0
442  "0111111110",            --1
443  "0000000100",            --2
444  "0000001000",            --3
445  "0000010000",            --4         --Z
446  "0001111000",            --5
447  "0000100000",            --6
448  "0001000000",            --7
449  "0010000000",            --8
450  "0111111110"             --9
451  );
452  constant letra_00: letra :=(
453  "0000000000",            --0
454  "0000000000",            --1
455  "0000000000",            --2
456  "0000000000",            --3
457  "0000000000",            --4         --
458  "0000000000",            --5
459  "0000000000",            --6
460  "0000000000",            --7
461  "0000000000",            --8
462  "0000000000"             --9
463  );
464  begin
465
466  --PROCESO: ASIGNA UN VALOR A CADA LETRA
467
468  process (dir_vert,g_letra)
469  variable j : integer := 0;
470  begin
471      j := conv_integer(dir_vert);
472      case g_letra is
473          when 0 => dir_hor <= letra_0(j);
474          when 1 => dir_hor <= letra_1(j);
475          when 2 => dir_hor <= letra_2(j);
476          when 3 => dir_hor <= letra_3(j);
477          when 4 => dir_hor <= letra_4(j);
478          when 5 => dir_hor <= letra_5(j);
479          when 6 => dir_hor <= letra_6(j);
480          when 7 => dir_hor <= letra_7(j);
481          when 8 => dir_hor <= letra_8(j);
482          when 9 => dir_hor <= letra_9(j);
483          when 10 => dir_hor <= letra_a(j);
484          when 11 => dir_hor <= letra_b(j);
```

```
485           when 12 => dir_hor <= letra_c(j);
486           when 13 => dir_hor <= letra_d(j);
487           when 14 => dir_hor <= letra_e(j);
488           when 15=> dir_hor <= letra_f(j);
489           when 16 => dir_hor <= letra_g(j);
490           when 17 => dir_hor <= letra_h(j);
491           when 18 => dir_hor <= letra_i(j);
492           when 19 => dir_hor <= letra_j(j);
493           when 20 => dir_hor <= letra_k(j);
494           when 21 => dir_hor <= letra_l(j);
495           when 22 => dir_hor <= letra_m(j);
496           when 23 => dir_hor <= letra_n(j);
497           when 24 => dir_hor <= letra_o(j);
498           when 25 => dir_hor <= letra_p(j);
499           when 26 => dir_hor <= letra_q(j);
500           when 27 => dir_hor <= letra_r(j);
501           when 28 => dir_hor <= letra_s(j);
502           when 29 => dir_hor <= letra_t(j);
503           when 30 => dir_hor <= letra_u(j);
504           when 31 => dir_hor <= letra_v(j);
505           when 32 => dir_hor <= letra_w(j);
506           when 33 => dir_hor <= letra_x(j);
507           when 34 => dir_hor <= letra_y(j);
508           when 35 => dir_hor <= letra_z(j);
509           when others => dir_hor <= letra_00(j);
510       end case;
511   end process;
512
513   end Monitor_Letra;
514
515
516
```

Monitor_Tecla_Top

```
1    library IEEE;
2    use IEEE.STD_LOGIC_1164.ALL;
3    use IEEE.STD_LOGIC_ARITH.ALL;
4    use IEEE.STD_LOGIC_UNSIGNED.ALL;
5    entity Monitor_Letra_Top is
6        port(
7        act_video    : in  std_logic;                        --Señal que
     indica si se encuentra en la matriz visual
8        s_h          : in  std_logic_vector(9 downto 0);    --Posición
     del pixel horizontal
9        s_v          : in  std_logic_vector(9 downto 0);    --Posición
     del pixel vertical
10       mux_v        : in  std_logic_vector(3 downto 0);    --Señal de
     multiplexor
11       dir_hor      : in  std_logic_vector(0 to 9);        --Señal de
     posición del pixel horizontal en la matriz de pixel de la letra
12       g_letra      : out integer;                          --Asignación
     de la letra
13       dir_vert     : out std_logic_vector(3 downto 0);    --Señal de
     posición de la fila vertical de pixel en la letra
14       rgb_out      : out std_logic_vector(7 downto 0)     --Señal de
     salida RGB de los Leds
15       );
16
17   end Monitor_Letra_Top;
18   architecture Monitor_Letra_Top of Monitor_Letra_Top is
19
20   signal rgb: std_logic_vector(7 downto 0):= x"00";
     --Bits auxiliar, indicador de color
21   signal pix_hor : std_logic_vector(9 downto 0):="0000000000";
     --Señal auxiliar del pixel horizontal
22   signal pix_ver : std_logic_vector(9 downto 0):="0000000000";
     --Señal auxiliar del pixel vertical
23
24   --PROCEDIMIENTO CICLO: PIGMENTA LA LETRA
25
26   procedure ciclo(
27   n           : in  integer;
28   dir_hor     : in  std_logic_vector(0 to 9);
29   pix_hor     : in  std_logic_vector(9 downto 0);
30   signal rgb : out std_logic_vector(7 downto 0)
31   ) is
32
33   variable j : integer :=0;
34   variable red,green,blue: std_logic:='0';
35   variable k: std_logic_vector(9 downto 0):="0000000000";
36
37   begin
38       case n is
39           when 1      => k:= pix_hor;
40           when 2      => k:= pix_hor - 10;
41           when 3      => k:= pix_hor - 20;
42           when 4      => k:= pix_hor - 30;
43           when 5      => k:= pix_hor - 40;
44           when 6      => k:= pix_hor - 50;
45           when 7      => k:= pix_hor - 60;
46           when others => k:= pix_hor - 70;
47       end case;
48       j := conv_integer(k(3 downto 0));
49       if dir_hor(j)='1' then
50           red := '0';
```

```
51          green:='1';
52          blue:= '1';
53      else
54          red := '0';
55          green:='0';
56          blue:= '0';
57      end if;
58      rgb<=red & red & red & green & green & green & blue & blue;
59  end ciclo;
60
61  --PROCEDIMIENTO PALABRA: INDICA LA POSICIÓN DE LA LETRA A PIGMENTAR

62
63  procedure palabra(
64  L0,L1,L2,L3,L4,L5,L6,L7 : in  integer;
65  pix_hor                 : in  std_logic_vector(9 downto 0);
66  dir_hor                 : in  std_logic_vector(0 to 9);
67  signal g_letra          : out integer;
68  signal rgb              : out std_logic_vector(7 downto 0)
69  )is
70
71  begin
72      if pix_hor < 10 then
73          g_letra<=L0;
74          ciclo(1,dir_hor,pix_hor,rgb);
75      elsif pix_hor >=10 and pix_hor<20 then
76          g_letra<=L1;
77          ciclo(2,dir_hor,pix_hor,rgb);
78      elsif pix_hor >=20 and pix_hor<30 then
79          g_letra<=L2;
80          ciclo(3,dir_hor,pix_hor,rgb);
81      elsif pix_hor >=30 and pix_hor<40 then
82          g_letra<=L3;
83          ciclo(4,dir_hor,pix_hor,rgb);
84      elsif pix_hor >=40 and pix_hor<50 then
85          g_letra<=L4;
86          ciclo(5,dir_hor,pix_hor,rgb);
87      elsif pix_hor >=50 and pix_hor<60 then
88          g_letra<=L5;
89          ciclo(6,dir_hor,pix_hor,rgb);
90      elsif pix_hor >=60 and pix_hor<70 then
91          g_letra<=L6;
92          ciclo(7,dir_hor,pix_hor,rgb);
93      elsif pix_hor >=70 and pix_hor<80 then
94          g_letra<=L7;
95          ciclo(8,dir_hor,pix_hor,rgb);
96      else
97          rgb <= x"00";
98      end if;
99  end;
100
101
102 begin
103     dir_vert <= pix_ver(3 downto 0);
104     rgb_out  <= rgb ;
105
106 --PROCESO: INDICA LA POSICIÓN DE LAS PALABRAS EN LA SECCION DE LA
    MATRIZ DE PIXELES
107
108     process(mux_v,act_video,s_h,s_v,dir_hor,pix_hor,pix_ver,rgb)
109     begin
```

```
110          if act_video='1' then
111              if    mux_v= "0000" then
112                  --TITULO "LEDS"
113                  pix_ver <= s_v - 31;
114                  pix_hor <= s_h - 266;
115                  palabra(21,14,13,28,36,36,36,36,pix_hor,dir_hor,
     g_letra,rgb);
116              elsif mux_v= "0010" then
117                  --LED: DESCRIPCION 1RA HILERA DE LED
118                  pix_ver <= s_v - 76;
119                  pix_hor <= s_h - 41;
120                  if s_h <100 then
121                      palabra(21,13,1,5,36,36,36,36,pix_hor,dir_hor,
     g_letra,rgb);
122                  elsif s_h >290 and s_h <=320 then
123                      palabra(21,13,8,36,36,36,36,36,pix_hor - 250,
     dir_hor,g_letra,rgb);
124                  elsif s_h >570 then
125                      palabra(21,13,0,36,36,36,36,36,pix_hor - 530,
     dir_hor,g_letra,rgb);
126                  else
127                      rgb<=x"00";
128                  end if;
129              elsif mux_v= "0100" then
130                  --LED: DESCRIPCION 2DA HILERA DE LED
131                  pix_ver <= s_v - 121;
132                  pix_hor <= s_h - 41;
133                  if s_h <100 then
134                      palabra(21,13,3,1,36,36,36,36,pix_hor,dir_hor,
     g_letra,rgb);
135                  elsif s_h >285 and s_h <=325 then
136                      palabra(21,13,2,4,36,36,36,36,pix_hor - 245,
     dir_hor,g_letra,rgb);
137                  elsif s_h >565 then
138                      palabra(21,13,1,6,36,36,36,36,pix_hor - 525,
     dir_hor,g_letra,rgb);
139                  else
140                      rgb<=x"00";
141                  end if;
142              elsif mux_v= "0101" then
143                  --TITULO "SWITCHES"
144                  pix_ver <= s_v - 151;
145                  pix_hor <= s_h - 251;
146                  palabra(28,32,18,29,12,17,14,28,pix_hor,dir_hor,
     g_letra,rgb);
147              elsif mux_v= "0111" then
148                  --BOTON: DESCRIPCION HILERA DE SWITCH
149                  pix_ver <= s_v - 216;
150                  pix_hor <= s_h - 41;
151                  if s_h <100 then
152                      palabra(28,32,1,5,36,36,36,36,pix_hor,dir_hor,
     g_letra,rgb);
153                  elsif s_h >290 and s_h <=320 then
154                      palabra(28,32,8,36,36,36,36,36,pix_hor - 250,
     dir_hor,g_letra,rgb);
155                  elsif s_h >570 then
156                      palabra(28,32,0,36,36,36,36,36,pix_hor - 530,
     dir_hor,g_letra,rgb);
157                  else
158                      rgb<=x"00";
159                  end if;
```

```
160             elsif mux_v= "1000" then
161                 --TITULC "7 SEG"
162                 pix_ver <= s_v - 246;
163                 pix_hor <= s_h - 251;
164                 palabra(7,36,28,14,16,36,36,36,pix_hor,dir_hor,
    g_letra,rgb);
165             elsif mux_v= "1010" then
166                 --7SEG: DESCRIPCION DE HILERA DE 7SEG
167                 pix_ver <= s_v - 316;
168                 pix_hor <= s_h - 41;
169                 if s_h <100 then
170                     palabra(7,28,16,8,36,36,36,36,pix_hor,dir_hor,
    g_letra,rgb);
171                 elsif s_h > 470 then
172                     palabra(7,28,16,1,36,36,36,36,pix_hor - 430,
    dir_hor,g_letra,rgb);
173                 else
174                     rgb<=x"00";
175                 end if;
176             else
177                 rgb<=x"00";
178             end if;
179         end if;
180     end process;
181
182 end Monitor_Letra_Top;
183
184
```

Códigos de adaptadores y convertidores

Conv_7Seg

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3
4   entity Conv_7Seg is
5        port(
6             Numero_Hex : in STD_LOGIC_VECTOR(3 downto 0);
7             D_7Seg : out STD_LOGIC_VECTOR(6 downto 0)
8             );
9   end Conv_7Seg;
10
11  architecture Conv_7Seg of Conv_7Seg is
12  begin
13
14      process (Numero_Hex)
15      begin
16         case Numero_Hex is
17                     when "0000" => D_7Seg <= "1111110"; -- 0
18                     when "0001" => D_7Seg <= "0110000"; -- 1
19                     when "0010" => D_7Seg <= "1101101"; -- 2
20                     when "0011" => D_7Seg <= "1111001"; -- 3
21                     when "0100" => D_7Seg <= "0110011"; -- 4
22                     when "0101" => D_7Seg <= "1011011"; -- 5
23                     when "0110" => D_7Seg <= "1011111"; -- 6
24                     when "0111" => D_7Seg <= "1110010"; -- 7
25                     when "1000" => D_7Seg <= "1111111"; -- 8
26                     when "1001" => D_7Seg <= "1111011"; -- 9
27                     when "1010" => D_7Seg <= "1110111"; -- A
28                     when "1011" => D_7Seg <= "0011111"; -- B
29                     when "1100" => D_7Seg <= "1001110"; -- C
30                     when "1101" => D_7Seg <= "0111101"; -- D
31                     when "1110" => D_7Seg <= "1001111"; -- E
32                     when "1111" => D_7Seg <= "1000111"; -- F
33                     when others => D_7Seg <= "0000000"; -- APAGADO
34          end case;
35     end process;
36
37  end Conv_7Seg;
38
```

Adap_2_Num_8Bits.vhd

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use ieee.std_logic_arith.all;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity Adap_2_Num_8Bits is
7       port(
8       tecla_val:  in std_logic_vector(5 downto 0);
9       xkey    : in std_logic_vector(15 downto 0);
10      clk :   in  std_logic;
11      num_1_8bits : out std_logic_vector(7 downto 0);
12      num_2_8bits : out std_logic_vector(7 downto 0)
13      );
14  end Adap_2_Num_8Bits;
15
16  architecture Adap_2_Num_8Bits of Adap_2_Num_8Bits is
17
18  signal bandera1 : std_logic := '0';
19  signal bandera2 : std_logic := '0';
20  signal bandera3 : std_logic := '0';
21  signal bandera4 : std_logic := '0';
22  signal f1_f2 : std_logic := '0';
23  signal valor1 : std_logic_vector(7 downto 0) := (others => '0');
24  signal valor2 : std_logic_vector(7 downto 0) := (others => '0');
25  signal numero1 : std_logic_vector(7 downto 0) := x"00";
26  signal numero2 : std_logic_vector(7 downto 0) := x"00";
27
28  begin
29
30  num_1_8bits <= numero1;
31  num_2_8bits <= numero2;
32
33      process (xkey,tecla_val,bandera1,bandera2,bandera3)
34      begin
35          if clk'event and clk = '1' then
36              if xkey = x"F005" or xkey = x"F006" then    -- F1
37                  if xkey = x"F005" then
38                      f1_f2 <= '0';
39                  else
40                      f1_f2 <= '1';
41                  end if;
42                  bandera1 <= '1';
43                  valor1 <= "00000000";
44                  valor2 <= "00000000";
45                  bandera2 <= '0';
46                  bandera3 <= '0';
47              elsif bandera1 = '1' and xkey(15 downto 8) = x"F0" and
    tecla_val >=0 and tecla_val <=15 then
48                  valor1(5 downto 0) <= tecla_val;
49                  bandera2 <= '1';
50                  bandera1 <= '0';
51              elsif bandera2 = '1' and xkey(15 downto 8) /= x"F0" then
52                  bandera3 <= '1';
53                  bandera2 <= '0';
54              elsif bandera3 = '1' and xkey(15 downto 8) = x"F0" and
    tecla_val >=0 and tecla_val <=15 then
55                  valor2(5 downto 0) <= tecla_val;
56                  bandera4 <= '1';
57                  bandera3 <= '0';
58              elsif xkey = x"F05A" and (bandera2 = '1' or bandera4 =
    '1') then
```

```
59                  if f1_f2 = '0' then
60                      numero1 <= valor1(3 downto 0) & valor2(3 downto
    0);
61                  else
62                      numero2 <= valor1(3 downto 0) & valor2(3 downto
    0);
63                  end if;
64                  valor1 <= "00000000";
65                  valor2 <= "00000000";
66                  bandera1 <= '0';
67                  bandera2 <= '0';
68                  bandera3 <= '0';
69                  bandera4 <= '0';
70              end if;
71              end if;
72      end process;
73
74  end Adap_2_Num_8Bits;
75
```

Adap_Contador

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.STD_LOGIC_ARITH.ALL;
4   use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6   entity Adap_Contador is
7        port(
8           clk : in std_logic;
9           xkey: in std_logic_vector(15 downto 0);
10          t_H: out std_logic;
11          t_L: out std_logic;
12          t_R: out std_logic
13          );
14  end Adap_Contador;
15
16  architecture Adap_Contador of Adap_Contador is
17
18  signal b1,b2,b3 : std_logic := '0';
19  signal cuenta: std_logic;
20
21  begin
22
23      process(pulsado,xkey)
24      begin
25          if clk'event and clk ='1' then
26              -- tecla H
27              if xkey(7 downto 0) = x"33" and xkey(15 downto 8) /= x
    "F0" then
28                  b1 <= '1';
29                  t_H <= '1';
30              elsif b1 = '1' and xkey = x"F033" then
31                  t_H <= '0';
32                  b1 <= '0';
33              -- tecla L
34              elsif xkey(7 downto 0) = x"4B" and xkey(15 downto 8) /=
    x"F0" then
35                  b2 <= '1';
36                  t_L <= '1';
37              elsif b2 = '1' and xkey = x"F04B" then
38                  t_L <= '0';
39                  b2 <= '0';
40              -- para la tecla R
41              elsif xkey(7 downto 0) = x"2D" and xkey(15 downto 8) /=
    x"F0" then
42                  b3 <= '1';
43                  t_R <= '1';
44              elsif b3 = '1' and xkey = x"F02D" then
45                  t_R <= '0';
46                  b3 <= '0';
47              end if;
48          end if;
49      end process;
50
51  end Adap_Contador;
52
```

Códigos del ejemplo VGA_Stripes

Top_VGA_Stripes

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use ieee.std_logic_unsigned.all;
4
5   entity top_vga_stripes is
6        port(
7        mclk: in std_logic;
8        HSYNC : out STD_LOGIC;
9            VSYNC : out STD_LOGIC;
10            OutRed : out STD_LOGIC_VECTOR(2 downto 0);
11            OutGreen : out STD_LOGIC_VECTOR(2 downto 0);
12            OutBlue : out STD_LOGIC_VECTOR(1 downto 0)
13            );
14  end top_vga_stripes;
15
16  --}} End of automatically maintained section
17
18  architecture top_vga_stripes of top_vga_stripes is
19      -- Component declaration of the
    "Monitor_Sincronizador(monitor_sincronizador)" unit defined in
20      -- file: "./../src/Monitor_Sincronizador.vhd"
21      component Monitor_Sincronizador
22      port(
23          mclk : in STD_LOGIC;
24          clk25 : out STD_LOGIC;
25          HSYNC : out STD_LOGIC;
26          VSYNC : out STD_LOGIC;
27          act_video : out STD_LOGIC;
28          s_hout : out STD_LOGIC_VECTOR(9 downto 0);
29          s_vout : out STD_LOGIC_VECTOR(9 downto 0));
30      end component;
31      for all: Monitor_Sincronizador use entity work.
    Monitor_Sincronizador(monitor_sincronizador);
32      -- Component declaration of the "vga_stripes(vga_stripes)" unit
    defined in
33
34      component vga_stripes
35      port(
36          vidon : in STD_LOGIC;
37          hc : in STD_LOGIC_VECTOR(9 downto 0);
38          vc : in STD_LOGIC_VECTOR(9 downto 0);
39          red : out STD_LOGIC_VECTOR(2 downto 0);
40          green : out STD_LOGIC_VECTOR(2 downto 0);
41          blue : out STD_LOGIC_VECTOR(1 downto 0));
42      end component;
43      for all: vga_stripes use entity work.vga_stripes(vga_stripes);
44
45
46      signal clk25_cable : std_logic;
47      signal act_video_cable : std_logic;
48      signal hc_cable : std_logic_vector(9 downto 0);
49      signal vc_cable : std_logic_vector(9 downto 0);
50  begin
51      Label2 : vga_stripes
52      port map(
53          vidon => act_video_cable,
54          hc => hc_cable,
55          vc => vc_cable,
56          red => OutRed,
57          green => OutGreen,
58          blue => OutBlue
```

```
59      );
60
61   Label1 : Monitor_Sincronizador
62       port map(
63           mclk => mclk,
64           clk25 => clk25_cable,
65           HSYNC => HSYNC,
66           VSYNC => VSYNC,
67           act_video => act_video_cable,
68           s_hout => hc_cable,
69           s_vout => vc_cable
70       );
71
72
73
74
75   end top_vga_stripes;
```

VGA_Stripes

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use ieee.std_logic_unsigned.all;
4
5   entity vga_stripes is
6        port(
7        vidon: in std_logic;
8        hc : in STD_LOGIC_VECTOR(9 downto 0);
9            vc : in STD_LOGIC_VECTOR(9 downto 0);
10           red : out STD_LOGIC_VECTOR(2 downto 0);
11           green : out STD_LOGIC_VECTOR(2 downto 0);
12           blue : out STD_LOGIC_VECTOR(1 downto 0)
13           );
14  end vga_stripes;
15
16  architecture vga_stripes of vga_stripes is
17  begin
18
19       process (vidon, vc)
20       begin
21           red <= "000";
22           green <= "000";
23           blue <= "00";
24           if vidon = '1' then
25               red <= vc (4)& vc (4) &  vc (4);
26               green <= not (vc(4) & vc (4) & vc (4));
27           end if;
28       end process;
29
30  end vga_stripes;
31
```

Códigos de ejemplo VGA_Pulsador

VGA_Pulsador

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use IEEE.STD_LOGIC_UNSIGNED.all;
4
5   entity VGA_Pulsador is
6       port(
7            clk : in std_logic;
8            act_video  : in  STD_LOGIC;
9            hc : in  std_logic_vector(9 downto 0);
10           vc : in  std_logic_vector(9 downto 0);
11           xkey : in STD_LOGIC_VECTOR(15 downto 0);
12           OutRed : out std_logic_vector(2 downto 0);
13           OutGreen : out std_logic_vector(2 downto 0);
14           OutBlue : out std_logic_vector(1 downto 0)
15           );
16  end VGA_Pulsador;
17
18  architecture VGA_Pulsador of VGA_Pulsador is
19
20  signal rgb_aux : std_logic_vector(7 downto 0) := x"00";
21
22  begin
23
24  P1: process (act_video, hc, vc,clk)
25  begin
26      if clk'event and clk= '1' then
27          if act_video = '1' then
28              if ((vc >= 330) and (vc < 360)) and ((hc >= 250) and (hc
    < 300)) then
29                  if xkey(7 downto 0) = x"5A" and xkey(15 downto 8) /=
    x"F0" then  --Si la tecla enter se encuentra pulsada
30                      rgb_aux<="00111000";
31                  else
32                      rgb_aux<="11000000";
33                  end if;
34              else
35                  rgb_aux<="00000000";
36              end if;
37          end if;
38      end if;
39
40      end process;
41  OutRed <= rgb_aux(2 downto 0);
42  OutGreen <= rgb_aux(5 downto 3);
43  OutBlue <= rgb_aux(7 downto 6);
44
45  end VGA_Pulsador;
46
```

Códigos de ejemplo Sumador

Programa_Sumador

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5   entity Programa_Sumador is
6        port(
7             Num_1_8Bits : in STD_LOGIC_VECTOR(7 downto 0);
8             Num_2_8Bits : in STD_LOGIC_VECTOR(7 downto 0);
9             Acarreo : out STD_LOGIC;
10            Hexadecimal_1 : out STD_LOGIC_VECTOR(3 downto 0);
11            Hexadecimal_2 : out STD_LOGIC_VECTOR(3 downto 0)
12            );
13  end Programa_Sumador;
14
15  architecture Programa_Sumador of Programa_Sumador is
16
17  signal resultado :  std_logic_vector(8 downto 0) := "000000000";
18
19  begin
20
21      resultado(7 downto 0) <= Num_1_8Bits + Num_2_8Bits;
22      resultado(8) <= '1' when Num_1_8Bits(7) = '1' and Num_2_8Bits(7)
    = '1' else '0';
23
24      Hexadecimal_2 <= resultado (7 downto 4);
25      Hexadecimal_1 <= resultado (3 downto 0);
26      Acarreo <= resultado(8);
27
28
29  end Programa_Sumador;
30
```

Códigos de ejemplo Contador

Programa_Contador

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.all;
3   use IEEE.STD_LOGIC_UNSIGNED.ALL;
4
5   entity Programa_Contador is
6        port(
7            Clk : in STD_LOGIC;
8            Tecla_H : in STD_LOGIC;
9            Tecla_L : in STD_LOGIC;
10           Tecla_R : in STD_LOGIC;
11           Seg7_1 : out STD_LOGIC_VECTOR(3 downto 0);
12           Seg7_2 : out STD_LOGIC_VECTOR(3 downto 0)
13           );
14  end Programa_Contador;
15
16  architecture Programa_Contador of Programa_Contador is
17  signal cuenta: std_logic_vector (7 downto 0):= (others => '0');
18  signal b1,b2,b3 : std_logic := '0';
19
20  begin
21
22      process (Clk,Tecla_H,Tecla_R,Tecla_L)
23      begin
24          if Clk'event and Clk = '1' then
25              if Tecla_H ='1' then
26                  b1 <= '1';
27              elsif Tecla_H ='0' and b1 = '1' then
28                  cuenta <= cuenta + 1;
29                  b1 <= '0';
30              elsif Tecla_L ='1'then
31                  b2 <= '1';
32              elsif Tecla_L ='0' and  b2 = '1' then
33                  cuenta <= cuenta - 1;
34                  b2 <= '0';
35              elsif Tecla_R ='1'then
36                  b3 <= '1';
37              elsif Tecla_R ='0' and  b3 = '1' then
38                  cuenta <= x"00";
39                  b3 <= '0';
40              else
41                  cuenta <= cuenta;
42              end if;
43          end if;
44      end process;
45      Seg7_1 <= cuenta (3 downto 0);
46      Seg7_2 <= cuenta (7 downto 4);
47
48  end Programa_Contador;
49
```

APENDICE E

Archivos de restricción (extensión .ucf)

-Archivo Basys2_Mon_OsciladorExt.ucf

```
# clock pin for Basys2 Board
#NET "mclk" LOC = "B8";#Signal name = MCLK
NET "mclk" LOC = "M6"; #Signal name = UCLK
#NET "mclk" CLOCK_DEDICATED_ROUTE = FALSE;

# Pin assignment for DispCtl
# Connected to Basys2 onBoard 7seg display
NET "seg<0>" LOC = "L14";
NET "seg<1>" LOC = "H12";
NET "seg<2>" LOC = "N14";
NET "seg<3>" LOC = "N11";
NET "seg<4>" LOC = "P12";
NET "seg<5>" LOC = "L13";
NET "seg<6>" LOC = "M12";
NET "dp" LOC = "N13";
NET "an<3>" LOC = "K14";
NET "an<2>" LOC = "M13";
NET "an<1>" LOC = "J12";
NET "an<0>" LOC = "F12";

# Pin assignment for LEDs
NET "Led<7>" LOC = "G1" ;
NET "Led<6>" LOC = "P4" ;
NET "Led<5>" LOC = "N4" ;
NET "Led<4>" LOC = "N5" ;
NET "Led<3>" LOC = "P6" ;
NET "Led<2>" LOC = "P7" ;
NET "Led<1>" LOC = "M11";
NET "Led<0>" LOC = "M5" ;

# Pin assignment for SWs
NET "sw<7>" LOC = "N3";
NET "sw<6>" LOC = "E2";
NET "sw<5>" LOC = "F3";
NET "sw<4>" LOC = "G3";
NET "sw<3>" LOC = "B4";
NET "sw<2>" LOC = "K3";
NET "sw<1>" LOC = "L3";
NET "sw<0>" LOC = "P11";
#NET "sw<0>"
CLOCK_DEDICATED_ROUTE = FALSE;

NET "btn<3>" LOC = "A7";
NET "btn<2>" LOC = "M4";
NET "btn<1>" LOC = "C11";
NET "btn<0>" LOC = "G12";

# Pin assignment for PS2
#NET "PS2C"    LOC = "B1"   | DRIVE = 2  | PULLUP ;
#NET "PS2D"    LOC = "C3"   | DRIVE = 2  | PULLUP ;

# Pin assignment for VGA
NET "HSYNC"   LOC = "J14"  | DRIVE = 2  | PULLUP ;
NET "VSYNC"   LOC = "K13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<2>"  LOC = "F13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<1>"  LOC = "D13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<0>"  LOC = "C14"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<2>"  LOC = "G14"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<1>"  LOC = "G13"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<0>"  LOC = "F14"  | DRIVE = 2  | PULLUP ;
```

```
NET "OutBlue<1>"  LOC = "J13"  | DRIVE = 2  | PULLUP ;
NET "OutBlue<0>"  LOC = "H13"  | DRIVE = 2  | PULLUP ;
```

-Archivo Basys2_Mon_OsciladorInt.ucf

```
# clock pin for Basys2 Board
NET "mclk" LOC = "B8"; # Bank = 0, Signal name = MCLK
#NET "mclk" LOC = "M6"; # Bank = 2, Signal name = UCLK
#NET "mclk" CLOCK_DEDICATED_ROUTE = FALSE;

# Pin assignment for DispCtl
# Connected to Basys2 onBoard 7seg display
NET "seg<0>" LOC = "L14";
NET "seg<1>" LOC = "H12";
NET "seg<2>" LOC = "N14";
NET "seg<3>" LOC = "N11";
NET "seg<4>" LOC = "P12";
NET "seg<5>" LOC = "L13";
NET "seg<6>" LOC = "M12";
NET "dp" LOC = "N13";

NET "an<3>" LOC = "K14";
NET "an<2>" LOC = "M13";
NET "an<1>" LOC = "J12";
NET "an<0>" LOC = "F12";

# Pin assignment for LEDs
NET "Led<7>" LOC = "G1" ;
NET "Led<6>" LOC = "P4" ;
NET "Led<5>" LOC = "N4" ;
NET "Led<4>" LOC = "N5" ;
NET "Led<3>" LOC = "P6" ;
NET "Led<2>" LOC = "P7" ;
NET "Led<1>" LOC = "M11" ;
NET "Led<0>" LOC = "M5" ;

# Pin assignment for SWs
NET "sw<7>" LOC = "N3";
NET "sw<6>" LOC = "E2";
NET "sw<5>" LOC = "F3";
NET "sw<4>" LOC = "G3";
NET "sw<3>" LOC = "B4";
NET "sw<2>" LOC = "K3";
NET "sw<1>" LOC = "L3";
NET "sw<0>" LOC = "P11";
#NET "sw<0>"
CLOCK_DEDICATED_ROUTE = FALSE;

NET "btn<3>" LOC = "A7";
NET "btn<2>" LOC = "M4";
NET "btn<1>" LOC = "C11";
NET "btn<0>" LOC = "G12";

# Loop back/demo signals
# Pin assignment for PS2
#NET "PS2C"    LOC = "B1"   | DRIVE = 2  | PULLUP ;
#NET "PS2D"    LOC = "C3"   | DRIVE = 2  | PULLUP ;

# Pin assignment for VGA
NET "HSYNC"   LOC = "J14"  | DRIVE = 2  | PULLUP ;
NET "VSYNC"   LOC = "K13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<2>"  LOC = "F13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<1>"  LOC = "D13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<0>"  LOC = "C14"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<2>"  LOC = "G14"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<1>"  LOC = "G13"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<0>"  LOC = "F14"  | DRIVE = 2  | PULLUP ;
NET "OutBlue<1>"  LOC = "J13"  | DRIVE = 2  | PULLUP ;
```

```
NET "OutBlue<0>"  LOC = "H13"  | DRIVE = 2  | PULLUP ;
```

-Archivo Basys2_Mon_Tec_OsciladorExt.ucf

```
# clock pin for Basys2 Board
#NET "mclk" LOC = "B8"; # Bank = 0, Signal name = MCLK
NET "mclk" LOC = "M6"; # Bank = 2, Signal name = UCLK
#NET "mclk" CLOCK_DEDICATED_ROUTE = FALSE;

# Pin assignment for DispCtl
# Connected to Basys2 onBoard 7seg display
NET "seg<0>" LOC = "L14";
NET "seg<1>" LOC = "H12";
NET "seg<2>" LOC = "N14";
NET "seg<3>" LOC = "N11";
NET "seg<4>" LOC = "P12";
NET "seg<5>" LOC = "L13";
NET "seg<6>" LOC = "M12";
NET "dp" LOC = "N13";

NET "an<3>" LOC = "K14";
NET "an<2>" LOC = "M13";
NET "an<1>" LOC = "J12";
NET "an<0>" LOC = "F12";

# Pin assignment for LEDs
NET "Led<7>" LOC = "G1" ;
NET "Led<6>" LOC = "P4" ;
NET "Led<5>" LOC = "N4" ;
NET "Led<4>" LOC = "N5" ;
NET "Led<3>" LOC = "P6" ;
NET "Led<2>" LOC = "P7" ;
NET "Led<1>" LOC = "M11" ;
NET "Led<0>" LOC = "M5" ;

# Pin assignment for SWs
NET "sw<7>" LOC = "N3";
NET "sw<6>" LOC = "E2";
NET "sw<5>" LOC = "F3";
NET "sw<4>" LOC = "G3";
NET "sw<3>" LOC = "B4";
NET "sw<2>" LOC = "K3";
NET "sw<1>" LOC = "L3";
NET "sw<0>" LOC = "P11";
#NET "sw<0>"
CLOCK_DEDICATED_ROUTE = FALSE;

NET "btn<3>" LOC = "A7";
NET "btn<2>" LOC = "M4";
NET "btn<1>" LOC = "C11";
NET "btn<0>" LOC = "G12";

# Loop back/demo signals
# Pin assignment for PS2
NET "PS2C"    LOC = "B1"   | DRIVE = 2  | PULLUP ;
NET "PS2D"    LOC = "C3"   | DRIVE = 2  | PULLUP ;

# Pin assignment for VGA
NET "HSYNC"   LOC = "J14"  | DRIVE = 2  | PULLUP ;
NET "VSYNC"   LOC = "K13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<2>"  LOC = "F13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<1>"  LOC = "D13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<0>"  LOC = "C14"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<2>"  LOC = "G14"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<1>"  LOC = "G13"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<0>"  LOC = "F14"  | DRIVE = 2  | PULLUP ;
NET "OutBlue<1>"  LOC = "J13"  | DRIVE = 2  | PULLUP ;
NET "OutBlue<0>"  LOC = "H13"  | DRIVE = 2  | PULLUP ;
```

-Archivo Basys2_Mon_Tec_OsciladorInt.ucf

```
# clock pin for Basys2 Board
NET "mclk" LOC = "B8"; # Bank = 0, Signal name = MCLK
#NET "mclk" LOC = "M6"; # Bank = 2, Signal name = UCLK
#NET "mclk" CLOCK_DEDICATED_ROUTE = FALSE;

# Pin assignment for DispCtl
# Connected to Basys2 onBoard 7seg display
NET "seg<0>" LOC = "L14";
NET "seg<1>" LOC = "H12";
NET "seg<2>" LOC = "N14";
NET "seg<3>" LOC = "N11";
NET "seg<4>" LOC = "P12";
NET "seg<5>" LOC = "L13";
NET "seg<6>" LOC = "M12";
NET "dp" LOC = "N13";

NET "an<3>" LOC = "K14";
NET "an<2>" LOC = "M13";
NET "an<1>" LOC = "J12";
NET "an<0>" LOC = "F12";

# Pin assignment for LEDs
NET "Led<7>" LOC = "G1" ;
NET "Led<6>" LOC = "P4" ;
NET "Led<5>" LOC = "N4" ;
NET "Led<4>" LOC = "N5" ;
NET "Led<3>" LOC = "P6" ;
NET "Led<2>" LOC = "P7" ;
NET "Led<1>" LOC = "M11" ;
NET "Led<0>" LOC = "M5" ;

# Pin assignment for SWs
NET "sw<7>" LOC = "N3";
NET "sw<6>" LOC = "E2";
NET "sw<5>" LOC = "F3";
NET "sw<4>" LOC = "G3";
NET "sw<3>" LOC = "B4";
NET "sw<2>" LOC = "K3";
NET "sw<1>" LOC = "L3";
NET "sw<0>" LOC = "P11";
#NET "sw<0>"
CLOCK_DEDICATED_ROUTE = FALSE;

NET "btn<3>" LOC = "A7";
NET "btn<2>" LOC = "M4";
NET "btn<1>" LOC = "C11";
NET "btn<0>" LOC = "G12";

# Loop back/demo signals
# Pin assignment for PS2
NET "PS2C"    LOC = "B1"   | DRIVE = 2  | PULLUP ;
NET "PS2D"    LOC = "C3"   | DRIVE = 2  | PULLUP ;

# Pin assignment for VGA
NET "HSYNC"   LOC = "J14"  | DRIVE = 2  | PULLUP ;
NET "VSYNC"   LOC = "K13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<2>"  LOC = "F13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<1>"  LOC = "D13"  | DRIVE = 2  | PULLUP ;
NET "OutRed<0>"  LOC = "C14"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<2>"  LOC = "G14"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<1>"  LOC = "G13"  | DRIVE = 2  | PULLUP ;
NET "OutGreen<0>"  LOC = "F14"  | DRIVE = 2  | PULLUP ;
NET "OutBlue<1>"  LOC = "J13"  | DRIVE = 2  | PULLUP ;
NET "OutBlue<0>"  LOC = "H13"  | DRIVE = 2  | PULLUP ;
```

-Archivo Basys2_Tec_OsciladorExt.ucf

```
# clock pin for Basys2 Board
#NET "mclk" LOC = "B8"; # Bank = 0, Signal name = MCLK
NET "mclk" LOC = "M6"; # Bank = 2, Signal name = UCLK
#NET "mclk" CLOCK_DEDICATED_ROUTE = FALSE;

# Pin assignment for DispCtl
# Connected to Basys2 onBoard 7seg display
NET "seg<0>" LOC = "L14";
NET "seg<1>" LOC = "H12";
NET "seg<2>" LOC = "N14";
NET "seg<3>" LOC = "N11";
NET "seg<4>" LOC = "P12";
NET "seg<5>" LOC = "L13";
NET "seg<6>" LOC = "M12";
NET "dp" LOC = "N13";

NET "an<3>" LOC = "K14";
NET "an<2>" LOC = "M13";
NET "an<1>" LOC = "J12";
NET "an<0>" LOC = "F12";

# Pin assignment for LEDs
NET "Led<7>" LOC = "G1" ;
NET "Led<6>" LOC = "P4" ;
NET "Led<5>" LOC = "N4" ;
NET "Led<4>" LOC = "N5" ;
NET "Led<3>" LOC = "P6" ;
NET "Led<2>" LOC = "P7" ;
NET "Led<1>" LOC = "M11" ;
NET "Led<0>" LOC = "M5" ;

# Pin assignment for SWs
NET "sw<7>" LOC = "N3";
NET "sw<6>" LOC = "E2";
NET "sw<5>" LOC = "F3";
NET "sw<4>" LOC = "G3";
NET "sw<3>" LOC = "B4";
NET "sw<2>" LOC = "K3";
NET "sw<1>" LOC = "L3";
NET "sw<0>" LOC = "P11";
#NET "sw<0>"
CLOCK_DEDICATED_ROUTE = FALSE;

NET "btn<3>" LOC = "A7";
NET "btn<2>" LOC = "M4";
NET "btn<1>" LOC = "C11";
NET "btn<0>" LOC = "G12";

# Loop back/demo signals
# Pin assignment for PS2
NET "PS2C"    LOC = "B1"   | DRIVE = 2  | PULLUP ;
NET "PS2D"    LOC = "C3"   | DRIVE = 2  | PULLUP ;

# Pin assignment for VGA
#NET "HSYNC"   LOC = "J14"  | DRIVE = 2  | PULLUP ;
#NET "VSYNC"   LOC = "K13"  | DRIVE = 2  | PULLUP ;
#NET "OutRed<2>"  LOC = "F13"  | DRIVE = 2  | PULLUP ;
#NET "OutRed<1>"  LOC = "D13"  | DRIVE = 2  | PULLUP ;
#NET "OutRed<0>"  LOC = "C14"  | DRIVE = 2  | PULLUP ;
#NET "OutGreen<2>"  LOC = "G14"  | DRIVE = 2  | PULLUP ;
#NET "OutGreen<1>"  LOC = "G13"  | DRIVE = 2  | PULLUP ;
#NET "OutGreen<0>"  LOC = "F14"  | DRIVE = 2  | PULLUP ;
#NET "OutBlue<1>"  LOC = "J13"  | DRIVE = 2  | PULLUP ;
#NET "OutBlue<0>"  LOC = "H13"  | DRIVE = 2  | PULLUP ;
```

-Archivo Basys2_Tec_OsciladorInt.ucf

```
# clock pin for Basys2 Board
NET "mclk" LOC = "B8"; # Bank = 0, Signal name = MCLK
#NET "mclk" LOC = "M6"; # Bank = 2, Signal name = UCLK
#NET "mclk" CLOCK_DEDICATED_ROUTE = FALSE;

# Pin assignment for DispCtl
# Connected to Basys2 onBoard 7seg display
NET "seg<0>" LOC = "L14";
NET "seg<1>" LOC = "H12";
NET "seg<2>" LOC = "N14";
NET "seg<3>" LOC = "N11";
NET "seg<4>" LOC = "P12";
NET "seg<5>" LOC = "L13";
NET "seg<6>" LOC = "M12";
NET "dp" LOC = "N13";

NET "an<3>" LOC = "K14";
NET "an<2>" LOC = "M13";
NET "an<1>" LOC = "J12";
NET "an<0>" LOC = "F12";

# Pin assignment for LEDs
NET "Led<7>" LOC = "G1" ;
NET "Led<6>" LOC = "P4" ;
NET "Led<5>" LOC = "N4" ;
NET "Led<4>" LOC = "N5" ;
NET "Led<3>" LOC = "P6" ;
NET "Led<2>" LOC = "P7" ;
NET "Led<1>" LOC = "M11" ;
NET "Led<0>" LOC = "M5" ;

# Pin assignment for SWs
NET "sw<7>" LOC = "N3";
NET "sw<6>" LOC = "E2";
NET "sw<5>" LOC = "F3";
NET "sw<4>" LOC = "G3";
NET "sw<3>" LOC = "B4";
NET "sw<2>" LOC = "K3";
NET "sw<1>" LOC = "L3";
NET "sw<0>" LOC = "P11";
#NET "sw<0>"
CLOCK_DEDICATED_ROUTE = FALSE;

NET "btn<3>" LOC = "A7";
NET "btn<2>" LOC = "M4";
NET "btn<1>" LOC = "C11";
NET "btn<0>" LOC = "G12";
```

```
# Loop back/demo signals
# Pin assignment for PS2
NET "PS2C"    LOC = "B1"   | DRIVE = 2  | PULLUP ;
NET "PS2D"    LOC = "C3"   | DRIVE = 2  | PULLUP ;

# Pin assignment for VGA
#NET "HSYNC"   LOC = "J14"  | DRIVE = 2  | PULLUP ;
#NET "VSYNC"   LOC = "K13"  | DRIVE = 2  | PULLUP ;
#NET "OutRed<2>"  LOC = "F13"  | DRIVE = 2  | PULLUP ;
#NET "OutRed<1>"  LOC = "D13"  | DRIVE = 2  | PULLUP ;
#NET "OutRed<0>"  LOC = "C14"  | DRIVE = 2  | PULLUP ;
#NET "OutGreen<2>"  LOC = "G14"  | DRIVE = 2  | PULLUP ;
#NET "OutGreen<1>"  LOC = "G13"  | DRIVE = 2  | PULLUP ;
#NET "OutGreen<0>"  LOC = "F14"  | DRIVE = 2  | PULLUP ;
#NET "OutBlue<1>"  LOC = "J13"  | DRIVE = 2  | PULLUP ;
#NET "OutBlue<0>"  LOC = "H13"  | DRIVE = 2  | PULLUP ;
```

REFERENCIAS BIBLIOGRÁFICAS

[1] Marulanda, A., Pires, M & Delgado, J.(2010) *Ingeniería eléctrica: Hacia una propuesta de currículo por competencias*. Revista de la Facultad de Ingeniería Universidad Central de Venezuela. 25(3).

[2] PEREZ, A, R; SIMONE, A, P. 2006. *Incorporación del lenguaje VHDL para la síntesis, descripción y simulación de circuitos lógicos digitales bajo el estándar IEEE 1076-2002.; Caso de estudio: Cátedra de Sistemas Digitales de la Escuela de Ingeniería Eléctrica, de la Universidad de Carabobo*. Trabajo de Ascenso. Esp. Valencia, Universidad de Carabobo, Facultad de Ingenieria. 343 p.

[3] Manual de referencia Basys2. XILINX. (2010). *Xilinx ISE 6 Software Manuals and Help*, PDF Collection. USA. 12p.

[4] Márquez M. *Viabilidad de la tarjeta Basys 2 para su implementación en el control de un proceso.* Instituto Tecnológico de Tehuacan. Tehuacan. México.

[5] López P, Miguel A; Anzueto R, Alvaro (1 enero de 2013) *Control y despliegue de imágenes en monitor (VGA) mediante el uso de lenguaje VHDL y una placa Nexys 2*. Boletín Nº 35, Unidad Profesional Interdisciplinaria en Ingeniería y Tecnologías Avanzadas.

[6] Haskell R., Hanna D. (2010): *Digital Design Using Digilent FPGA Boards*. Michigan: LBE Books

[7] Manual Esquemático de conexiones de la Basys2, DIGILENT página web http://www.digilentinc.com, PDF Collection. USA. 7p.

[8] Delgado C., Lecha E., Moré M., Terés Ll., Sánchez L. (1993): *Introducción a los lenguajes VHDL, Verilog* UDL/Í. Novática No. 112, España.

[9] Pardo F, Boluda, J. (1999): *VHDL Lenguaje para síntesis y modelado de circuitos. XILINX*. Primera Edición. Valencia, España. Editorial RA-MA.

[10] VENEZUELA, U. C. (13 de Noviembre de 2014). *REVISTA DE LA FACULTAD INGENIERIA DE LA UNIVERSIDAD CENTRAL DE VENEZUELA*. Recuperado el 25 de Enero de 2015, de http://www.scielo.org.ve/scielo.php?pid=S0798-40652010000300008&script=sci_arttext

[11] Maxinez D., Alcalá J. 2002: *VHDL El arte de programar sistemas digitales*, Primera Edición, México, Continental.

[12] Machado F., Borromeo S., Rodriguez C. (2011). *Diseño de sistemas digitales con VHDL*. Primera Edición. Madrid, España.

[13] González, L, C; Rico, A, S. 2010): *Implementación de un procesador elemental en un dispositivo Lógico programable*. Tesis Licenciatura. Universidad Autónoma de Zacatecas. Unidad académica de Ingeniería Eléctrica. Zacatecas, México.

[14] Rey, Lago D. (2012). *Active HDL: Guía rápida de Síntesis e Implementación (con VHDL)*. Departamento de Sistemas y Automática. Escuela de Eléctrica, Facultad de Ingeniería de la Universidad de Carabobo. Valencia, Venezuela

[15] Universidad Pedagógica Experimental Libertador (2006). *Manual de trabajos de grado de especialización y maestría y tesis doctorales*. Tercera Edición: Caracas FEDUPEL.

[16] Herrera J. (2012): *Formulación de una guía de estudio y problemario de sistemas lógicos combinacionales usando lenguaje VHDL*. Trabajo de Ascenso. Universidad de Carabobo. Facultad de Ingeniería. Naguanagua.