ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Probador de Componentes para el Laboratorio de Sistemas Digitales Ing. Ludmila Gorenkova Labikova, usando tecnología SoC (System on Chip) en FPGA"

INFORME DEL PROYECTO DE GRADUACIÓN

Previa la obtención del Título de:

INGENIERO EN TELEMÁTICA

Presentado por:

Génesis Paola Gallegos León

Lenin Antonio Rodríguez Vera

GUAYAQUIL – ECUADOR

Año 2015

AGRADECIMIENTO

A Dios por bendecirme con la familia que tengo.

A mis padres por darme la oportunidad de seguir estudiando, por todo ese amor y palabras de aliento que me brindaron en toda mi carrera.

A mis hermanos por ser mi alegría y mi ejemplo de superación, por contagiarme de sus ganas de salir adelante.

A mi enamorado por ser un hombre constante que sabe darme las fuerzas para seguir luchando en la vida.

Al Msc. Ponguillo por compartir sus conocimientos y ayuda para culminar este proyecto con éxito.

Génesis Paola Gallegos León

A Dios por permitir que esto sea posible.

A mis padres que nunca dejaron de creer en mí y soy lo que soy por ellos.

A nuestro tutor Msc. Ronald Ponguillo por su ayuda y consejos en la elaboración de nuestro proyecto.

A Génesis Gallegos por darme un ejemplo de vida, lucha y superación. Y a todos mis amigos que de una u otra forma ayudaron para que este proyecto se lleve a cabo.

Lenin Antonio Rodríguez Vera

DEDICATORIA

A Dios, a mi familia y a mis dos pequeños angelitos, mis hijas, que cambiaron mi vida para bien, que con sus miradas y risas me dieron el empuje necesario para cumplir mis metas.

Génesis Paola Gallegos León

A mi madre porque se merece esto y mucho más, por ser una mujer comprometida y dispuesta a ayudar, porque siempre creyó en mí y se nunca me va a fallar.

A mi hija por ser motivo de inspiración y motivación y por ser ese angelito que ilumina mi vida. Te Amo Solange. Dios, y a mi familia por todos los consejos brindados.

Lenin Antonio Rodríguez Vera

TRIBUNAL DE SUSTENTACIÓN

Msc. Carlos Salazar L.

PRESIDENTE DEL TRIBUNAL DE GRADO

Msc. Ronald Ponguillo I.

DIRECTOR DEL PROYECTO DE GRADUACIÓN

Msc. Carlos Valdivieso A.

MIEMBRO PRINCIPAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este informe, nos corresponde exclusivamente; y el patrimonio intelectual del mismo a la ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL". (Reglamento de exámenes y títulos profesionales de la ESPOL).

_____

Génesis Paola Gallegos León

_____

Lenin Antonio Rodríguez Vera

RESUMEN

El proyecto consiste en desarrollar un prototipo de Probador de Componentes usando Plataforma NIOS II, que es una versión mejorada de un proyecto de graduación anterior que busca facilitar y automatizar las pruebas de elementos en el Laboratorio de Sistemas Digitales Ing. Ludmila Gorenkova Labikova.

Para el desarrollo del proyecto hemos utilizado la tarjeta de desarrollo DE-0 Nano de Altera basada en un dispositivo Cyclone IV® EP4C22 FPGA, en el que se ha diseñado la arquitectura de un mini-computador basado en el Microprocesador NIOS II.

El sistema se ha divido en etapas, en las cuales se encuentra la adquisición y procesamiento de los datos para luego ser visualizados en una pantalla LCD 4x20. El proyecto se lo ha estructurado en 4 capítulos que se detallan a continuación:

En el capítulo 1, damos a conocer las generalidades, se describe el problema existente, su justificación y propuesta de solución, así como también los objetivos generales y específicos del proyecto, planteando los alcances, la metodología y resultados esperados.

En el capítulo 2, se describe la parte teórica, detallando cada elemento de la tecnología utilizada para el desarrollo del proyecto.

En el capítulo 3, se muestra el diseño e implementación del proyecto, donde se describe cada etapa del sistema que se implementó, además de la programación para la verificación de los circuitos integrados digitales.

En el capítulo 4, se presenta los resultados obtenidos de las diferentes pruebas realizadas al sistema. Finalmente, las conclusiones y recomendaciones del proyecto.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

ADC	Convertidor Analógico - Digital
CI	Circuitos Integrados
CLK	Reloj
CMOS	Semiconductor complementario de óxido metálico
DAC	Convertidor Digital – Analógico
DCE	Equipo de comunicación de datos
DTE	Equipo terminal de datos
HDL	Lenguaje de descripción de hardware
FPGA	Matriz de puertas lógicas programables
GPIO	Entrada/Salida de propósito general
LCD	Pantalla de cristal líquido
PCB	Placa de circuitos impresos
RX	Línea de recepción
SPI	Interfaz de periféricos serie

TX	Línea de transmisión
TTL	Lógica transistor a transistor
UART	Transmisor-Receptor Asíncrono Universal

INTRODUCCIÓN

En la actualidad, el gobierno busca innovar en tecnología. Por lo cual, las universidades se enfocan en incentivar a sus estudiantes a desarrollar nuevos proyectos, aprovechando recursos de tecnología de electrónica digital actual que posee la institución y aplicando sus conocimientos adquiridos durante su estadía en la universidad. Por lo tanto, se espera obtener proyectos de características comerciales, que sean fiables y escalables.

Debido a que el Laboratorio de Sistemas Digitales Ing. Ludmila Gorenkova Labikova de la Escuela Superior Politécnica del Litoral está en proceso de actualización y automatización, se están empleando nuevas tecnologías como son las FPGA’s.

Para mejorar los recursos que posee actualmente el Laboratorio se ha desarrollado un probador de componentes fiable y escalable, usando

dispositivos FPGA (Field Programmable Gate Array) por su versatilidad y porque permite al usuario crear, simular e implementar sus propios diseños de acuerdo a sus necesidades.

Decidimos usar la tarjeta DE0-Nano Cyclone IV de Altera, ya que posee características útiles para la implementación de nuestro proyecto, por ejemplo, permite la adquisición de señales analógicas a través de su ADC ubicado en los pines de expansión JP3, además cuenta con una gran cantidad de pines de expansión que satisface los requerimientos del proyecto.

.

CAPÍTULO 1

1. ANTECEDENTES Y JUSTIFICACIÒN

En este capítulo se describe el problema existente, se realiza una justificación, una propuesta de solución, así como también se plantean los objetivos generales y específicos del proyecto, dando a conocer los alcances, la metodología y los resultados esperados.

.

1.1 DESCRIPCIÓN DEL PROBLEMA

Nos encontramos con un sistema de verificación de componentes muy rústico en el laboratorio de digitales, actualmente hay componentes que se verifican con ayuda de protoboard. Identificamos que se necesita de un equipo que sea capaz de probar los componentes que se encuentran en el Laboratorio de Sistemas Digitales, ya que es necesario mantener el stock de los elementos en buen estado para que sean utilizados por los estudiantes en sus proyectos de laboratorio de sistemas digitales.

1.2 JUSTIFICACIÓN

Debido a que en el laboratorio de sistemas digitales existe un probador de circuitos integrados de uso limitado, que depende de un ordenador para realizar las pruebas y que no es capaz de determinar el funcionamiento de todos los componentes que posee el laboratorio.

Además, en una tesis anterior se implementó un sistema automatizado del proceso de préstamo y devolución de los implementos del laboratorio de sistemas digitales, el cual se enfocó

más en el desarrollo del software y dejaron un prototipo de probador de elementos, que es funcional pero que aún verifica componentes de forma visual y manual, por lo que quedaron cosas por mejorar.

1.3 PROPUESTA DE SOLUCIÓN

Desarrollar e implementar un probador de componentes en base a un prototipo realizado previamente en la tesis realizada por Giler - Quinteros, el cual usará tecnología de electrónica digital actual, permitiendo que sea un producto con características comerciales.[1]

1.4 OBJETIVOS

1.4.1 OBJETIVO GENERAL

Construir un equipo de características comerciales basado en tecnología digital actual para el Laboratorio de Sistemas Digitales.

1.4.2 OBJETIVOS ESPECÍFICOS

- Construir un equipo que se adapte al proceso de actualización y automatización del laboratorio utilizando electrónica digital actual.

- Diseñar un circuito capaz de verificar en forma automática el estado de componentes tales como: convertidores ADC/DAC, display´s, bus de datos, oscilador IC555.

- Implementar una pantalla LCD para la visualización del estado de los componentes.

- Diseñar la nueva arquitectura de la mini-computadora basado en el microprocesador NIOS II.

- Crear el código en lenguaje C necesario para realizar la automatización del proceso de verificación de componentes.

- Integrar el nuevo módulo al hardware y software ya implementados en el laboratorio de sistemas digitales.

1.5 ALCANCES

Entre los alcances del proyecto se tiene:

- Es capaz de probar componentes tales como, osciladores IC555, convertidores ADC/DAC, bus de datos, display´s y circuitos integrados digitales.

- El probador funciona sin necesidad de estar conectado con un ordenador, ya que cuenta con una pantalla LCD donde se puede visualizar el estado de los componentes.

- Se ha desarrollado un código en lenguaje C para NIOS II IDE que permite determinar y verificar el correcto funcionamiento de los componentes.

1.6 METODOLOGÍA

La metodología se llevará a cabo con el desarrollo de las siguientes etapas:

- Etapa 1: Análisis de posibles mejoras que se pueden realizar al prototipo que ya existe en el laboratorio de sistemas digitales.

- Etapa 2: Diseño del circuito probador automático, usando diagramas de bloques y esquemáticos.
- Etapa 3: Implementación en protoboard.
- Etapa 4: Pruebas realizadas en protoboard.
- Etapa 5: Construcción de PCB.
- Etapa 6: Ensamble de PCB.
- Etapa 7: Pruebas en el PCB.
- Etapa 8: Integración con el software desarrollado en un trabajo previo e implementado en el laboratorio.
- Etapa 9: Pruebas finales en ambiente de producción en el laboratorio de sistemas digitales.

1.7 RESULTADOS ESPERADOS

Se obtendrá un equipo de características comerciales que tendrá compatibilidad con el software instalado en el laboratorio de sistemas digitales.

CAPÍTULO 2

2. MARCO TEÓRICO

En este capítulo se muestra las características principales de los componentes que se han utilizado para el desarrollo de este proyecto. Además se da información detallada sobre la Tarjeta DE0-Nano de Altera, FPGA Cyclone IV y el Microprocesador NIOS II.

2.1 PLATAFORMA TECNOLÓGICA

2.1.1 TARJETA DE DESARROLLO DE0-NANO

La DE0-Nano es una tarjeta de desarrollo de Altera, que contiene una matriz de puertas lógicas programables (FPGA), que por su tamaño compacto es ideal para diseñar e implementar prototipos de proyectos portátiles como robots y circuitos de diseño digital. [2]

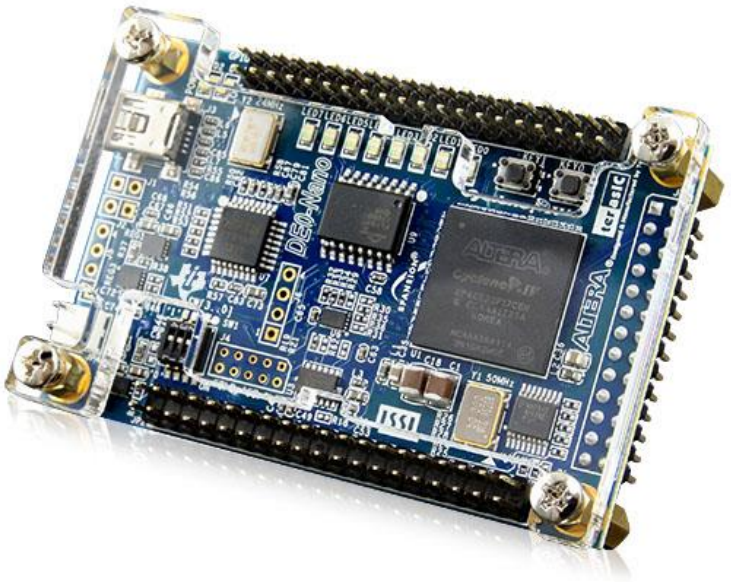


Figura 2-1 Tarjeta de desarrollo DE0-Nano [2]

La tarjeta de desarrollo DE0-Nano posee las siguientes características: [2]

Características	Descripción
FPGA	Cyclone IV EP4CE22F17C6 con EPCS64. 153 pines de E/S.
Interfaces I/O	Entrada
Switches, LED's y pulsadores	8 LED's verdes. 2 Pulsadores. 1 Switch DIP de 4 posiciones.
Dispositivos de Memoria	SDRAM de 32 MB. EEPROM I2C de 2KB.
Reloj	Oscilador de 50 MHz.
Interfaces I/O	Dos conectores de expansión de 40 pines (GPIO´s) que proveen 72 pines de I/O
Convertidor A/D	Convertidor A/D (NS ADC128S022) de 12 bits de 8 canales que muestrea de 50 Ksps a 200 Ksps.

Tabla I Características de la DE0-Nano [3]

2.1.2 FPGA CYCLONE IV EP4C22

FPGA es un dispositivo semiconductor diseñado para ser configurado por el programador después de su fabricación. La configuración de una FPGA es generalmente usando un lenguaje de Descripción de Hardware.

Las FPGA´s internamente tienen componentes lógicos programables llamados Bloques Lógicos con interconexión reprogramable que permiten a los bloques ser cableados entre sí. Los bloques lógicos pueden ser configurados para realizar complejas funciones combinatoriales u otras básicas como compuertas lógicas. [4]

Figura 2-2 FPGA Cyclone IV [4]

2.1.3 MICROPROCESADOR NIOS II

NIOS II es un Microprocesador de 32 bits diseñado exclusivamente para las FPGA de Altera, que proporciona una infraestructura completa para crear sistemas de Microprocesador Embebido según las necesidades del diseñador, por medio de la combinación de una serie de componentes configurables sobre sus FPGAs.

Para desarrollar Sistemas Embebidos, Altera proporciona un entorno de desarrollo al que denomina Qsys, que permite la configuración a medida del sistema Microprocesador NIOS II y que gracias a la herramienta de síntesis Quartus II puede ser implementado directamente sobre una FPGA.

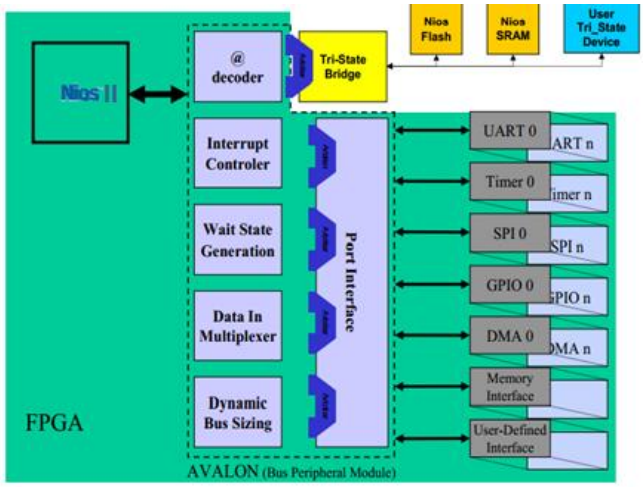


Figura 2-3 Microprocesador basado en NIOS II [5]

Este Sistema Microprocesador está compuesto por el núcleo NIOS II, memoria interna de programa y de datos, periféricos integrados e interfaces para memoria externa y/o entrada/salida.

El microprocesador NIOS II puede ser configurado con 3 diferentes versiones como me muestra a continuación: [5]

- El NIOS II/f (“fast”) es la versión diseñada para un rendimiento superior, y que proporciona opciones de configuración para aumentar su desempeño, como memorias caché de instrucciones y datos, o una unidad de manejo de memoria (MMU, Memory Management Unit).

- El NIOS II/s (“standard”) es la versión que contiene la unidad aritmético lógica (ALU, Arithmetic Logic Unit) y busca combinar rendimiento y consumo de recursos.

- El NIOS II/e (“economy”) es la versión que requiere menor cantidad de recursos de la FPGA, pero también tiene limitaciones de funciones configurables por el usuario. Además carece de las operaciones de multiplicación y división.

Cada versión se complementa con diferentes componentes tales como memoria y periféricos que por medio de su interconexión a través del bus de comunicación “Avalon Switch Fabric”, se puede obtener un sistema NIOS II completo en un chip (SOC, System On Chip).

Posee las siguientes características y arquitectura:

NIOS II es un Microprocesador de 32 bits de propósito general que usa buses separados para instrucciones y datos cuyas principales características son: [5]

- Tamaño de palabra asignado de 32 bits.
- Juego de instrucciones RISC de 32 bits.
- 32 registros de propósito general de 32 bits (r0 – r31)
- 6 registros de control de 32 bits (ctl0 - ctl5)32 fuentes de interrupción externa.
- Capacidad de direccionamiento de 32 bits.
- Operaciones de multiplicación y división de 32 bits.
- Instrucciones dedicadas para multiplicaciones de 64 y 128 bits.
- Instrucciones para operaciones de coma flotante en precisión simple.

- Acceso a variedad de periféricos integrados e interfaces para manejo de memorias y periféricos.

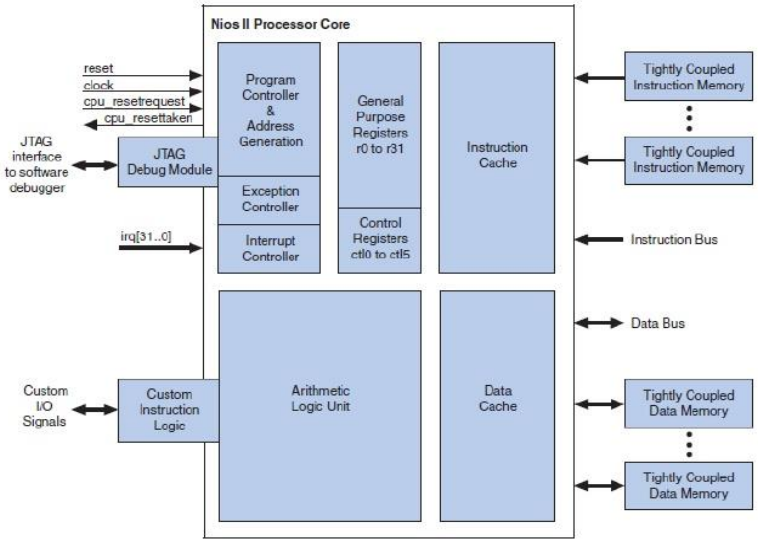


Figura 2-4 Diagrama de Bloques de la Arquitectura NIOS II [5]

2.2 PROTOCOLOS DE COMUNICACIÓN

2.2.1 PROTOCOLO SPI

Es una interfaz de periféricos serial donde se transmiten paquetes de información en una sola dirección, cuenta con tres líneas que permiten la comunicación entre dispositivos

donde cada dispositivo conectado por SPI puede actuar como transmisor y receptor al mismo tiempo, logrando una comunicación serial full dúplex. Dos de estas líneas son las encargadas de transferir los datos y la tercera es la de reloj la cual permite sincronizar la información enviada.

Los dispositivos conectado a través del protocolo SPI se comunican por medio de la relación maestro-esclavo, donde el maestro inicia la transferencia de datos generando las señales de reloj y control, mientras que el esclavo es el dispositivo controlado por el maestro que generalmente contiene una línea selectora (Chip Select o Slave Select). [6]

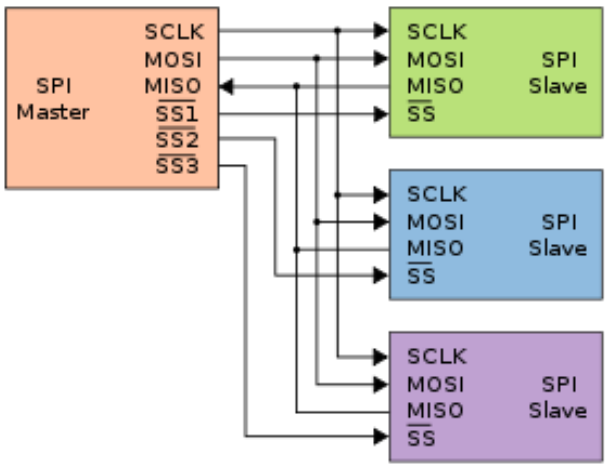


Figura 2-5 Bus SPI [6]

2.2.2 PROTOCOLO RS232

Es un estándar de comunicación serial entre dos dispositivos distantes entre sí, empleando un intercambio de datos binarios en modo serial.

Este protocolo es más utilizado en aplicaciones de bajo costo que requieren la interconexión serial entre un DTE y un DCE. Este estándar posee especificaciones físicas, mecánicas, eléctricas como se muestra a continuación:

- Mecánicas: El RS-232 consiste en un conector tipo DB-25 (de 25 pines), se lo puede encontrar también en la versión de 9 pines (DB-9). El estándar define que el conector hembra se situará en los DCE y el macho en el DTE. Cada pin puede ser de entrada o de salida, teniendo una función específica cada uno de ellos. [7]

- Eléctricas: Los estados lógicos son definidos por los siguientes niveles de voltaje: 1 lógico entre -3V y -15V, 0 lógico entre +3V y +15V. La interfaz RS-232 está diseñada para distancias cortas, de hasta 15 metros y

para velocidades de comunicación bajas, de no más de 20 Kb/s.[7]

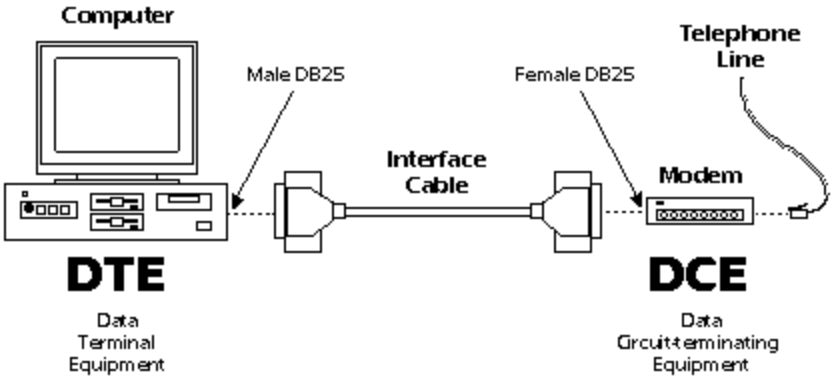


Figura 2-6 Comunicación RS232 entre dos dispositivos [8]

2.3 POTENCIÓMETRO DIGITAL MCP4132

Un potenciómetro digital es un circuito integrado cuya función es similar al de uno analógico. Se compone de un divisor resistivo de n+1 resistencias que reparte la tensión de una fuente entre una o más impedancias puramente resistivas que se encuentran conectadas en serie. Este potenciómetro contiene puntos intermedios después de cada resistencia que se encuentran conectados a un multiplexor analógico que selecciona la salida, que se manejan a través de una interfaz serial. Estos dispositivos poseen una limitación que es drenar la corriente máxima que está en el orden de los mA. [9]

Figura 2-7 Potenciómetro Digital MCP4132 [9]

2.4 CONVERTIDOR USB-RS232

Es un convertidor de puerto serial USB a puerto RS232, esto se debe a que muchas de las computadoras modernas no incluyen el puerto serial RS232, ya que se considera obsoleto para aplicaciones informáticas. Sin embargo existen muchas aplicaciones en electrónica donde resulta conveniente usar el protocolo RS232 para el intercambio de información.

Estos adaptadores cuentan con un software que al instalarlo crea un puerto serie virtual a través del puerto USB. [10]

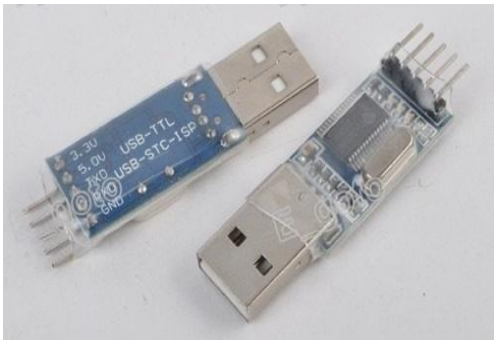


Figura 2-8 Convertidor USB-RS232 [10]

2.5 PANTALLA LCD 4X20 JHD 204A

LCD es una pantalla de cristal líquido que está integrada con el controlador y los pines de habilitación.

En total se pueden visualizar 4 líneas de 20 caracteres cada una, es decir, 80 caracteres. A pesar de que el display solo puede mostrar 20 caracteres por línea, puede almacenar en total 40 por línea. El LCD dispone de una matriz de 5x8 puntos para representar cada carácter.

En total se pueden representar 256 caracteres diferentes, estos caracteres están grabados dentro del LCD que representan las letras mayúsculas, minúsculas, signos de puntuación, números, etc.

Tiene un consumo de energía de menos de 5mA y son ideales para dispositivos que requieran una visualización detallada. [11]

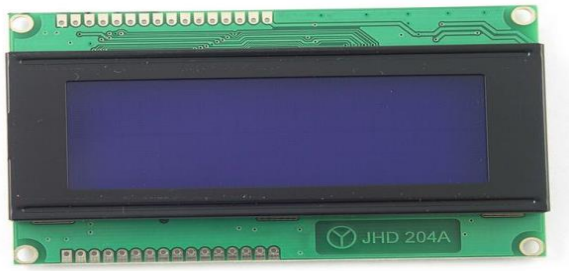


Figura 2-9 Pantalla LCD 4x20 JHD 204A [12]

2.6 TECLADO MATRICIAL 4X4

Es un dispositivo de entrada de datos que posee 16 teclas, que están interconectados en filas y columnas. Cuando se oprime un pulsador se conecta una fila con una columna.

En la Figura 2-10 se muestra el esquema de la conexión interna del teclado matricial y sus respectivos pines de salida.

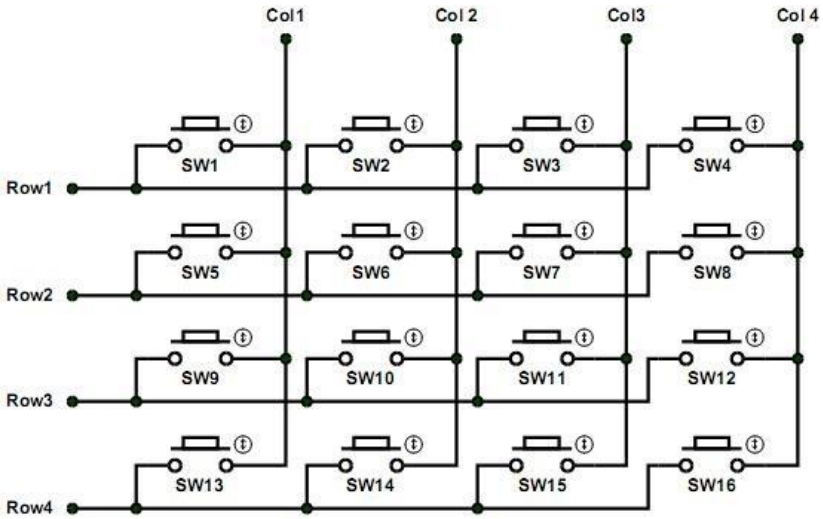


Figura 2-10 Teclado matricial 4x4 [13]

CAPÍTULO 3

3. DISEÑO E IMPLEMENTACIÓN

En este capítulo se presenta el diseño e implementación del proyecto, mostrando las diferentes etapas del Probador de Circuitos Integrados Digitales y como van interconectados entre sí.

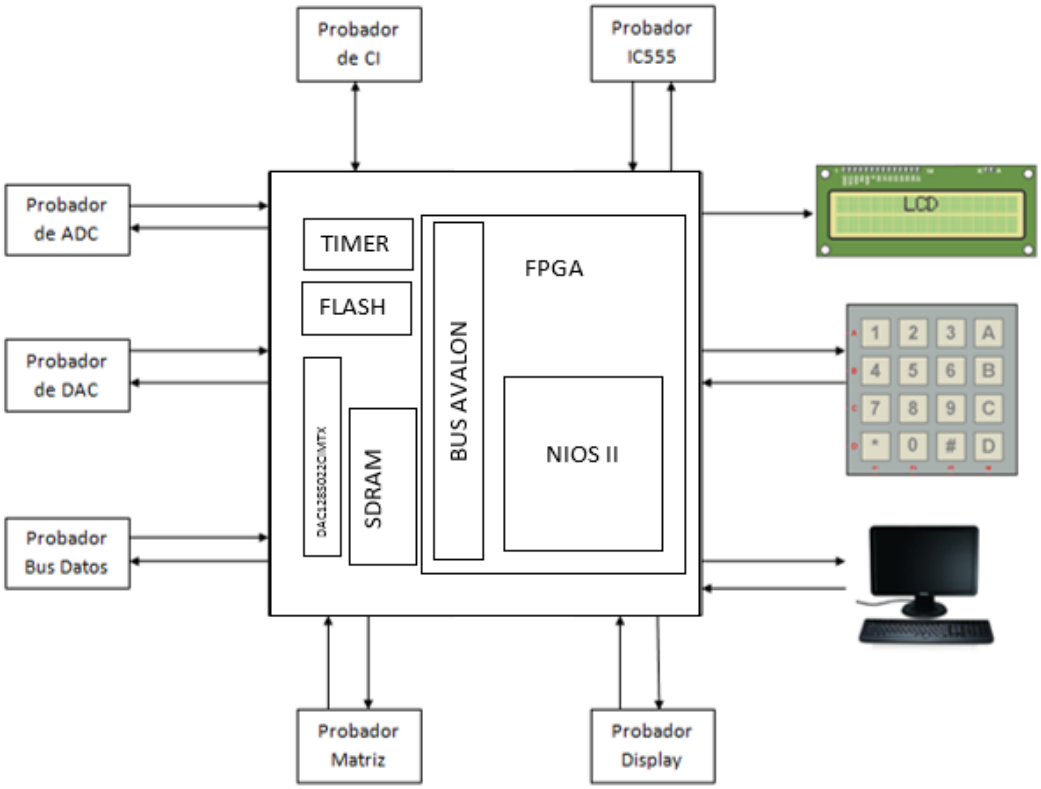


Figura 3-1 Diagrama de Bloques del sistema

En la Figura 3-1 podemos apreciar el diagrama de bloque general, es decir, los elementos que conforman el sistema.

3.1 ESQUEMÁTICOS

3.1.1 ESQUEMÁTICO PARA PROBAR CIRCUITOS INTEGRADOS DIGITALES

Para probar el estado de los CI de la familia TTL/CMOS se ha utilizado un sócalo de 24 pines (SOCKET1, SOCKET2) que está directamente conectado al puerto de expansión de la tarjeta DE0-Nano (IC_PORT), ésta envía vectores de prueba al sócalo y espera la respuesta del CI para luego ser comparado con los vectores de prueba correspondiente al número de serie del integrado que se desea verificar. Determinando así el estado del CI.

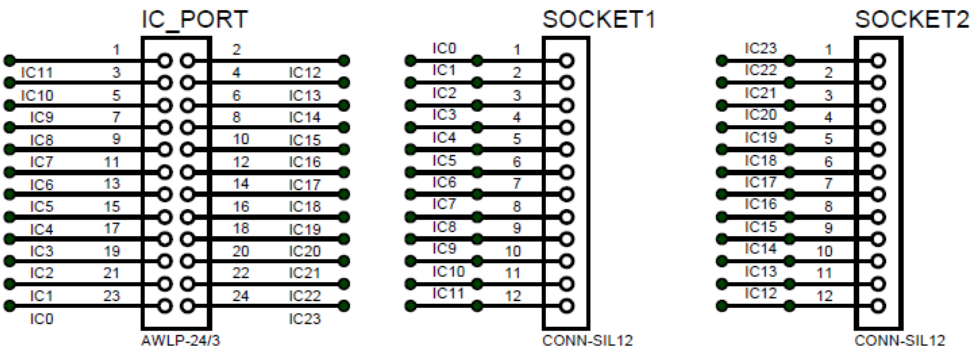


Figura 3-2 Esquemático para probar los CI

3.1.2 ESQUEMÁTICOS PARA PROBAR LOS CONVERTIDORES

3.1.2.1 CONVERTIDOR ADC

Para probar el estado del ADC se ha utilizado un potenciómetro digital MCP4132 que es activado por un comando enviado desde la FPGA, éste configura un valor de resistencia al POT para que envíe un nivel de voltaje correspondiente a dicho valor a la entrada del ADC. Los 8 bits de salida del ADC son enviados a la FPGA para comparar si la señal digital corresponde al nivel de voltaje de la entrada.

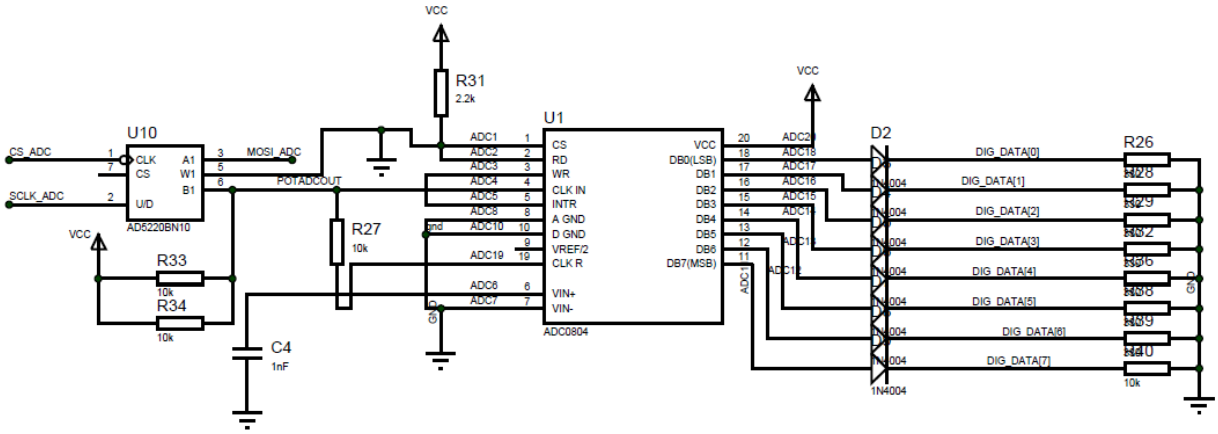


Figura 3-3 Esquemático para probar convertidor ADC

3.1.2.2 CONVERTIDOR DAC

Para probar el estado del DAC la FPGA envía una señal digital de 8 bits en paralelo al convertidor digital a analógico. Se ha utilizado un DAC con salida de corriente debido a que el DAC con salida de voltaje presenta retardo producido por la conversión interna. Por tal motivo fue necesario convertir la corriente en voltaje para esto se ha implementado un amplificador operacional externo como se muestra en la Figura 3-4. El voltaje de salida del DAC es enviado a la FPGA para ser digitalizado y luego comparado con la señal digital enviada.

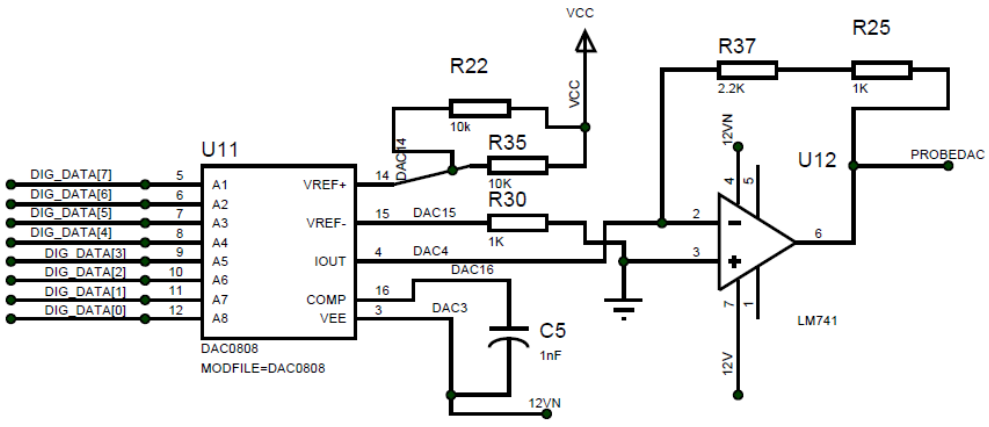


Figura 3-4 Esquemático para probar convertidor DAC

Se realizaron pruebas para configurar los intervalos de voltaje en los que debe encontrarse el DAC para verificar su estado, estas pruebas se las puede encontrar en los anexos.

3.1.3 ESQUEMÁTICO PARA PROBAR BUS DE DATOS

La FPGA a través del puerto de expansión JP1 envía señales de nivel alto “1” a un extremo del bus de datos, al otro extremo se ha colocado puertas AND cuyas salidas son enviadas a la FPGA para que verifique automáticamente el estado del bus de datos.

Las puertas AND se implementaron a partir de diodos y resistencias debido a que nuestro proyecto es fiable y escalable, es decir, que es capaz de ajustarse a nuevas tecnologías.

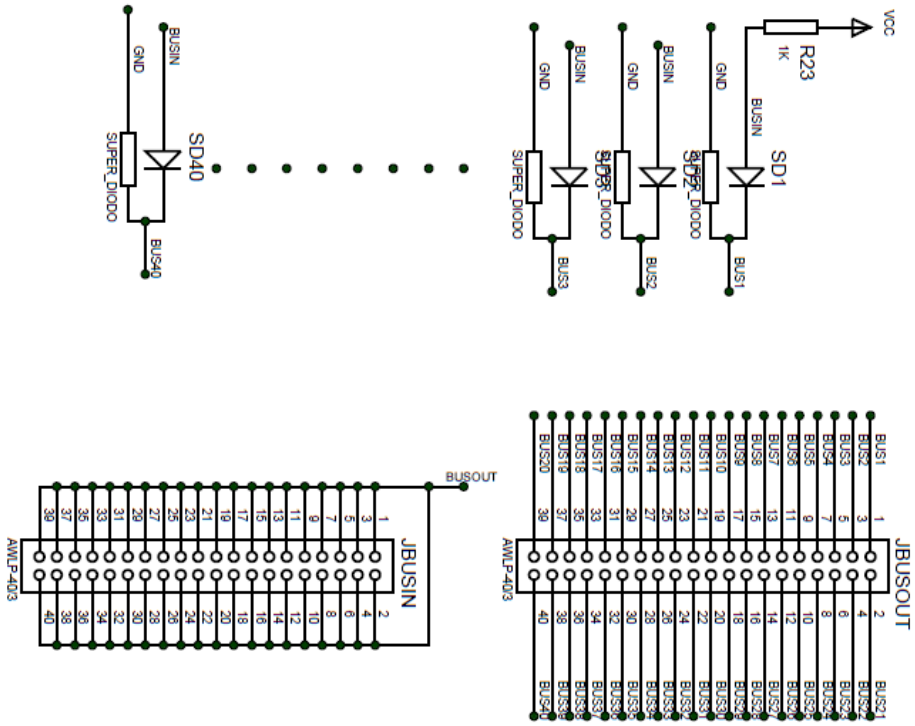


Figura 3-5 Esquemático para probar el bus de datos

3.1.4 ESQUEMÁTICO PARA PROBAR IC555

A través del protocolo SPI la FPGA por medio del puerto de expansión JP1 envía un dato serial al potenciómetro digital

MCP4132 que se configura con un valor de resistencia que está directamente conectado al pin 7 de descarga del IC555 para establecer su frecuencia de oscilación. La salida 555_3 como se muestra en la Figura 3-6 está conectada a la FPGA permitiendo verificar su funcionamiento mediante un algoritmo ya establecido, tal como se muestra en la Figura 3-24.

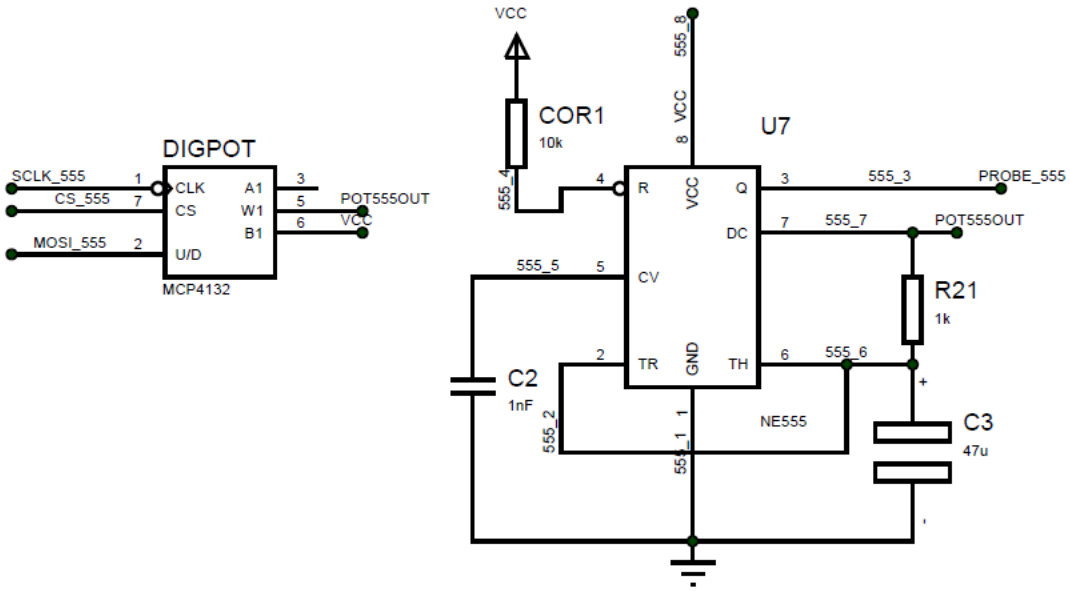


Figura 3-6 Esquemático para probar IC555

3.1.5 DISPLAY 7 SEGMENTOS

El diseño del probador de display de 7 segmentos se ha dividido en etapas como se detallan a continuación.

3.1.5.1 ETAPA DE SELECCIÓN DE VOLTAJE DEL DISPLAY 7 SEGMENTOS

La FPGA reconoce automáticamente que tipo de display se desea verificar, es decir, si es ánodo común o cátodo común.

De acuerdo a esto la FPGA envía valores (1 o 0) por medio de las señales SEG7OUT y DISPLAYTYPE que conmuta al relé seleccionando así el voltaje.

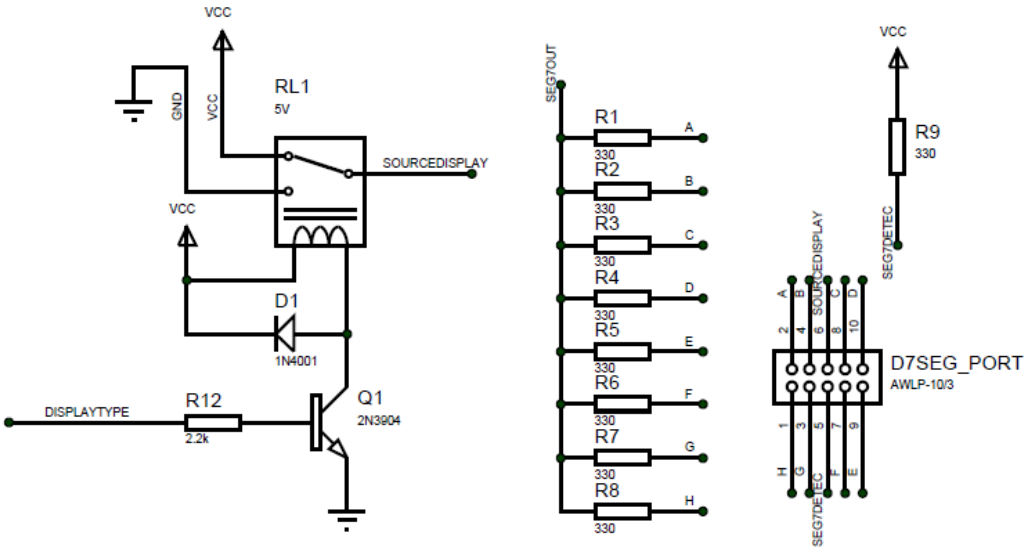


Figura 3-7 Etapa de selección de voltaje del display 7 segmentos

3.1.5.2 ETAPA DE COMPARACIÓN DE VOLTAJE DEL DISPLAY 7 SEGMENTOS

Para optimizar recursos de la tarjeta DE0-Nano, se ha implementado un comparador de voltaje externo con el integrado LM324 como se muestra en la Figura 3-8. Cada uno de los pines del display de 7 segmentos es comparado con un voltaje de referencia, para obtener dicho voltaje se realizaron pruebas a los dos tipos de display´s.

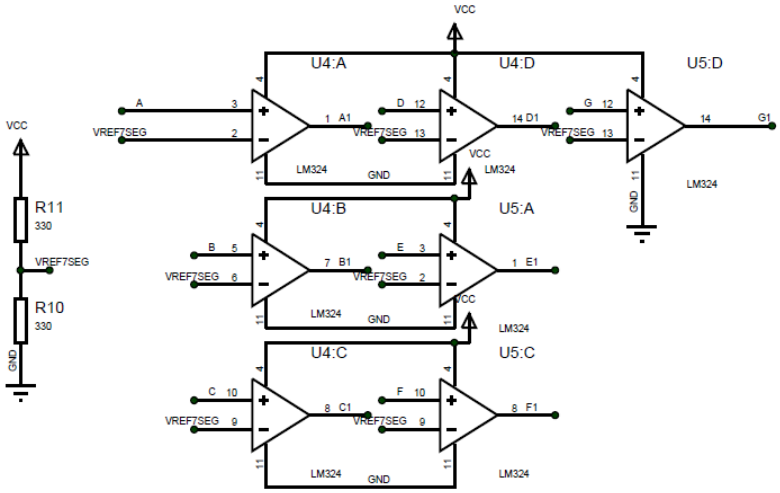


Figura 3-8 Etapa de comparación del display 7 Segmentos

Para obtener el voltaje de referencia del amplificador operacional se realizaron pruebas a display's de ánodo y cátodo común, las cuales se adjuntan en anexos.

3.1.5.3 ETAPA DE VERIFICACIÓN DEL DISPLAY 7 SEGMENTOS

A la salida del comparador del voltaje se ha colocado una puerta AND de 7 entradas, cuya salida está conectada a la FPGA que verifica automáticamente el estado de los display´s.

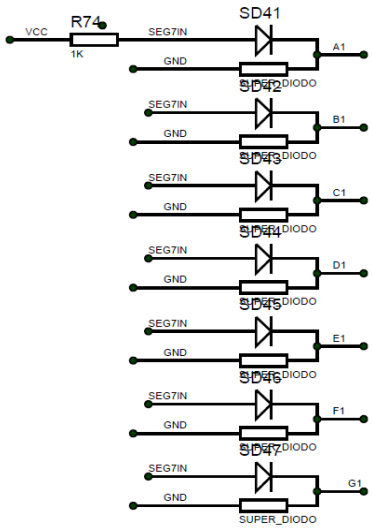


Figura 3-9 Etapa de verificación del display 7 Segmentos

3.1.6 DISPLAY MATRICIAL 7X5

El diseño del probador de display matricial 7x5 se ha dividido en etapas como se detallan a continuación.

3.1.6.1 ETAPA DE SELECCIÓN DE VOLTAJE DEL DISPLAY MATRICIAL 7X5

La FPGA reconoce automáticamente que tipo de display matricial se desea verificar, es decir, si es ánodo común o cátodo común. De acuerdo a esto la FPGA

envía una señal que conmuta al relé seleccionando así el voltaje.

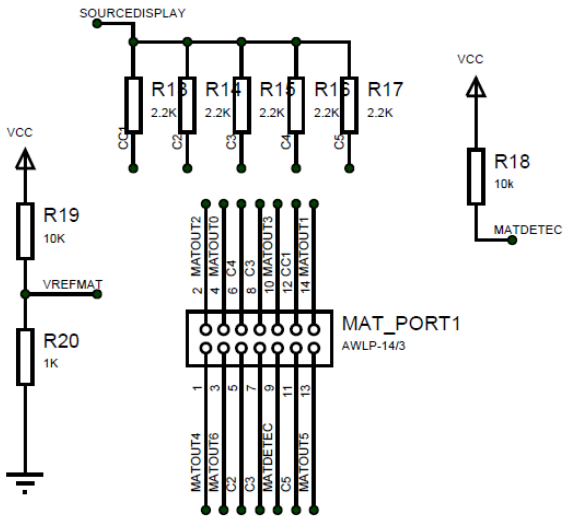


Figura 3-10 Etapa de selección de voltaje del display matricial

3.1.6.2 ETAPA DE COMPARACIÓN DE VOLTAJE DEL DISPLAY MATRICIAL 7X5

La FPGA envía a través de los puertos de expansión JP1 voltajes altos a cada fila de la matriz. Se ha implementado un comparador de voltaje con el integrado LM324 como se muestra en la Figura 3-11, éste compara el voltaje de las columnas con un voltaje de referencia de 0.4.

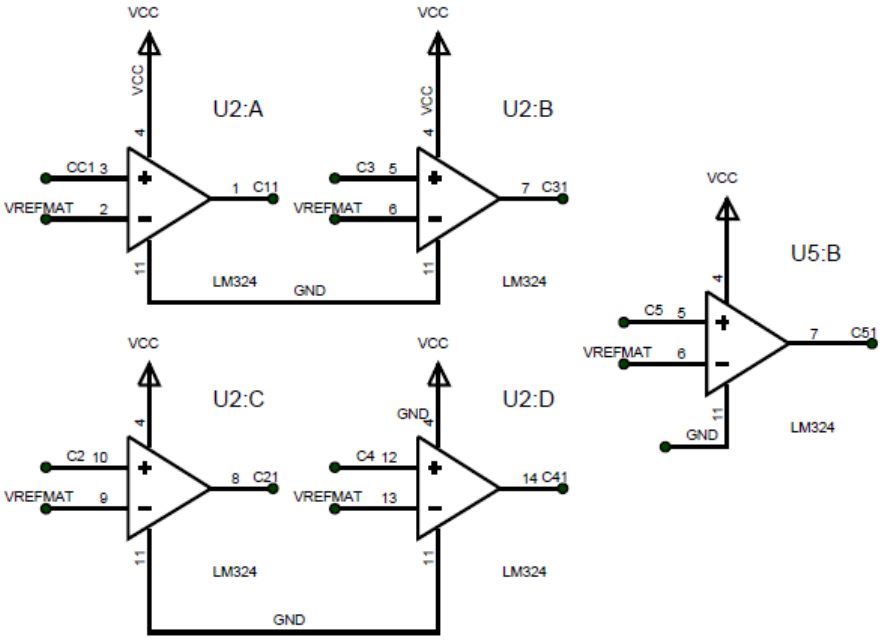


Figura 3-11 Etapa de comparación del display matricial

3.1.6.3 ETAPA DE VERIFICACIÓN DEL DISPLAY MATRICIAL 7X5

A la salida del comparador del voltaje se ha colocado una puerta AND de 5 entradas, cuya salida está conectada a la FPGA que verificar automáticamente el estado de cada fila de la matriz.

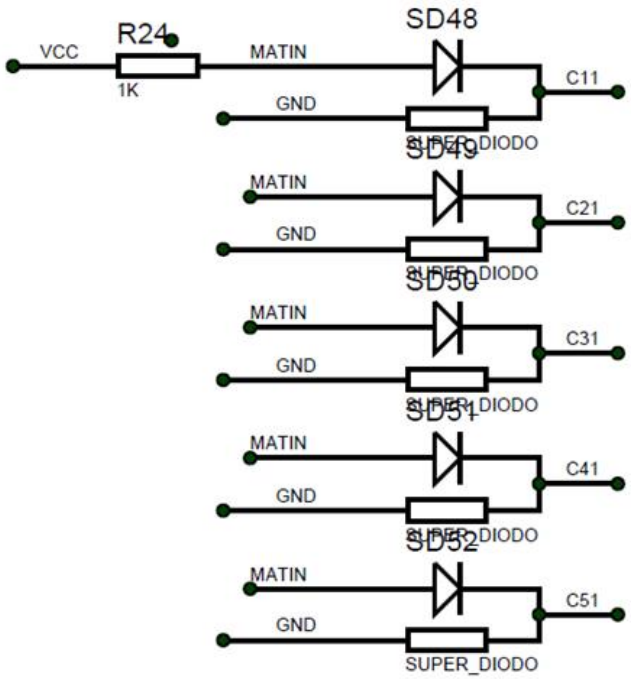


Figura 3-12 Etapa de verificación del display matricial

3.2 IMPLEMENTACIÓN DE HARDWARE EN QSYS

Para diseñar nuestro sistema basado en el Microprocesador Embebido NIOS II hemos utilizado el software Qsys, que permite generar automáticamente la lógica de interconexión, implementándolo en un solo chip.

El sistema basado en Microprocesador Embebido NIOS II es diseñado en la herramienta Qsys e implementado en la FPGA usando Quartus II.

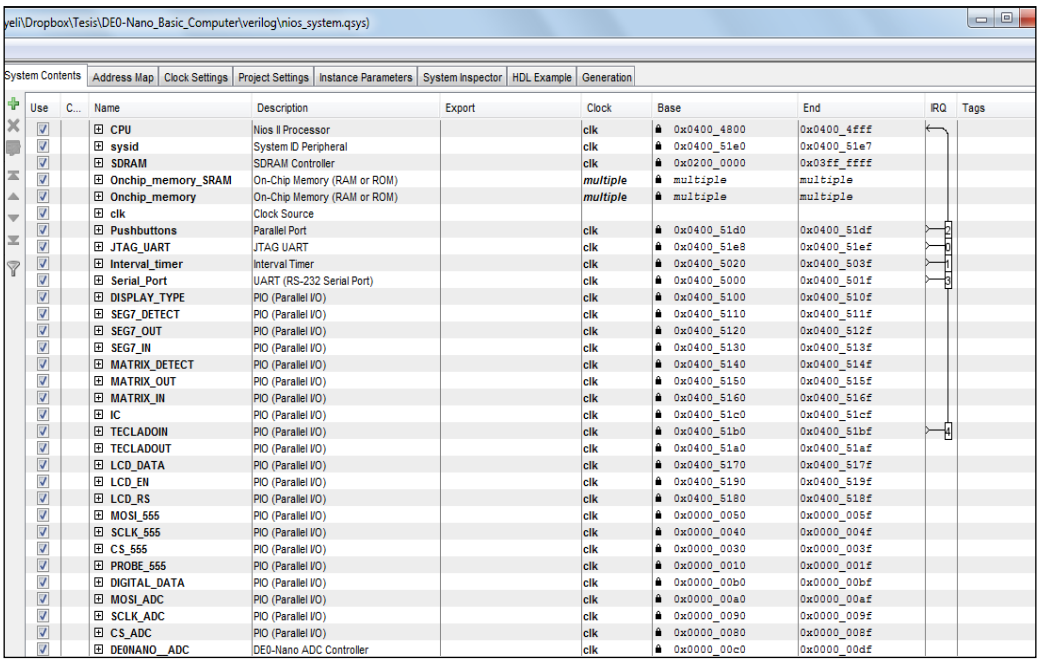


Figura 3-13 Diseño del sistema en Qsys

En la Figura 3-13 se muestra el diseño del sistema en Qsys y algunos de los componentes más importantes:

- El Procesador NIOS II que es el Procesador Central (CPU).
- Controlador de reloj.
- Puertos paralelos de E/S.
- Interfaz de protocolo SPI.
- Memoria

3.3 PROGRAMACIÓN EN LENGUAJE C USANDO SOFTWARE NIOS II

El programa se lo realizó en lenguaje C y fue compilado usando NIOS II IDE para Eclipse. En la Figura 3-14 se muestra el algoritmo del sistema.

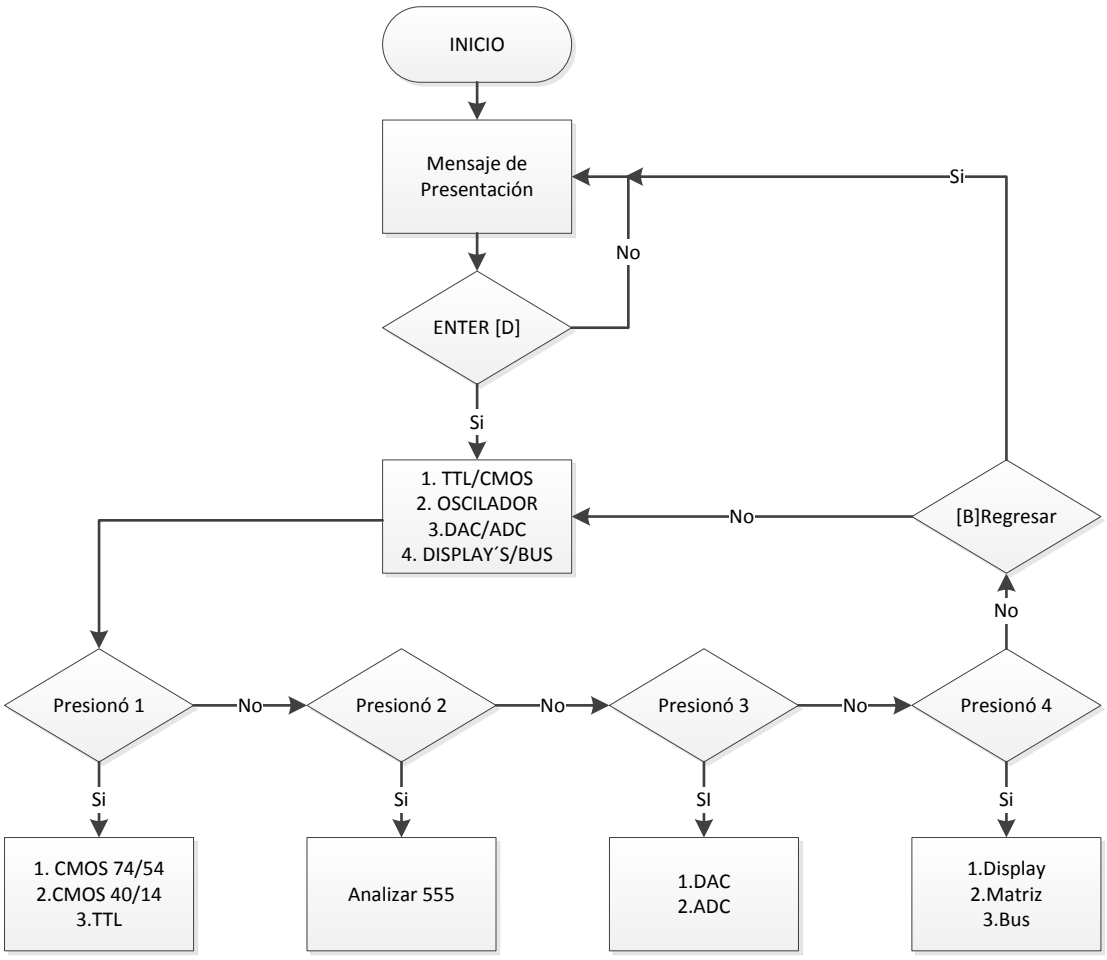


Figura 3-14 Diagrama de Flujo del programa Principal

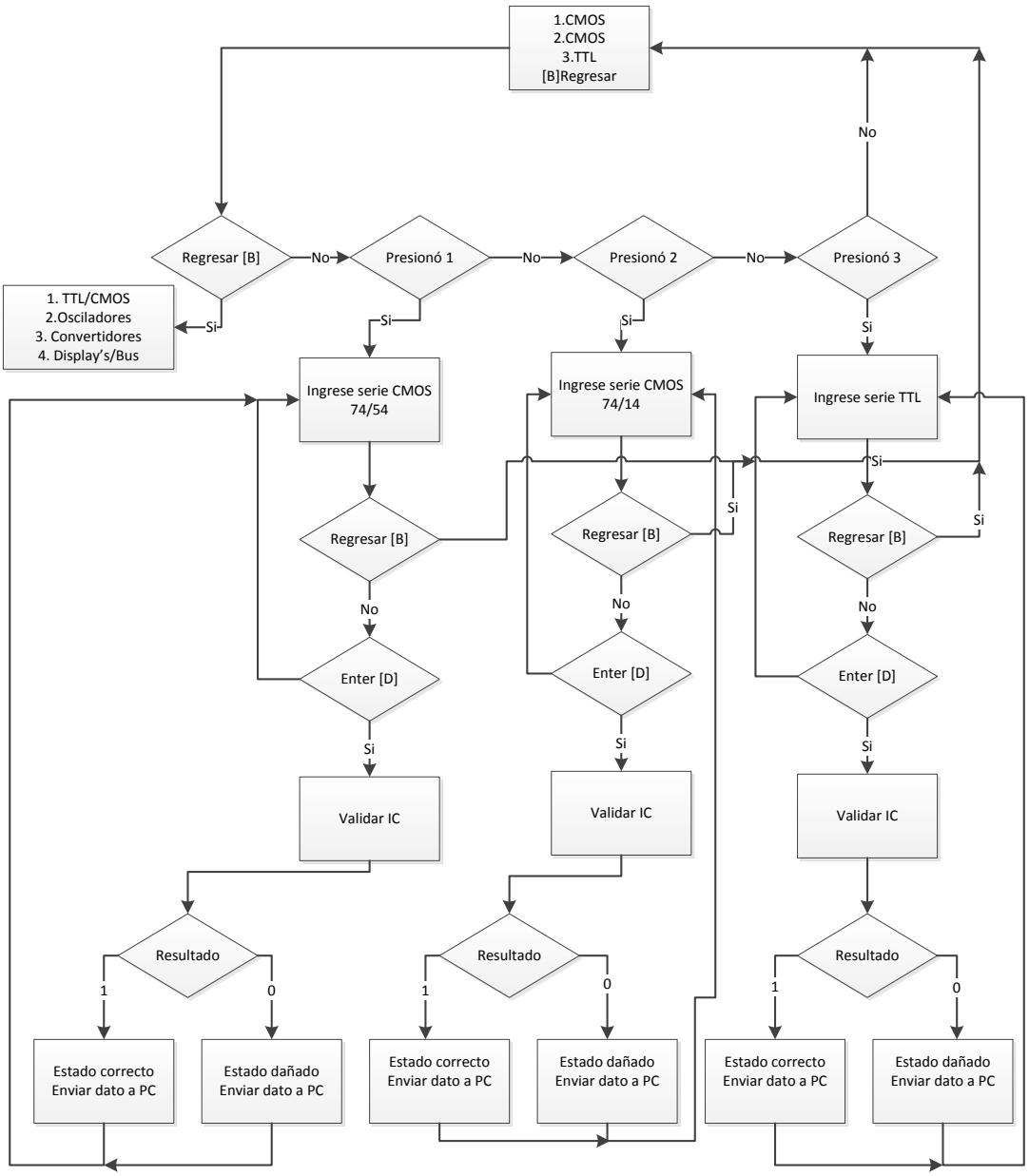


Figura 3-15 Diagrama de Flujo para prueba de CI TTL/CMOS

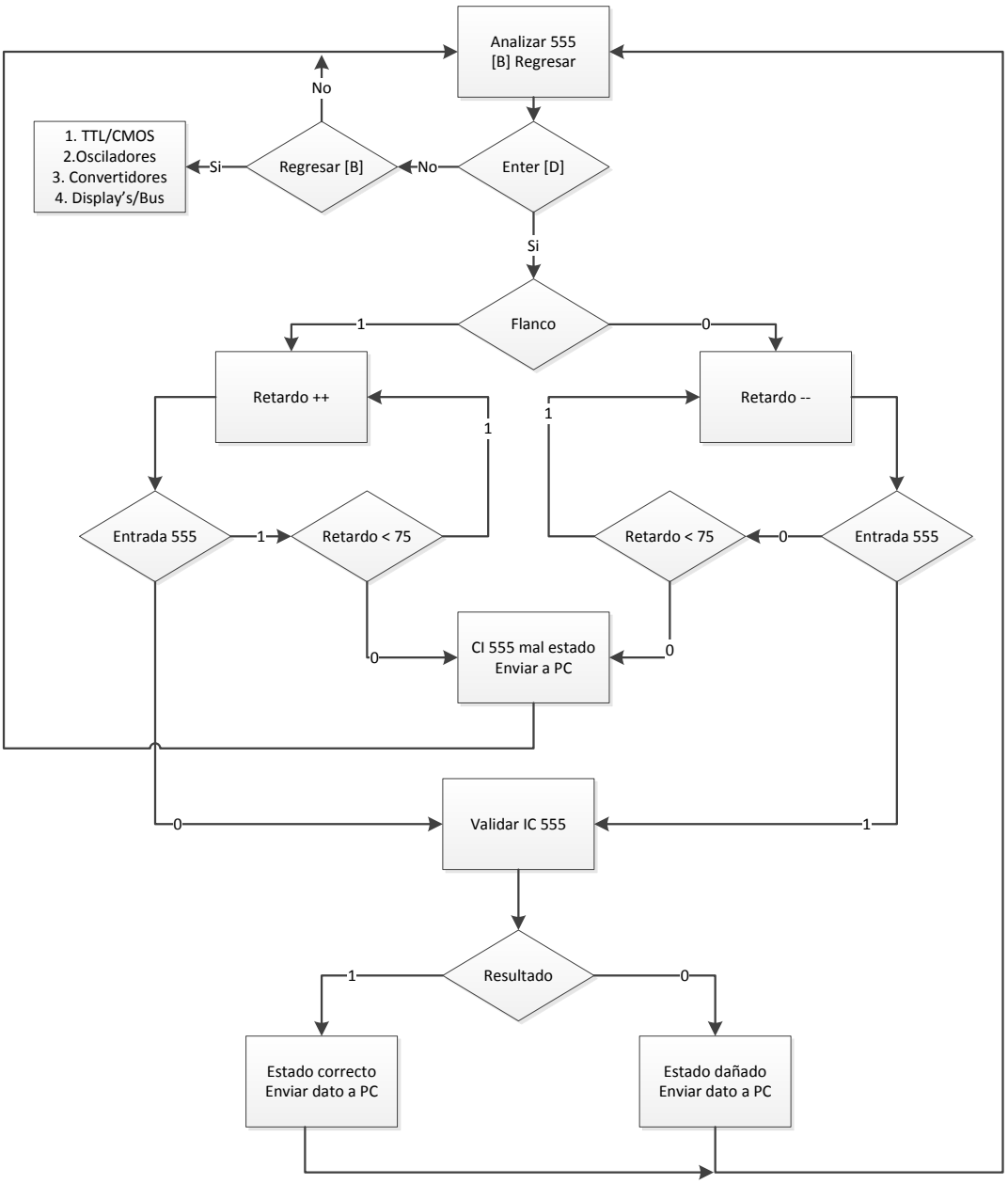


Figura 3-16 Diagrama de Flujo para prueba de CI 555

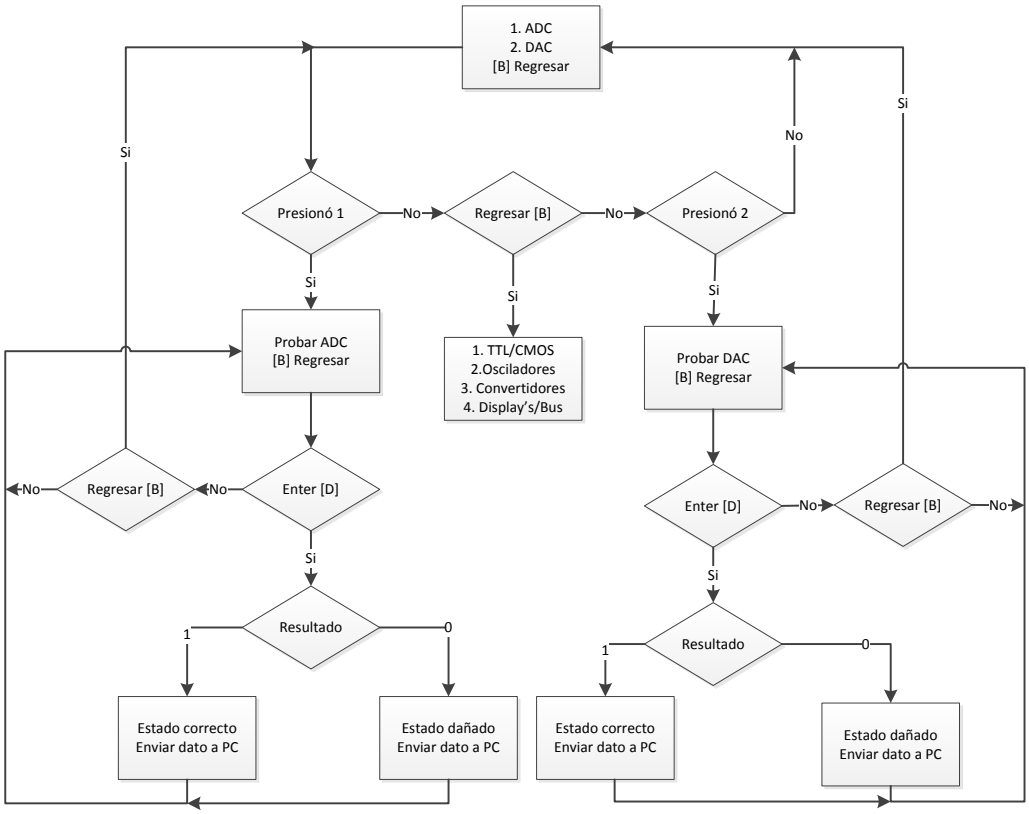


Figura 3-17 Diagrama de Flujo para prueba de convertidores

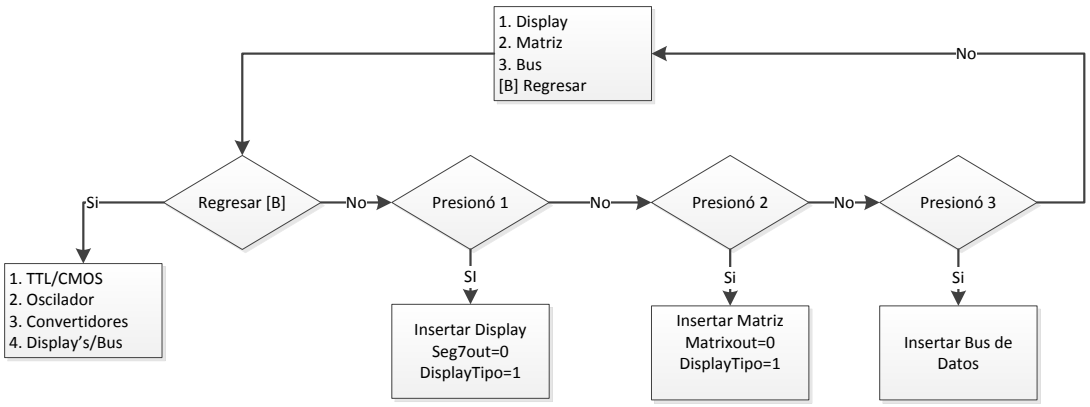


Figura 3-18 Diagrama de Flujo general para prueba de Display's/Bus

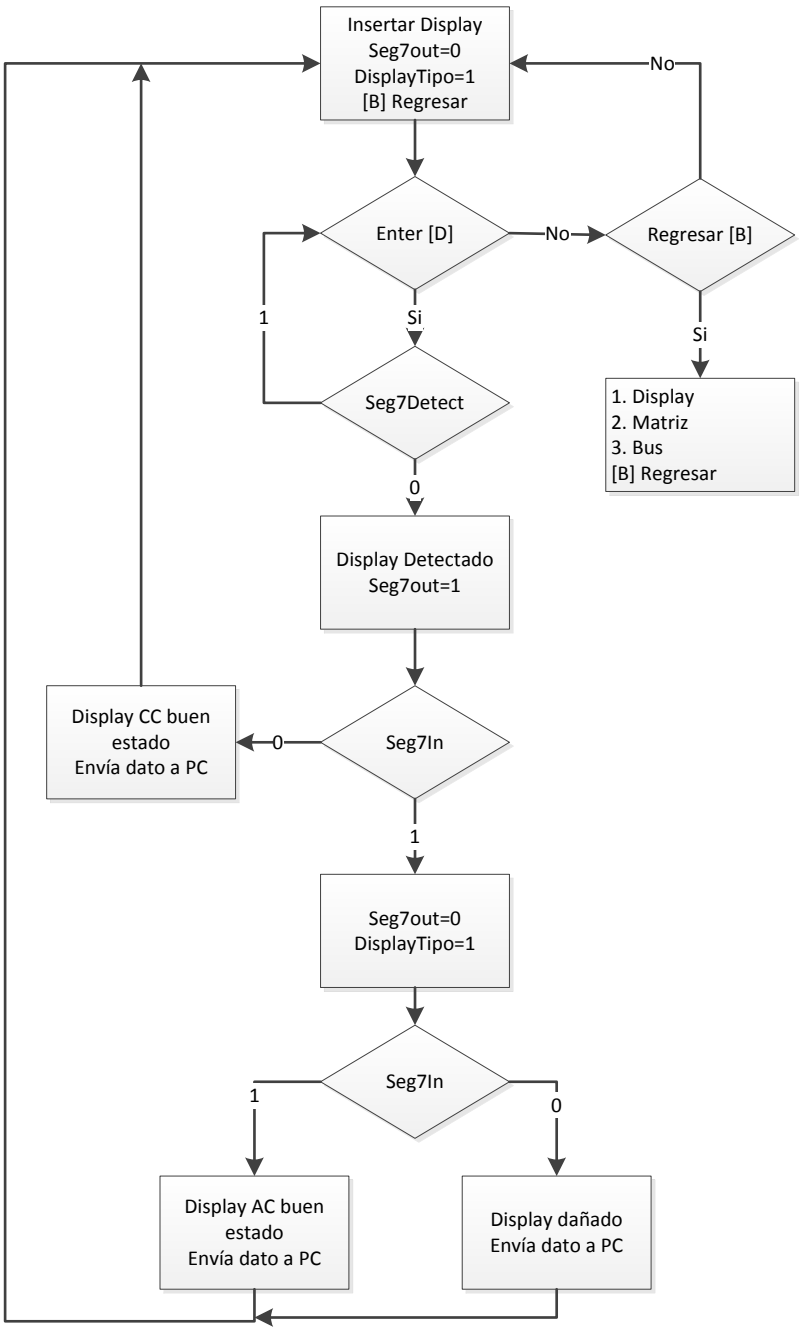


Figura 3-19 Diagrama de Flujo para prueba de Display 7 segmentos

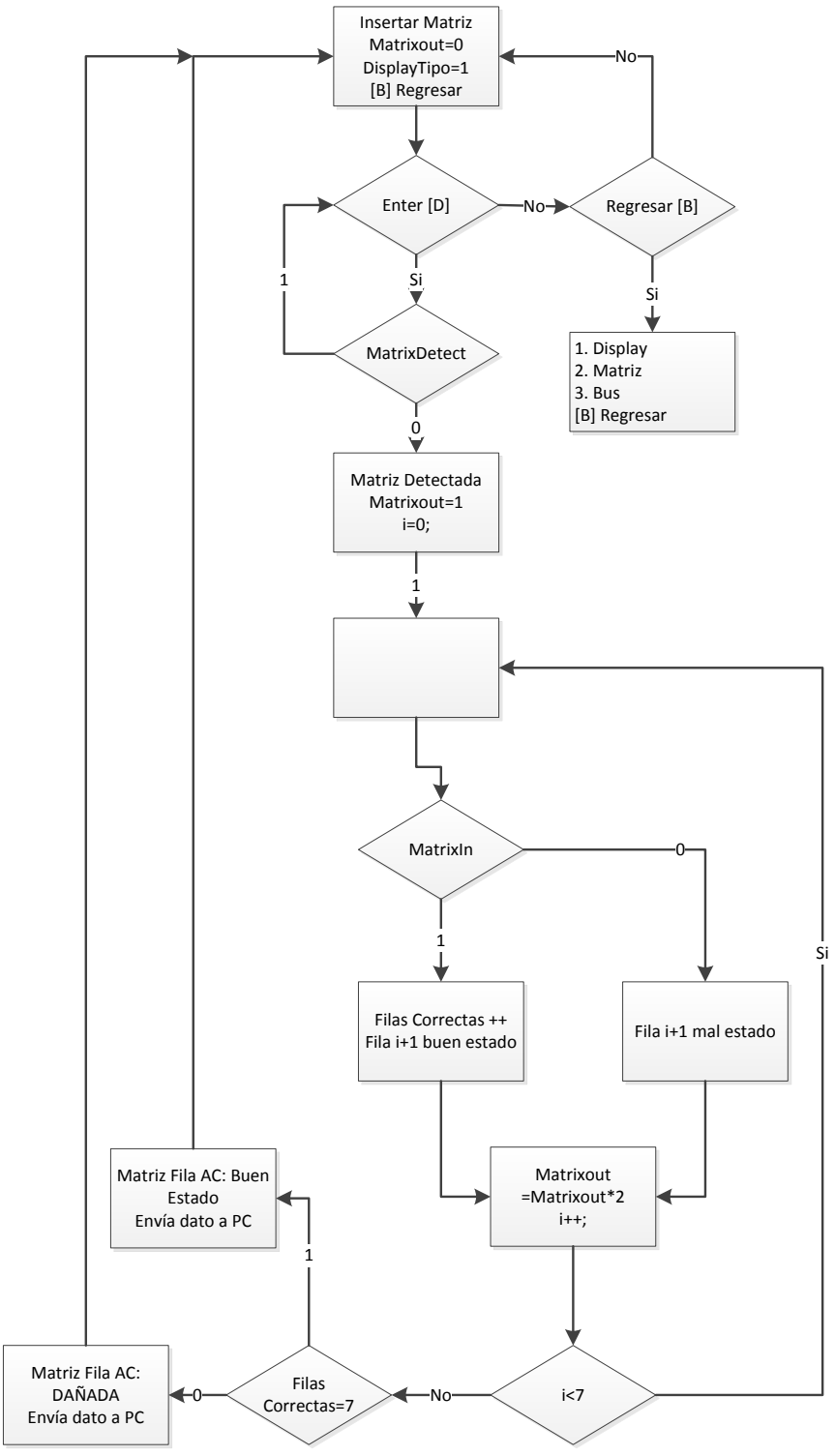


Figura 3-20 Diagrama de Flujo para prueba de Matriz 7x5

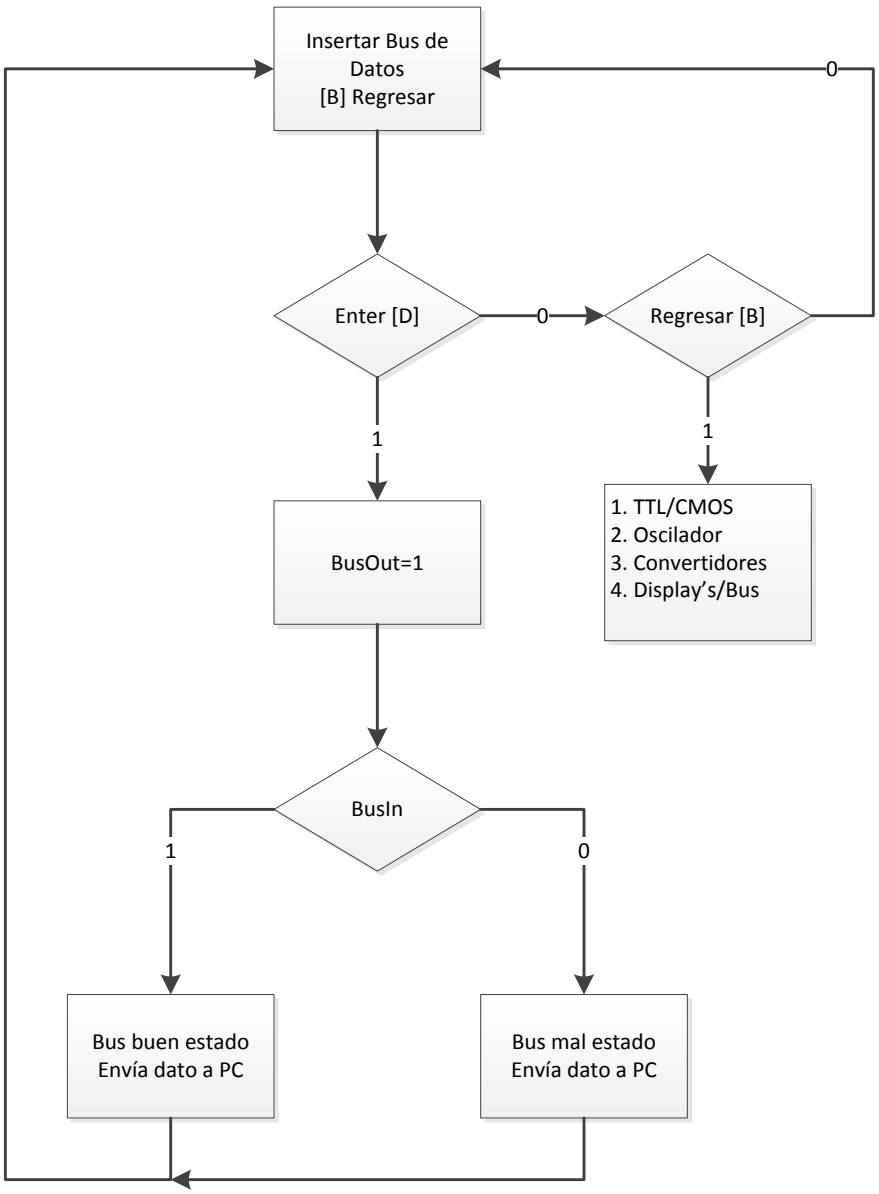


Figura 3-21 Diagrama de Flujo para prueba de Bus de Datos

En el programa se han usado varias librerías tanto del lenguaje C como del IDE de NIOS II para Eclipse que se presentan a continuación:

```
/* librerias */
#include "system.h"
#include <math.h>
#include <time.h>
#include <string.h>
#include <io.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <system.h>
#include <altera_up_avalon_parallel_port.h>
#include "altera_up_avalon_parallel_port_regs.h"
#include "paneles.h"
```

La librería ***stdlib.h*** es una librería estándar de propósito general de lenguaje C, que contiene los prototipos de funciones para gestión de memoria dinámica y control de procesos. La librería ***stdio.h*** contiene las definiciones de macros, constantes, y declaraciones de funciones.

```
void ProbarDisplay(){
    *Seg7Out = 0;
    *DisplayTipo = 1;
    while(*Seg7Detect!=0);
    LCD_Clear();
    LCD_Write("VERIFICANDO DISPLAY ",20,1);
    LCD_Write("--------------------",20,2);
    usleep(1500000);
    *Seg7Out = 1;
    usleep(1000);
    if(*Seg7In == 0){
        LCD_Write("DISPLAY CATODO COMUN",20,3);
        LCD_Write("     BUEN ESTADO    ",20,4);
    }
    else{
        *Seg7Out = 0;
        *DisplayTipo = 0;
        usleep(1000000);
        if(*Seg7In== 1){
            LCD_Write("DISPLAY ANODO COMUN ",20,3);
            LCD_Write("     BUEN ESTADO    ",20,4);
        }
        else
            LCD_Write("  DISPLAY: DANADO   ",20,4);
    }
    *DisplayTipo = 1;
    usleep(1000000);
    *Seg7Out = 0;
    LCD_Clear();
    LCD_Write("   RETIRE DISPLAY   ",20,2);
    while(*Seg7Detect==0);
    usleep(1000000);
}
```

Figura 3-22 Función para probar display 7 segmentos

En la Figura 3-22 se muestra la función ***ProbarDisplay()***, que se encarga de reconocer el tipo de display 7 segmentos que se desea probar, es decir, si es ánodo o cátodo común. Además verifica el

estado en el que se encuentra el componente y lo muestra en pantalla.

```
void ProbarMatriz(){
    int Filas_Correctas = 0, i=0;
    LCD_Clear();
    *MatrixOut = 0;
    *DisplayTipo = 1;
    while(*MatrixDetect!=0);
    usleep(1000000);
    LCD_Clear();
    LCD_Write("VERIFICANDO MATRIZ  ",20,1);
    LCD_Write("--------------------",20,2);
    *MatrixOut = 1;
    for(i=0;i<7;i++){
        usleep(1000);
        if(*MatrixIn == 1)
            Filas_Correctas++;//Fila %d en buen estado
        usleep(62500);
        *MatrixOut = (*MatrixOut)*2;
    }
        if(Filas_Correctas==7){
            LCD_Write("   MATRIZ FILA-AC:  ",20,3);
            LCD_Write("    BUEN ESTADO     ",20,4);
        }
        else
            LCD_Write("   MATRIZ DANADA    ",20,3);
    *DisplayTipo = 1;
    *MatrixOut = 127;
    usleep(1000000);
    *MatrixOut = 0;
    LCD_Clear();
    LCD_Write("   RETIRE MATRIZ    ",20,3);
    while(*MatrixDetect==0);
    usleep(1000000);
}
```

Figura 3-23 Función para probar display matricial

En la Figura 3-23 se muestra las líneas de código de la función ProbarMatriz(), que permite verificar y visualizar en la LCD el estado de la matriz fila ánodo común.

```
void Probar555(){
    LCD_Clear();
    int val1, val2, val3;
    Send16SPI(ALTO, CS_555, Sclk_555, Dout_555);
    usleep(10000);
    val1=Rate_555();
    LCD_Write("[T1 <0.8] s   : ",13,1);
    Mostrar555(val1);
    Send16SPI(MEDIO, CS_555, Sclk_555, Dout_555);
    usleep(10000);
    val2=Rate_555();
    LCD_Write("[2.11<T2<2.16]s:",13,2);
    Mostrar555(val2);
    Send16SPI(BAJO, CS_555, Sclk_555, Dout_555);
    usleep(10000);
    val3=Rate_555();
    LCD_Write("[T3 >3.58] s:   ",13,3);
    Mostrar555(val3);
    if(val1<1000000 && (val2>2000000 && val2<3000000) && val3>3000000){
        LCD_Write("CI-555 : BUEN ESTADO",20,4);
        serial("LM&LM555%OK");
    }
    else{
        LCD_Write("CI-555 : MAL ESTADO ",20,4);
        serial("LM&LM555%FAIL");
    }
    usleep(1000000);
}
```

Figura 3-24 Función para probar CI 555

En la Figura 3-24 se observa la función para probar el oscilador 555. Este segmento de código es capaz de reconocer el estado del integrado 555 mediante los tres valores de tiempo que le toma al CI en dar 10 flancos de reloj obtenidos con la función ***Rate_555***.

```
/* Funcion para medir el tiempo que demora en dar 10 flancos(555)*/
int Rate_555(){
    int TIME=0, CLOCKS=0;
    long HIGH, LOW, COUNTER;
    // Setear Contador en H:L (FFFF:FFFF)
    IOWR(INTERVAL_TIMER_BASE,2,0xFFFF);
    IOWR(INTERVAL_TIMER_BASE,3,0xFFFF);
    while(*Entrada_555==1);
    while(*Entrada_555==0);
    IOWR(INTERVAL_TIMER_BASE,1,4);//START
    for(CLOCKS=0 ; CLOCKS<10 ; CLOCKS++){
        while(*Entrada_555==1);
        while(*Entrada_555==0);
    }
    IOWR(INTERVAL_TIMER_BASE,1,8);//STOP
    IOWR(INTERVAL_TIMER_BASE,4,0);//CAPTURA PARATE BAJA
    IOWR(INTERVAL_TIMER_BASE,5,0);//CAPTURA PARATE ALTA
    LOW=0xFFFF & IORD(INTERVAL_TIMER_BASE,4);
    HIGH=0xFFFF & IORD(INTERVAL_TIMER_BASE,5);
    COUNTER=0xFFFFFFFF & (LOW + (HIGH<<16));
    TIME=((0xFFFFFFFF-COUNTER)/50);
    return TIME;
}
```

Figura 3-25 Función para determinar tiempo del 555 en dar 10 flancos de reloj

En la Figura 3-15 se muestra la función ***Rate_555(),*** que permite determinar el tiempo que se demora el CI 555 en capturar 10 flancos de reloj.

```
void ProbarADC0804(){
    int Dato1, Dato2, Dato3;
    IOWR_ALT_UP_PARALLEL_PORT_DIRECTION(DIGITAL_DATA_BASE, 0x00);
    Send16SPI(ALTO, CS_ADC, Sclk_ADC, Dout_ADC);
    usleep(100000);
    Dato1 = *Dato_Digital;
    LCD_Clear();
    LCD_Write("D1 [7-10]:  ",12,1);
    binario(Dato1);
    Send16SPI(MEDIO, CS_ADC, Sclk_ADC, Dout_ADC);
    usleep(100000);
    Dato2 = *Dato_Digital;
    LCD_Write("D2[126-129]:",12,2);
    binario(Dato2);
    Send16SPI(BAJO, CS_ADC, Sclk_ADC, Dout_ADC);
    usleep(100000);
    Dato3 = *Dato_Digital;
    LCD_Write("D3[168-171]:",12,3);
    binario(Dato3);
    if((Dato1>6 && Dato1<11) && (Dato2>125 && Dato2<130) && (Dato3>167 && Dato3<172)){
        LCD_Write("ADC: BUEN ESTADO    ",20,4);
        serial("ADC/DAC&ADC808%OK");
    }
    else{
        LCD_Write("ADC: MAL ESTADO     ",20,4);
        serial("ADC/DAC&ADC808%FAIL");

    }
}
```

Figura 3-26 Función para probar ADC0804

En la Figura 3-26 se muestra las líneas de código de la función para probar el convertidor ADC0804 el cual es capaz de reconocer el estado en el que se encuentra este elemento mediante 3 rangos de referencia escogidos para compararlos con valores obtenidos de la función ***Send16SPI*** .

```
void ProbarDAC0808(){
    float Dato1, Dato2, Dato3;
    IOWR_ALT_UP_PARALLEL_PORT_DIRECTION(DIGITAL_DATA_BASE, 0xFF);
    *Dato_Digital=0x00;
    usleep(1000000);
    *(ADC_DE0nano)=0;
    Dato1=(*(ADC_DE0nano)*3.33)/4096;
    LCD_Clear();
    LCD_Write("[D1<0.05]V    : ",16,1);
    MostrarDAC(Dato1);
    getchar();
    *Dato_Digital=0x80;
    usleep(1000000);
    *(ADC_DE0nano)=0;
    Dato2=(*(ADC_DE0nano)*3.3)/4096;
    LCD_Write("D2[1.55-1.64]V: ",16,2);
    MostrarDAC(Dato2);
    getchar();
    *Dato_Digital=0xFF;
    usleep(1000000);
    *(ADC_DE0nano)=0;
    Dato3=(*(ADC_DE0nano)*3.3)/4096;
    LCD_Write("[D3>3.20]V    : ",16,3);
    MostrarDAC(Dato3);
    getchar();
    if(Dato1<0.05 && (Dato2>1.55 && Dato2<1.64) && Dato3>3.20){
        LCD_Write("DAC: BUEN ESTADO    ",20,4);
    }
    else
        LCD_Write("DAC: MAL ESTADO     ",20,4);
        serial("ADC/DAC&ADC808%FAIL");
}
```

Figura 3-27 Función para probar DAC0808

En la figura 3-27 se puede observar en las líneas de código de la función ***ProbarDAC0808()***, que la FPGA envía 3 valores diferentes a la entrada del DAC cuyo valor de salida es leído por el canal 0 de la DE0-Nano, para ser comparados con rangos específicos que permiten determinar su estado.

```
void ProbarBUS(){
     int c;
     LCD_Write("  VERIFICANDO BUS   ",20,1);
     LCD_Write("--------------------",20,2);
        *BusOut =1;
        usleep(100000);
        c=*BusIn;
    if( *BusIn==1 ){
        LCD_Write("  BUS DE 40 PINES   ",20,3);
        LCD_Write("     BUEN ESTADO    ",20,4);
    }
    else{
        LCD_Write("     BUS DE DATOS   ",20,3);
        LCD_Write("      MAL ESTADO    ",20,4);

    }


}

void LCD_Init(){
    ProcesarComando(0x3C);
    ProcesarComando(0x0C);
    ProcesarComando(0x06);
}
```

Figura 3-28 Función para probar Bus de Datos

La Figura 3-28 muestra el código de la función ***ProbarBUS()***, utilizada para enviar “1” por medio de la variable ***BusOut*** al bus de datos, recibir la respuesta a través de ***BusIn*** para ser comparada y poder determinar su estado.

CAPÍTULO 4

4. PRUEBAS Y RESULTADOS

En este capítulo se muestran las pruebas realizadas a los diferentes componentes para verificar el estado y poder visualizarlas en una pantalla LCD. Además se realiza una comparación entre valores adquiridos en pruebas con instrumentos de medición y el probador de componentes.

Para mostrar el buen funcionamiento del proyecto se han planteado 4 escenarios que se detallan a continuación:

4.1 ESCENARIOS

- Escenario A: Prueba de circuitos integrados digitales.
- Escenario B: Prueba de Displays.
- Escenario C: Prueba de convertidores.
- Escenario D: Prueba de bus de datos.

4.1.1 ESCENARIO A: PRUEBA DE CIRCUITOS INTEGRADOS

Al inicio se muestra un mensaje inicial del probador, luego se despliega el menú principal que contiene los elementos que se pueden probar, como se muestra en la Figura 4-1.

Figura 4-1 Menú del Probador

4.1.1.1 CIRCUITOS INTEGRADOS TTL/CMOS

Al presionar la tecla “1” se muestra los tres diferentes tipos de familia de circuitos integrados que se desean probar ya sea CMOS7454, CMOS1440 ó TTL. Cuando se haya seleccionado el tipo de integrado se debe ingresar la serie y luego presionar la tecla “D” para proceder a verificar el estado del componente mediante vectores de prueba, tal como se muestra en la Figura 4-2.

Figura 4-2 Verificación del estado del integrado

4.1.1.2 CIRCUITO INTEGRADO IC 555

Para realizar la prueba del oscilador se debe presionar la tecla “2” y a su vez la tecla “D” para iniciar la prueba. Para verificar el estado del oscilador se realiza 3

pruebas internas, que consiste en determinar el tiempo que se demora en dar 10 flancos de reloj. En la primera, segunda y tercera prueba de tiempo debe encontrarse en el rango que se muestra en la figura 4-2. Si uno de estos tres valores no coincide con el rango especificado el IC 555 se diagnostica dañado.

Figura 4-3 Verificación del estado del IC555

Este rango de tiempos tomados de referencia se obtuvo mediante pruebas realizadas al IC 555 con tres valores de resistencias diferentes para variar su frecuencia, estas pruebas se encuentran adjuntas en anexos.

4.1.2 ESCENARIO C: PRUEBA DE CONVERTIDORES

4.1.2.1 CONVERTIDOR ADC

Al presionar la tecla “3” del menú principal se inicia la prueba de los convertidores, enseguida muestra el tipo de convertidor que se desea probar en este caso se presiona “1” y a su vez la tecla “D” para iniciar.

Al realizar la prueba se muestran 3 rangos de valores que se han escogido como referencia de acuerdo al valor de voltaje que se obtuvo al configurar el potenciómetro digital dirigido a la entrada del ADC, si la salida binaria del convertidor se encuentra dentro del rango establecido el componente se encuentra en buen estado como se muestra en la Figura4-4.

Figura 4-4 Prueba del convertidor ADC

Para establecer el rango de referencia se realizaron varias pruebas de lectura a la salida del ADC con

distintos valores de resistencia a la entrada. Estas pruebas se encuentran adjuntas en anexos.

4.1.2.2 CONVERTIDOR DAC

Al presionar la tecla “2” del menú de convertidores se muestra la opción de probar DAC que se inicia al presionar la tecla “D”.

Al realizar la prueba se muestran 3 rangos de valores que se han escogido como referencia de acuerdo al voltaje obtenido a la salida del DAC que varía de acuerdo a los datos binarios enviados a la entrada. Si estos valores de voltaje se encuentran dentro del rango establecido el componente se encuentra en buen estado como se muestra en la Figura 4-5.

Figura 4-5 Prueba del convertidor DAC

Para establecer el rango de referencia se realizaron varias pruebas de lectura a la salida del DAC que se encuentran adjuntas en anexos.

4.1.3 ESCENARIO C: PRUEBA DE DISPLAYS

4.1.3.1 DISPLAY 7 SEGMENTOS

Para realizar la prueba del display 7 segmentos se debe presionar la tecla “4” en el menú principal, enseguida se muestran 3 opciones de elementos display, matriz y bus. Cuando se haya seleccionado la opción del display de 7 segmentos se debe presionar la tecla “D” para iniciar la prueba, inmediatamente se verifica el estado y el tipo de display, es decir, si es ánodo común o cátodo común como se muestra en la figura 4-6.

Figura 4-6 Prueba del Display 7 segmentos

Despues de mostrar el resultado se pide retirar el display para continuar realizando pruebas.

4.1.3.2 DISPLAY MATRICIAL 7X5

Para determinar el estado de una matriz de 7x5, se debe escoger la opción “4 DISPLAY/BUS” del menú principal y a su vez la opción “2 MATRTIZ 7X5”, una vez presionado “D” se inicia la prueba. Luego se muestra el estado de cada fila y el tipo de matriz que se está probando tal como se muestra en la Figura 4-7.

Figura 4-7 Prueba del Display matricial 7x5

Una vez terminada la prueba se debe retirar la matriz para continuar realizando más pruebas.

4.1.4 ESCENARIO D: PRUEBA DE BUS DE DATOS

Al presionar la tecla “4” del menú principal y a su vez la tecla “3” que es la opción para probar bus de datos se debe presionar la tecla “D” para iniciar.

El estado del bus de datos se puede visualizar en pantalla como se muestra en la Figura 4-7.

Figura 4-8 Prueba de Bus de Datos

4.2 COMPARACIÓN ENTRE LECTURAS MANUALES Y AUTOMATIZADAS

Para determinar la eficiencia del probador de elementos se ha realizado una comparación entre valores obtenidos por medio de instrumentos de medición y la mini-computadora desarrollada en NIOS II.

INSTRUMENTOS DE MEDICIÓN	MEDICIONES			ERROR[%]		
	DAC0808					
	Lectura1 [V]	Lectura2 [V]	Lectura3 [V]	Lectura1 [V]	Lectura2 [V]	Lectura3 [V]
NIOS II	0	1,6	3,25	0	1,23	0,61
MULTIMETRO	0	1,62	3,27			
	ADC0804					
	Lectura1	Lectura2	Lectura3	Lectura1	Lectura2	Lectura3
NIOS II	1001	10000000	10101001	11,11	0,78	0,59
PUNTA DE PRUEBA	100∅	1000000∅	1010100∅			
	CI 555					
	Tiempo1 [us]	Tiempo2 [us]	Tiempo3 [us]	Tiempo1 [us]	Tiempo2 [us]	Tiempo3 [us]
NIOS II	779890	2133765	3602529	0,03	0,04	0,03
OSCILOSCOPIO	780159	2134569	3603695			

Tabla II Mediciones manuales y automatizadas

En la Tabla II podemos observar los valores obtenidos al realizar una prueba manual por medio de instrumentos de medición como multímetro, punta de prueba y osciloscopio, así como también valores determinados en NIOS II.

Para calcular el porcentaje de error se consideró como dato teórico el valor obtenido en los instrumentos de medición, los resultados obtenidos nos muestran la mínima diferencia que existe entre las mediciones manuales y automatizadas, se esperaría un porcentaje de error del 0% pero al existir circuitos integrados, resistencias y capacitores que poseen errores de fábrica, así como también instrumentos de medición que pueden encontrarse no calibrados, esta mínima diferencia de valores nos demuestra que el probador de elementos es eficiente al realizar las pruebas para determinar el estado de los componentes.

CONCLUSIONES

1. El probador de elementos implementado cumple con los objetivos planteados ya se logró integrar el módulo de probador de integrados TTL/CMOS con total éxito, además se realizó satisfactoriamente la automatización de pruebas de los componentes propuestos (matriz 7x5, display 7 segmentos, DAC0808, ADC0804, oscilador 555 y Bus de datos de 40 pines).

2. Entre las opciones de lenguaje de programación que podemos utilizar en NIOS II para el desarrollo del proyecto, decidimos

usar el lenguaje C, porque una de sus principales características es que permite tener acceso a memoria de bajo nivel mediante el uso de punteros, esto proporciona mayor flexibilidad en el manejo de hardware.

3. Para optimizar los recursos de la DE0nano, decidimos usar compuertas lógicas AND para determinar el funcionamiento de una matriz, un display 7 segmentos o Bus de datos, ya que con un solo pin GPIO de la DE0Nano podemos establecer el estado de cada uno de ellos.

4. Se decidió construir las compuertas lógicas AND con resistencias y diodos, ya que en el futuro estos elementos se encontrarán con mayor facilidad en el mercado que los integrados.

5. Para de determinar el funcionamiento del oscilador 555, se usa una herramienta existente dentro del conjunto de funciones que posee NIOS II llamada TIMER, la cual toma el tiempo que demora el oscilador en dar 10 flancos de reloj.

6. Para la prueba del oscilador 555 y la del convertidor ADC se decidió usar el potenciómetro digital MCP4132, ya que este dispositivo permite configurar diferentes valores resistivos durante las pruebas, haciendo variar frecuencia y ganancia de los componentes, con esto podemos obtener varias muestras para determinar sus estados.

7. Para realizar las pruebas del DAC y ADC se utilizan pines bidireccionales GPIO de la tarjeta DE0 Nano configurados como entrada cuando se prueba el ADC0804 y de salida cuando se prueba el DAC0808, de esta manera optimizamos recursos.

RECOMENDACIONES

1. Se recomienda realizar varias pruebas a los integrados para poder determinar los valores de referencia y programarlos en NIOS II IDE para verificar automáticamente el estado de los componentes.

2. Para la lectura de la señal de salida del DAC0808 se recomienda que el amplificador operacional LM741 sea regulado con salida máxima de 3.3V para no afectar el funcionamiento del ADC que posee la tarjeta DE0nano, ya que

en las especificaciones técnicas del fabricante se detalla que la tarjeta no soporta voltajes superiores a 3.3V.

3. Se recomienda usar pulseras antiestáticas, al momento de manipular la circuitería interna del probador.

BIBLIOGRAFÍA

[1] Giler Ortiz Víctor Moisés, Quinteros Zea Wilson Antonio, Implementación de un sistema automatizado y centralizado del proceso de préstamo y devolución de los implementos del Laboratorio de Sistemas Digitales de la Escuela Superior Politécnica del Litoral, http://www.cib.espol.edu.ec/Digipath/D_Tesis_PDF/D-99143.pdf, Diciembre 2013.

[2] Terasic Technologies Inc., Tarjeta de Desarrollo y Educación DE0-Nano, https://www.terasic.com.tw/cgi-bin/page/archive.pl?No=593, Diciembre 2013.

[3] Terasic Technologies Inc., DE0-Nano Specification, https://www.terasic.com.tw/cgibin/page/archive.pl?Language=English&CategoryNo=165&No=593&PartNo=2, Diciembre 2013.

[4] Altera Corporation, DE0-Nano Development and Education Board, http://www.altera.com/education/univ/materials/boards/de0-nano/unv-de0-nano-board.html, Diciembre 2013.

[5] Jorge Rodríguez Araújo, "Estudio del Procesador Nios II",http://es.scribd.com/doc/28358833/Estudio-del-microprocesador-Nios-II, Diciembre 2013

[6] Ing. Eric López Pérez, Protocolo SPI, http://www.i-micro.com/pdf/articulos/spi.pdf, Diciembre 2013.

[7] Ronald Mijaíl Dueñas, “El Estándar RS232 y V24”, http://interface-serial-rs232.blogspot.com/, Diciembre 2013.

[8] Christopher E. Strangio, The RS232 Standard, http://www.camiresearch.com/Data_Com_Basics/RS232_standard.html, Diciembre 2013.

[9] Wikipedia, “Potenciómetro”, http://es.wikipedia.org/wiki/Potenci%C3%B3metro, Diciembre 2013.

[10] Patagonia Tecnology, USB TTL Puerto de Comunicación UART, http://saber.patagoniatecnology.com/usb-ttl-puerto-de-comunicacion-uart-arduino-argentina-ptec/, Diciembre 2013.

[11] José Hipólito Pascual, Conexión de una Pantalla LCD a un Microcontrolador,http://server-die.alc.upv.es/asignaturas/lsed/2002-03/Pantallas_LCD/LCD.pdf, Diciembre 2013.

[12] Richard Pazán, LCD 4x20 char JHD204A STN blue/white LED, http://www.electrokit.com/en/lcd-4x20-char-jhd204a-stn-blue-white-led.46173, Diciembre 2013.

[13] Módulo teclado matricial http://www.disca.upv.es/aperles/web51/modulos/teclado/, Diciembre 2013.

ANEXOS

ANEXO A

CÓDIGO FUENTE

```
#include <math.h>
#include <time.h>
#include <string.h>
#include <io.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <system.h>
#include <altera_up_avalon_parallel_port.h>
#include "altera_up_avalon_parallel_port_regs.h"
#include "paneles.h"
#define ALTO 0x8000
#define MEDIO 0x0200
#define BAJO 0x0100
void ProbarDisplay();
void ProbarMatriz();
void Analizar555();
void Probar555();
void Send16SPI(int CMD16, volatile int *CS, volatile int *Sclk, volatile
int *Dout);
int Rate_555();
void ProbarADC0804();
void ProbarDAC0808();
void ProbarBUS();
void LCD_Init();
void LCD_Clear();
void ProcesarComando(char comando);
void ProcesarDato(char letra);
void LCD_Write(char* cadena, int longitud, int linea);
void Mostrar555(int valor);
void MostrarDAC(float valor);
void proceso();

void binario(int n){
       char c;
    if (n > 0) {
        binario(n/2);
        printf("%d", n%2);
        c = (n%2) + 48;
        ProcesarDato(c);

    }
}
```

```c
// Punteros para Probar Displays (7segmentos y Matriz)
volatile int *DisplayTipo = (int *)DISPLAY_TYPE_BASE;
volatile int *Seg7Detect = (int *)SEG7_DETECT_BASE;
volatile int *Seg7Out = (int *)SEG7_OUT_BASE;
volatile int *Seg7In = (int *)SEG7_IN_BASE;
volatile int *MatrixDetect = (int *)MATRIX_DETECT_BASE;
volatile int *MatrixOut = (int *)MATRIX_OUT_BASE;
volatile int *MatrixIn = (int *)MATRIX_IN_BASE;
// Punteros para Probar CI-555
volatile int *Dout_555 = (int *)MOSI_555_BASE;
volatile int *CS_555 = (int *)CS_555_BASE;
volatile int *Sclk_555 = (int *)SCLK_555_BASE;
volatile int *Entrada_555 = (int *)PROBE_555_BASE;
// Punteros para Probar ADC-DAC
volatile int *Dout_ADC = (int *)MOSI_ADC_BASE;
volatile int *CS_ADC = (int *)CS_ADC_BASE;
volatile int *Sclk_ADC = (int *)SCLK_ADC_BASE;
volatile int *Dato_Digital = (int *)DIGITAL_DATA_BASE;
volatile int *ADC_DE0nano = (int *)DE0NANO__ADC_BASE;
// Puntero para Probar IC's
volatile int *IC = (int *)IC_BASE;
// Puntero para Probar BUS
volatile int *BusIn = (int *)BUS_IN_BASE;
volatile int *BusOut = (int *)BUS_OUT_BASE;
// Punteros para manejo de LCD
volatile int *LCD_Data = (int *)LCD_DATA_BASE;
volatile int *LCD_Rs = (int *)LCD_RS_BASE;
volatile int *LCD_En = (int *)LCD_EN_BASE;
// Punteros para manejo de Indicador
volatile int *Indicador_Ok = (int *)INDICADOR_OK_BASE;
volatile int *Indicador_Err = (int *)INDICADOR_ERROR_BASE;


int main(){

	LCD_Init();
	LCD_Clear();
	LCD_Write("     FIEC-ESPOL     ",20,1);
	LCD_Write("********************",20,2);
	LCD_Write("   LABORATORIO DE   ",20,3);
	LCD_Write(" SISTEMAS DIGITALES ",20,4);
	usleep(500000);
	//ProbarDisplay();
	proceso();
	//serial();

	return 0;
}
```

```
void proceso(){
	char tecla;

	int resultado=7;
	char* message;
	char* message2;
	while(1){

	 inicio();
	do{
		tecla=leer_tecla();
		printf("  %c '\n'",tecla);
	}while(tecla!='D');
	while(1)
	{
	menu_principal();
	do{
		tecla=leer_tecla();
	  }while(tecla!='1' && tecla!='2' && tecla!='B');
	retardo(5);
	if(tecla=='1')
	 {
	  while(1){
			mi_menu_probar();
	do{
	    tecla=leer_tecla();
	}while(tecla!='1' && tecla!='2' && tecla!='3' && tecla!='4' &&
tecla!='B');
	retardo(5);
	if(tecla=='B')
	 {
	   tecla='X';
	   break;
	 }
	if(tecla=='1')
	 {
	  while(1){
			mi_menu_ic();
	 do{
	    tecla=leer_tecla();
	    }while(tecla!='1' && tecla!='2'&& tecla!='3'&& tecla!='B');
	 if(tecla=='B')
	  {
		tecla='X';
		break;
	   }
	 if(tecla=='1')
	  {
		while(1)
		{
```

```
          mi_menu_probar_cmos74();
          tecla=leer_dato();
 if(tecla=='B')
 break;
 strcpy(message,"CMOS&74");
 strcat(message,fila);
 printf("\n Valor de fila:%s",fila);

resultado=validar_numero(fila);
 printf("\nValor de resultado:%d",resultado);

validar_resultado(resultado);

printf("\nmessage2 ini: ");

printf("%s",message2);
printf("\nmessage ini: ");

printf("%s",message);
if (resultado==1)
{

strcpy(message2,message);

strcat(message2,"%OK");

printf("\n%s\n",message2);

LCD_Write("ESTADO: CORRECTO    ",20,4);

//serial(message2);
}
else {

       strcpy(message2,message);

       strcat(message2,"%FAIL");

       printf("\n%s\n",message2);

       LCD_Write("ESTADO:  DANADO     ",20,4);

//serial(message2);
      }
 retardo(1000);
 }
 }

if(tecla=='2')
 {
while(1)
```

```
 {

mi_menu_probar_cmos14();

tecla=leer_dato();

if(tecla==esc)break;

strcpy(message,"CMOS&14/40");

strcat(message,fila);
printf("\n Valor de fila:%s",fila);

resultado=validar_numero(fila);
printf("\nValor de resultado:%d",resultado);

validar_resultado(resultado);

printf("\nmessage2 ini: ");

printf("%s",message2);
printf("\nmessage ini: ");

printf("%s",message);
if (resultado==1)
{

       strcpy(message2,message);

       strcat(message2,"%OK");

       printf("\n%s\n",message2);

       LCD_Write("ESTADO: CORRECTO    ",20,4);

//serial(message2);
}
else
{

strcpy(message2,message);

strcat(message2,"%FAIL");

printf("\n%s\n",message2);

LCD_Write("ESTADO:  DAÑADO     ",20,4);

//serial(message2);
}
```

```
 retardo(1000);
  }
  }
  if(tecla=='3')
  {
  while(1)
 {

  mi_menu_probar_ttl();

  tecla=leer_dato();

  if(tecla==esc)break;

  strcpy(message,"TTL&74");

  strcat(message,fila);
 printf("\n Valor de fila:%s",fila);

  resultado=validar_numero(fila);
  printf("\nValor de resultado:%d",resultado);

  validar_resultado(resultado);

  printf("\nmessage2 ini: ");

  printf("%s",message2);
  printf("\nmessage ini: ");

  printf("%s",message);
  if (resultado==1)
  {
         strcpy(message2,message);

         strcat(message2,"%OK");

         printf("\n%s\n",message2);

         LCD_Write("ESTADO: CORRECTO    ",20,4);

  //serial(message2);
  }
  else
  {

         strcpy(message2,message);

         strcat(message2,"%FAIL");

         printf("\n%s\n",message2);
```

```
      LCD_Write("ESTADO:  DAÑADO     ",20,4);

//serial(message2);
}
retardo(1000);
}
}
limpiar_elemento();
}
}
if(tecla=='2'){
while(1){
mi_menu_probar_555();
do{
tecla=leer_tecla();
}while(tecla!='D' && tecla != 'B');
if(tecla=='B')
{
tecla = 'X';
break;
}
Analizar555();
}
}

if(tecla=='3')
 {//PROBAR CONVERTIDORES
while(1)
{
mi_menu_convertidores();
do{
tecla=leer_tecla();
}while(tecla!='1' && tecla!='2'&& tecla!='B');
if(tecla=='B')
{
tecla='X';
break;
}
if(tecla=='1')
{
while(1){

        mi_menu_probar_ADC0804();
do{
      tecla=leer_tecla();
while(tecla!='D' && tecla != 'B');
if(tecla=='B')
{
  tecla='X';
```

```
  break;
}

ProbarADC0804();

retardo(1000);

}
}

if(tecla=='2')
{
while(1)
{

      mi_menu_probar_DAC0808();
do
{
tecla=leer_tecla();
}while(tecla!='D' && tecla != 'B');
if(tecla=='B')
{
tecla='X';
break;
}
ProbarDAC0808();
retardo(1000);

}
}
}
}
if(tecla=='4'){// PROBAR DISPLAY 7 SEG, MATRIZ  Y BUS
while(1){
mi_menu_ProbarDisplayBUS();
do{
      tecla=leer_tecla();
}while(tecla!='1' && tecla!='2' && tecla!='3'&& tecla!='B');
if(tecla=='B')
 {
      tecla='X';
      break;
 }
if(tecla=='1')
 {
      while(1){

      mi_menu_ProbarDisplay_7seg();
      do{
      tecla=leer_tecla();
```

```
            }while(tecla!='D' && tecla != 'B');
      if(tecla=='B')
       {
            tecla='X';
            break;
       }
      ProbarDisplay();
      retardo(1000);
      }
      }
if(tecla=='2'){
while(1)
{

      mi_menu_ProbarMAtriz7x5();
do{

      tecla=leer_tecla();
  }while(tecla!='D' && tecla != 'B');
if(tecla=='B')
{
      tecla='X';
      break;
}
ProbarMatriz();
retardo(1000);
}
}
if(tecla=='3')
{
  while(1){

            mi_menu_ProbarBus40Pines();
do{

      tecla=leer_tecla();
  }while(tecla!='D' && tecla != 'B');
if(tecla=='B')
{
      tecla='X';
      break;
}
ProbarBUS();
retardo(1000);
}}}}
}}}}
void ProbarDisplay(){
      printf("Inserte Display para verificar...\n");

//    LCD_Write("  INSERTAR DISPLAY  ",20,1);
```

```
//     LCD_Write("--------------------",20,2);
       *Seg7Out = 0;
       *DisplayTipo = 1;
       while(*Seg7Detect!=0);

       printf("Display Detectado\n");
       LCD_Clear();
//     LCD_Write(" DISPLAY DETECTADO  ",20,1);
       LCD_Write("VERIFICANDO DISPLAY ",20,1);
       LCD_Write("--------------------",20,2);

       usleep(1500000);
       *Seg7Out = 1;
       usleep(1000);
       if(*Seg7In == 0)
       {
              printf("Display Catodo Comun en buen estado\n");

              LCD_Write("DISPLAY CATODO COMUN",20,3);
              LCD_Write("     BUEN ESTADO    ",20,4);
       }
       else{
              *Seg7Out = 0;
              *DisplayTipo = 0;
              usleep(1000000);
              if(*Seg7In== 1){
                     printf("Display Anodo Comun en buen estado\n");
                     LCD_Write("DISPLAY ANODO COMUN ",20,3);
                     LCD_Write("     BUEN ESTADO    ",20,4);
            }
       else{
              printf("Display DAÑADO\n");
              LCD_Write("  DISPLAY: DANADO   ",20,4);
            }
       }
       *DisplayTipo = 1;
       usleep(1000000);
       *Seg7Out = 0;
       printf("Retire Display\n\n");
       LCD_Clear();
       LCD_Write("   RETIRE DISPLAY   ",20,2);
       while(*Seg7Detect==0);
       usleep(1000000);
       }

      void ProbarMatriz(){
       int Filas_Correctas = 0, i=0;
       printf("Inserte Matriz para verificar...\n");
       LCD_Clear();
     //LCD_Write("   INSERTE MATRIZ   ",20,1);
```

```
    //
      *MatrixOut = 0;
      *DisplayTipo = 1;
      while(*MatrixDetect!=0);
      printf("Matriz Detectada\n");
//    LCD_Write("MATRIZ DETECTADA    ",20,3);
      usleep(1000000);
      LCD_Clear();
      LCD_Write("VERIFICANDO MATRIZ  ",20,1);
      LCD_Write("--------------------",20,2);
      *MatrixOut = 1;
      for(i=0;i<7;i++){
            usleep(1000);
            if(*MatrixIn == 1){
                  Filas_Correctas++;
                  printf("Fila %d en buen estado\n", i+1);
            }
            else
                  printf("Fila %d en mal estado\n", i+1);
            usleep(62500);
            *MatrixOut = (*MatrixOut)*2;

           }
      if(Filas_Correctas==7)
      {
                  printf("Matriz Fila-Anodo Comun en buen estado\n");
                  LCD_Write("   MATRIZ FILA-AC:  ",20,3);
                  LCD_Write("    BUEN ESTADO     ",20,4);

      }
      else{
                  printf("Matriz DAÑADA\n");
                  //LCD_Write("--------------------",20,2);
                  LCD_Write("   MATRIZ DANADA    ",20,3);
          }
      *DisplayTipo = 1;
      *MatrixOut = 127;
      usleep(1000000);
      *MatrixOut = 0;
      printf("Retire Matriz\n\n");
      LCD_Clear();
      LCD_Write("   RETIRE MATRIZ    ",20,3);
      while(*MatrixDetect==0);
      usleep(1000000);
          }

     void Analizar555(){
      long retardos=0;
      Send16SPI(ALTO, CS_555, Sclk_555, Dout_555);
      if(*Entrada_555==1){// SI ESTA EN BAJO ESPERA POR UN ALTO
```

```
        while(retardos<75){
                usleep(1000);
                retardos++;
                if(*Entrada_555==0){
                        printf("CI-555 DETECTADO\n");
                        break;
                }
        }
        usleep(500000);
        if(retardos<75)
                Probar555();
        else{
                printf("CI-555: DAÑADO\n\n");
                LCD_Write("CI-555 : MAL ESTADO ",20,4);
        //      serial("LM&LM555%FAIL");
        }
      }
  else{// SI ESTA EN ALTO ESPERA POR UN BAJO
        while(retardos<75){
                usleep(1000);
                retardos++;
                if(*Entrada_555==1){
                        printf("CI-555 DETECTADO\n");
                        break;
                }
        }
        usleep(500000);
        if(retardos<75)
                Probar555();
        else{
                printf("CI-555: DAÑADO\n\n");
                LCD_Write("CI-555 : MAL ESTADO ",20,4);
        //      serial("LM&LM555%FAIL");
        }
     }
   }

void Probar555(){
 LCD_Clear();
 int val1, val2, val3;
 Send16SPI(ALTO, CS_555, Sclk_555, Dout_555);
 usleep(10000);
 val1=Rate_555();
 printf("Tiempo 1: %d useg\n",val1);
 LCD_Write("[T1 < 0.8]s:    ",15,1);
 Mostrar555(val1);
 ////////////////////////////////
 Send16SPI(MEDIO, CS_555, Sclk_555, Dout_555);
 usleep(10000);
 val2=Rate_555();
```

```
    printf("Tiempo 2: %d useg\n", val2);
    LCD_Write("T2[2.11-2.16]s: ",15,2);
    Mostrar555(val2);
    ////////////////////////////////////
    Send16SPI(BAJO, CS_555, Sclk_555, Dout_555);
    usleep(10000);
    val3=Rate_555();
    printf("Tiempo 3: %d useg\n", val3);
    LCD_Write("[T3 > 3]s:     ",15,3);
    Mostrar555(val3);
    printf(" %d   %d  %d\n",val1,val2,val3);
    ////////////////////////////
    if(val1<800000 && (val2>2110000 && val2<2160000) && val3>3580000){
           printf("CI-555 : Buen Estado\n\n");
           LCD_Write("CI-555 : BUEN ESTADO",20,4);
  //       serial("LM&LM555%OK");
           ////////////////////////////////////
    }
    else{
           printf("CI-555 : Mal Estado\n\n");
           LCD_Write("CI-555 : MAL ESTADO ",20,4);
 //        serial("LM&LM555%FAIL");
           ////////////////////////////////
    }
    usleep(1000000);
   }

void Send16SPI(int CMD16, volatile int *CS, volatile int *Sclk, volatile
int *Dout){
    int i, Data;
    Data = CMD16;
    *CS=1;
    usleep(100);
    *CS=0;
    usleep(100);
    for(i=0 ; i<16 ; i++){
           *Dout=Data%2;
           *Sclk=0;
           usleep(1);
           *Sclk=1;
           usleep(1);
           Data=Data/2;
    }
    *CS=1;
  }

/* Funcion para medir el tiempo que demora en dar 10 flancos(555)*/
    int Rate_555(){
    int TIME=0, CLOCKS=0;
    long HIGH, LOW, COUNTER;
```

```
	// Setear Contador en H:L (FFFF:FFFF)
	//                         reg2:reg3
	IOWR(INTERVAL_TIMER_BASE,2,0xFFFF);
	IOWR(INTERVAL_TIMER_BASE,3,0xFFFF);
	while(*Entrada_555==1);
	while(*Entrada_555==0);
	IOWR(INTERVAL_TIMER_BASE,1,4);//START
	for(CLOCKS=0 ; CLOCKS<10 ; CLOCKS++){
		while(*Entrada_555==1);
		while(*Entrada_555==0);
	}
	IOWR(INTERVAL_TIMER_BASE,1,8);//STOP
	IOWR(INTERVAL_TIMER_BASE,4,0);//CAPTURA PARATE BAJA
	IOWR(INTERVAL_TIMER_BASE,5,0);//CAPTURA PARATE ALTA
	LOW=0xFFFF & IORD(INTERVAL_TIMER_BASE,4);
	HIGH=0xFFFF & IORD(INTERVAL_TIMER_BASE,5);
	COUNTER=0xFFFFFFFF & (LOW + (HIGH<<16));
	TIME=((0xFFFFFFFF-COUNTER)/50);
	return TIME;
}

void ProbarADC0804(){
	int Dato1, Dato2, Dato3;
	IOWR_ALT_UP_PARALLEL_PORT_DIRECTION(DIGITAL_DATA_BASE, 0x00);
	Send16SPI(ALTO, CS_ADC, Sclk_ADC, Dout_ADC);
	usleep(100000);
	Dato1 = *Dato_Digital;
	//printf("D1 [7-10]:  ");
//	printf(" (Decimal: %d)\n",Dato1);
	LCD_Clear();
	LCD_Write("D1 [7-10]:  ",12,1);
	binario(Dato1);
	printf(" (Decimal: %d)\n",Dato1);
	////////////////////////////////////////
	Send16SPI(MEDIO, CS_ADC, Sclk_ADC, Dout_ADC);
	usleep(100000);
	Dato2 = *Dato_Digital;
	printf("Dato 2 [126-129]: ");
	LCD_Write("D2[126-129]:",12,2);
	binario(Dato2);

	printf(" (Decimal: %d)\n",Dato2);
	////////////////////////////////////
	Send16SPI(BAJO, CS_ADC, Sclk_ADC, Dout_ADC);
	usleep(100000);
	Dato3 = *Dato_Digital;
	printf("Dato 3 [168-171]: ");
	LCD_Write("D3[168-171]:",12,3);
	binario(Dato3);
```

```
	printf(" (Decimal: %d)\n",Dato3);
	////////////////////////////////////


	if((Dato1>6 && Dato1<11) && (Dato2>125 && Dato2<130) && (Dato3>167
&& Dato3<172)){
		printf("Convertidor ADC: Buen Estado\n\n");
		LCD_Write("ADC: BUEN ESTADO    ",20,4);
	//	serial("ADC/DAC&ADC808%OK");
	}
	else{
		printf("Convertidor ADC: Mal Estado\n\n");
		LCD_Write("ADC: MAL ESTADO     ",20,4);
		//serial("ADC/DAC&ADC808%FAIL");

	}
	 }

	void ProbarDAC0808(){
	float Dato1, Dato2, Dato3;
	IOWR_ALT_UP_PARALLEL_PORT_DIRECTION(DIGITAL_DATA_BASE, 0xFF);
	*Dato_Digital=0x00;
	printf("Enviando 00\n");
	usleep(1000000);
	*(ADC_DE0nano)=0;
	Dato1=(*(ADC_DE0nano)*3.33)/4096;
	////////////////////////////////
	LCD_Clear();
	LCD_Write("[D1<0.05]V    : ",16,1);
	MostrarDAC(Dato1);
	printf("Lectura 1: %fV\n", Dato1);
	getchar();
	*Dato_Digital=0x80;
	printf("Enviando 80\n");
	usleep(1000000);
	*(ADC_DE0nano)=0;
	Dato2=(*(ADC_DE0nano)*3.3)/4096;
	printf("Lectura 2: %fV\n", Dato2);
	////////////////////////////////
	LCD_Write("D2[1.55-1.64]V: ",16,2);
	MostrarDAC(Dato2);
	getchar();
	*Dato_Digital=0xFF;
	printf("Enviando FF\n");
	usleep(1000000);
	*(ADC_DE0nano)=0;
	Dato3=(*(ADC_DE0nano)*3.3)/4096;
	printf("Lectura 3: %fV\n", Dato3);
```

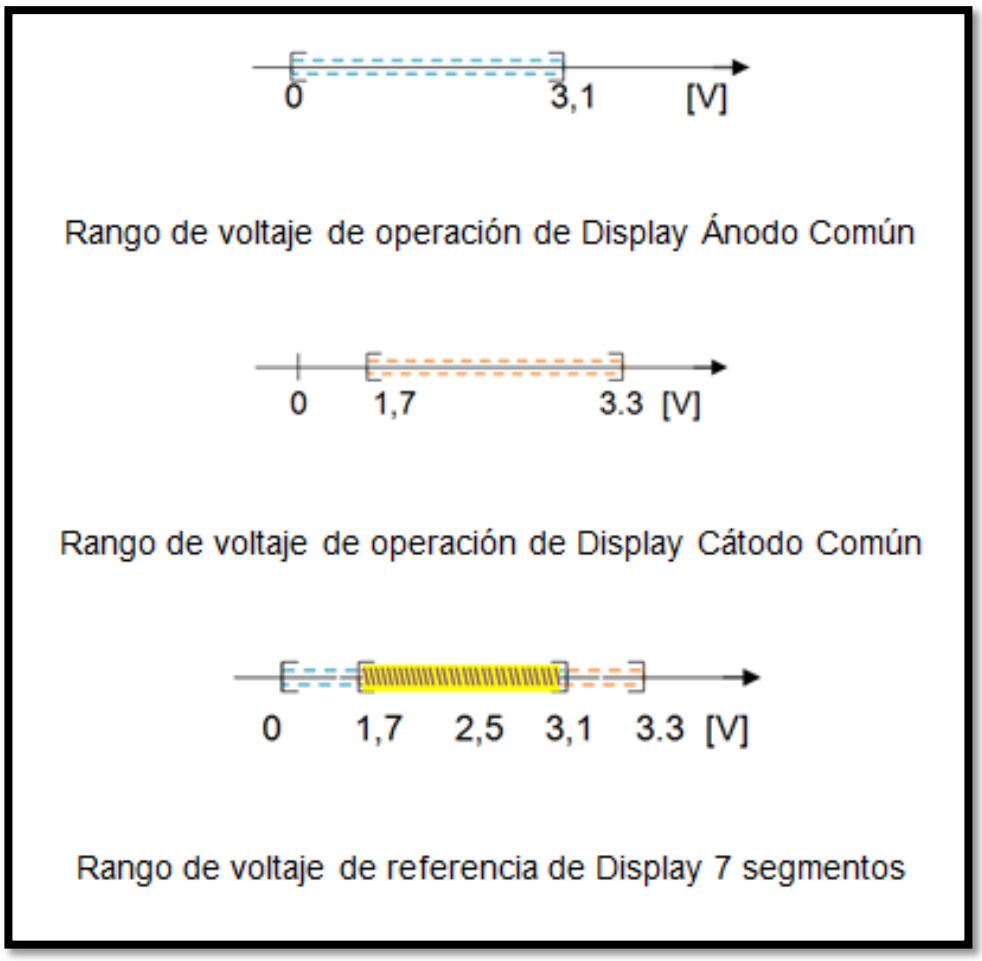
```
   ////////////////////////////////
   LCD_Write("[D3>3.20]V    : ",16,3);
   MostrarDAC(Dato3);
   getchar();
   if(Dato1<0.05 && (Dato2>1.55 && Dato2<1.64) && Dato3>3.20){
          printf("Convertidor DAC: Buen Estado\n\n");
          LCD_Write("DAC: BUEN ESTADO    ",20,4);
   }
   else{
          printf("Convertidor DAC: Mal Estado\n\n");
          LCD_Write("DAC: MAL ESTADO     ",20,4);
   }
   }

    void ProbarBUS(){
    int c;

    LCD_Write("  VERIFICANDO BUS   ",20,1);
    LCD_Write("--------------------",20,2);

          *BusOut =1;
          printf("Enviando Bus out 11\n");
          usleep(100000);
          c=*BusIn;
          printf("BusIn : %d",c);
   if( *BusIn==1 ){
          printf("BUS DE 40 PINES BUEN ESTADO\n");
          LCD_Write("  BUS DE 40 PINES   ",20,3);
          LCD_Write("     BUEN ESTADO    ",20,4);
   }
   else{
          printf("BUS DE 40 PINES BUEN ESTADO\n");
          LCD_Write("     BUS DE DATOS   ",20,3);
          LCD_Write("      MAL ESTADO    ",20,4);

   }

  }

void LCD_Init(){
ProcesarComando(0x3C);
ProcesarComando(0x0C);
ProcesarComando(0x06);
}

void LCD_Clear(){
ProcesarComando(0x01);
}
```

```
void ProcesarComando(char comando){
*LCD_Data=0x00;
*LCD_Rs=0;
*LCD_Data=comando;
usleep(1000);
*LCD_En=1;
   usleep(1);
   *LCD_En=0;
usleep(50);
}

void ProcesarDato(char letra){
   *LCD_Data=0x00;
*LCD_Rs=1;
*LCD_Data=letra;
usleep(1000);
*LCD_En=1;
   usleep(1);
   *LCD_En=0;
usleep(50);
}

void LCD_Write(char* cadena, int longitud, int linea){
int i=0;
char code;
switch(linea){
   case 1: code=0x80; break;
   case 2: code=0xC0; break;
   case 3: code=0x94; break;
   case 4: code=0xD4; break;
}
ProcesarComando(code);
usleep(2000);
for(i=0; i<longitud; i++){
    ProcesarDato(cadena[i]);
}
}

void Mostrar555(int valor){
   int temporal;
   char c;
   temporal = valor/1000000;
   c=(char)temporal+48;
   ProcesarDato(c);
   ProcesarDato('.');
   temporal = (valor/100000)%10;
   c=(char)temporal+48;
   ProcesarDato(c);
```

```
 temporal = (valor/10000)%10;
 c=(char)temporal+48;
 ProcesarDato(c);
 ProcesarDato('s');
}

void MostrarDAC(float valor){
 int temporal,newValue;
 char c;
 newValue = (valor * 100);
 temporal = (int)newValue/100;
 c=(char)temporal+48;
 ProcesarDato(c);
 ProcesarDato('.');
 temporal = (newValue/10)%10;
 c=(char)temporal+48;
 ProcesarDato(c);
 temporal = (newValue)%10;
 c=(char)temporal+48;
 ProcesarDato(c);
 }
```

ANEXO B

RANGO DE DECISIÓN PARA VOLTAJE DE REFERENCIA DEL DISPLAY 7 SEGMENTOS.



ANEXO C

ASIGNACIÓN DE PINES DE LA TARJETA DE0 NANO

JP1						
NOMBRE	PINES	PIN FPGA		PIN FPGA	PINES	NOMBRE
RDX	1	PIN_A8		PIN_D3	2	TDX
PROBE_555	3	PIN_B8		C3	4	
MOSI_555	5	PIN_A2		A3	6	MOSI_ADC
SCLK_555	7	PIN_B3		B4	8	SCLK_ADC
CS_555	9	PIN_A4		B5	10	CS_ADC
	11	VCC5		GND	12	
DISPLAYTYPE	13	PIN_A5		D5	14	MATIN
SEG7DETEC	15	PIN_B6		A6	16	MATOUT[0]
SEG7OUT	17	PIN_B7		D6	18	MATOUT[1]
SEG7IN	19	PIN_A7		C6	20	MATOUT[2]
INDICADOR_OK	21	PIN_C8		E6	22	MATOUT[3]
INDICADOR_ERROR	23	PIN_E7		D8	24	MATOUT[4]
BUS_IN	25	PIN_E8		F8	26	MATOUT[5]
BUS_OUT	27	PIN_F9		E9	28	MATOUT[6]
	29	VCC3		GND	30	
	31	PIN_C9		PIN_D9	32	MATDETEC
DIG_DATA[7]	33	PIN_E11		PIN_E10	34	DIG_DATA[0]
DIG_DATA[6]	35	PIN_C11		PIN_B11	36	DIG_DATA[1]
DIG_DATA[5]	37	PIN_A12		PIN_D11	38	DIG_DATA[2]
DIG_DATA[4]	39	PIN_D12		PIN_B12	40	DIG_DATA[3]

JP2					
NOMBRE	PINES	PIN FPGA	PIN FPGA	PINES	NOMBRE
	1	PIN_T9	PIN_F13	2	LCD_DATA[7]
	3	PIN_R9	PIN_T15	4	LCD_DATA[6]
LCD_RS	5	PIN_T14	PIN_T13	6	LCD_DATA[5]
LCD_EN	7	PIN_R13	PIN_T12	8	LCD_DATA[4]
LCD_DATA[0]	9	PIN_R12	PIN_T11	10	LCD_DATA[3]
	11	VCC5	GND	12	
LCD_DATA[1]	13	PIN_T10	PIN_R11	14	LCD_DATA[2]
IC[23]	15	PIN_P11	PIN_R10	16	IC[0]
IC[22]	17	PIN_N12	PIN_P9	18	IC[1]
IC[21]	19	PIN_N9	PIN_N11	20	IC[2]
IC[20]	21	PIN_L16	PIN_K16	22	IC[3]
IC[19]	23	PIN_R16	PIN_L15	24	IC[4]
IC[18]	25	PIN_P15	PIN_P16	26	IC[5]
IC[17]	27	PIN_R14	PIN_N16	28	IC[6]
	29	VCC3	GND	30	
IC[16]	31	PIN_N15	PIN_P14	32	IC[7]
IC[15]	33	PIN_L14	PIN_N14	34	IC[8]
IC[14]	35	PIN_M10	PIN_L13	36	IC[9]
IC[13]	37	PIN_J16	PIN_K15	38	IC[10]
IC[12]	39	PIN_J13	PIN_J14	40	IC[11]

JP3						
NOMBRE	PINES	PIN FPGA		PIN FPGA	PINES	NOMBRE
	1	VCC		PIN_E15	2	
	3	PIN_E16		PIN_M16	4	
TECLADOIN0	5	PIN_A14		PIN_B16	6	TECLADOOUT0
TECLADOIN1	7	PIN_C14		PIN_C16	8	TECLADOOUT1
TECLADOIN2	9	PIN_C15		PIN_D16	10	TECLADOOUT2
TECLADOIN3	11	PIN_D15		PIN_D14	12	TECLADOOUT3
	13	PIN_F15		PIN_F16	14	
	15	PIN_F14		PIN_G16	16	
	17	PIN_G15			18	
	19				20	
	21				22	
	23			CH0	24	DE0NANO_IN
	25			GND	26	

ANEXO D

VALORES DE LAS PRUEBAS REALIZADAS A LOS ELEMENTOS PARA DETERMINAR EL RANGO DE OPERACIÓN DE LOS INTEGRADOS.

PRUEBA CI 555		
Prueba[1]	Prueba[2]	Prueba[3]
0x8000	0x0200	0x0100
Tiempo1 [us]	Tiempo2[us]	Tiempo3[us]
779890	2133765	3602529
779701	2132415	3607725
779212	2132550	3606456
779455	2134081	3608006
779497	2132305	3608268
779573	2132299	3607477
778997	2133613	3607099
778823	2132531	3605100
780067	2133365	3607347
779487	2132783	3607991
779881	2131768	3606707
778897	2132035	3606764
778933	2132640	3608388
779658	2135121	3610489
779563	2134149	3607789
779221	2132765	3604435
778721	2134178	3607362
779057	2134034	3608568
778988	2133120	3606911
779388	2132464	3607238

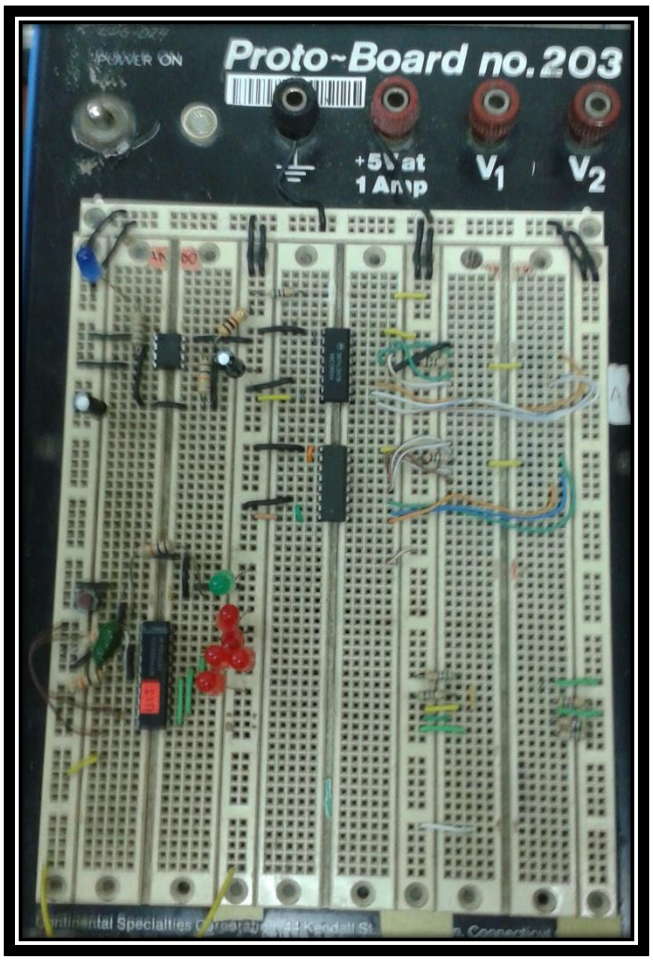
PRUEBA ADC0804					
Prueba[1]		Prueba[2]		Prueba[3]	
0x8000		0x0200		0x0100	
Lectura1		Lectura2		Lectura3	
Binario	Int	Binario	Int	Binario	Int
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1000	8	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1000	8	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169
1001	9	10000000	128	10101001	169

PRUEBA DAC0808		
Dato[1]	Dato[2]	Dato[3]
0X00	0x80	0xFF
Lectura1 [V]	Lectura2 [V]	Lectura3 [V]
0.00	1.60	3.25
0.00	1.60	3.25
0.00	1.60	3.25
0.00	1.60	3.26
0.00	1.60	3.25
0.00	1.60	3.24
0.00	1.60	3.25
0.00	1.60	3.26
0.00	1.60	3.25
0.00	1.60	3.25
0.00	1.60	3.25
0.00	1.60	3.25
0.00	1.60	3.25
0.00	1.60	3.26
0.00	1.60	3.26
0.00	1.61	3.25
0.00	1.61	3.26
0.00	1.60	3.25
0.00	1.59	3.28
0.00	1.59	3.28

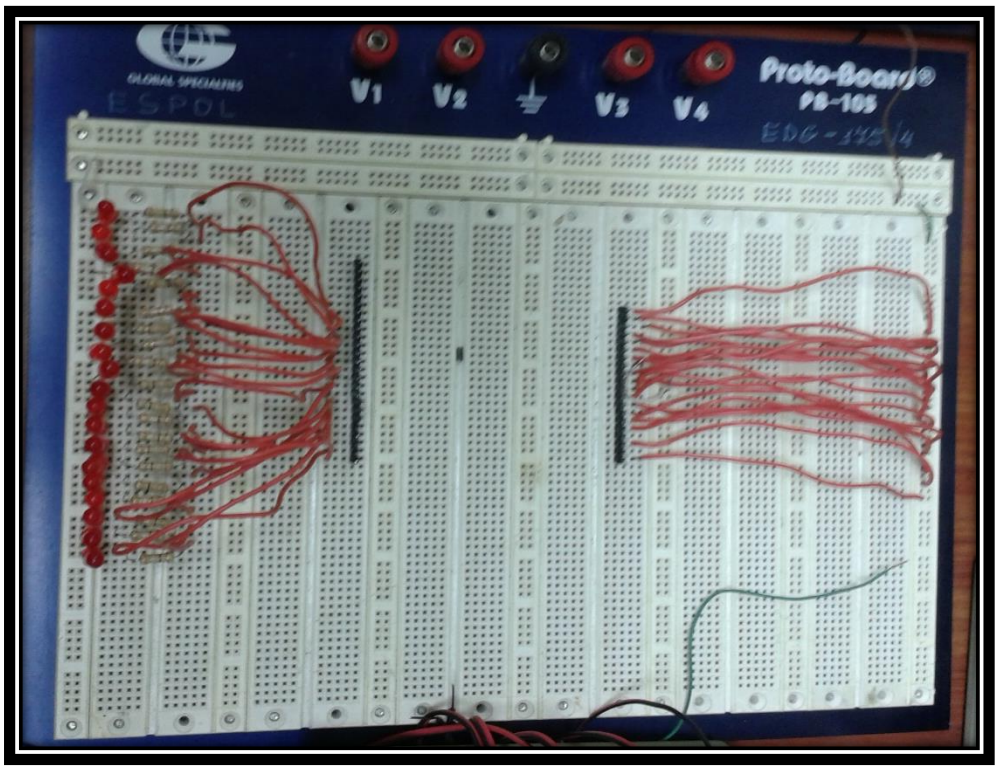
ANEXO E

IMÁGENES DEL SISTEMA DE VERIFICACION DE COMPONENTES QUE UTILIZA EL LABORATORIO DE SISTEMAS DIGITALES.

- PROBADOR 555, ADC0804, DAC0808, DISPLAY7SEG, MATRIZ 7X5



- PROBADOR BUS DATOS



- PROBADOR DE INTEGRADOS

ANEXO F

IMÁGEN DEL PROBADOR DE COMPONENTES ACTUALIZADO.

ANEXO G

DETALLE DE COSTO DEL PROBADOR DE COMPONENTES

	CANTIDAD	VALOR
PBC	1	$ 100,0
CASE	1	$ 200,0
RESISTENCIAS	60	$ 3,0
DIODOS	48	$ 2,4
CAPACITORES	2	$ 0,4
MCP4132	2	$ 5,0
LM324	7	$ 3,5
LM741	1	$ 0,5
LCD	1	$ 20,0
TECLADO 4X4	1	$ 10,0
DIP SOCKET	1	$ 20,0
ESPADINES	4	$ 4,0
LEDS	40	$ 8,0
FUENTE	1	$ 15,0
FPGA DE0Nano	1	$ 150,0
	TOTAL	$ 533,8