BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN CIENCIAS DE LA ELECTRÓNICA
OPCIÓN EN AUTOMATIZACIÓN

"DISEÑO E IMPLEMENTACIÓN DE UNA TARJETA CON FPGA RAM Y ROM EXTERNAS" **

T E S I S

Presentada para obtener el título de:

Maestro en Ciencias de la Electrónica Opción en Automatización

Presenta:

Ing. Geovanni Zepeda Arce*

Directores:

Dr. Sergio Vergara Limon

Dra. María Aurora Diozcora Vargas Treviño

Dr. José Fernando Reyes Cortés

Puebla, México **Enero 2018**

* Becario CONACYT.

** Trabajo financiado por proyecto VIEP.

BUAP

Agradecimientos

A Dios por haberme acompañado y guiado a lo largo de mi carrera, por darme la sabiduría, paciencia y fortaleza necesaria para poder lograr esta meta. Ya que sin su ayuda nada de esto sería posible.

A mis padres Pascual y Delfina que gracias a su apoyo, a su esfuerzo, confianza y cariño siempre me estuvieron alentando a seguir adelante sin importar las adversidades que se me presentaran tanto en el transcurso de mi carrera como en la realización de este trabajo.

A mi esposa Verónica, por haberme apoyado en las buenas y en las malas, por su paciencia, comprensión y cariño que me motivo a seguir luchando y no bajar los brazos. Sobre todo gracias por ser mi amiga y madre de mis hijas.

A mis hijas Ariadna, Marisa y Abigail que siempre me regalaron una sonrisa, un abrazo o un beso cuando más lo necesitaba y que son el motivo por el cual decidí realizar este trabajo y este objetivo en mi vida.

A mis amigos de toda la vida Ricardo y Alfredo, por animarme a entrar a la MCEA y a mis nuevos amigos Rigoberto, Francisco, Fernando, Moya, Raúl, Daniel, Hugo, Manuel, Gabino y Hermes por la ayuda que me brindaron en algún momento.

A mis asesores de tesis el Dr. Sergio Vergara Limon y el Dr. Fernando Reyes Cortés, por compartir tanto sus conocimientos científicos como personales que hicieron posible el desarrollo de este trabajo.

A la Dra. Aurora Vargas Treviño que siempre estuvo al pendiente de mis avances de tesis, que desinteresadamente siempre me ayudo en la elaboración y revisión de mis avances de tesis y artículo.

A mis sinodales por contribuir con sus conocimientos y consejos académicos para la correcta elaboración del presente trabajo.

A la Benemérita Universidad Autónoma de Puebla (BUAP), particularmente a la Facultad de Ciencias de la Electrónica por brindarme la oportunidad de adquirir los conocimientos que ahora tengo y poder concluir satisfactoriamente mis estudios de posgrado.

Al CONACYT por el apoyo económico que me dio en mi estancia en esta maestría y apoyar la creación de tecnología.

Resumen

El empleo de FPGAs (Field Programmable Gate Array) tiene ciertas ventajas, entre ellas la flexibilidad que se obtiene al desarrollar un prototipo. Un microprocesador o microcontrolador comercial tiene ciertas funciones que no pueden ser modificadas. En cambio, un FPGA puede ser reprogramado en poco tiempo con las funciones específicas que requiere un proyecto. En este trabajo se presenta el diseño e implementación de un microprocesador RISC (Reduced Instrucction Set Computer) de 32 bits destinado al control y automatización de sistemas. Las características principales del microprocesador son: una arquitectura Harvard y un conjunto de 16 instrucciones las cuales incluyen operaciones aritméticas con punto flotante (suma, resta, multiplicación y división), transferencia de datos, saltos condicionales y salto incondicional. Para la implementación de las instrucciones se diseñan los módulos: decodificador de instrucciones y Unidad Aritmética de Punto Flotante (ULA). El diseño de cada elemento se desarrolla en el lenguaje de descripción de hardware de Altera (AHDL). Para la instrumentación del sistema se diseña una tarjeta electrónica que tiene como base un FPGA Cyclone IV, una memoria RAM, una ROM y un módulo de interfaz inalámbrica Wi-Fi. La interfaz se utiliza para la carga, ejecución de algoritmos y la visualización de resultados desde un equipo de cómputo. Para demostrar la funcionalidad y practicidad de uso del sistema desarrollado, se presentan tres aplicaciones del sistema, este se aplicó en control de una mano robótica, en el control de un robot que emula los movimientos del cuello humano y en el control de un robot cartesiano de 3 grados de libertad. El procesador desarrollado permite resolver diversidad de problemas en forma secuencial suministrando versatilidad en la integración de sistemas digitales complejos, economizando recursos de hardware, esfuerzos de ingeniería y tiempo de desarrollo.

Contenido.

Índice de figuras.

Índice de tablas.

Introducción

Los sistemas digitales y su diseño han evolucionado mucho en las últimas cuatro décadas. Con el incremento en sus densidades y velocidad han proporcionado a los diseñadores una gran plataforma para crear sistemas digitales complejos. En la actualidad los sistemas embebidos utilizan microcontroladores de un solo chip. Los microcontroladores actuales están disponibles con procesamiento de 8, 16 y 32 bits, junto con un conjunto de periféricos como ADC, temporizador/contador, y redes (I^2C, SPI y UART). Para muchas aplicaciones el uso del microcontrolador en sistemas embebidos es adecuado. Pero para aplicaciones donde hay una necesidad de integrar lógica personalizada para un control más rápido y periféricos adicionales el microcontrolador o el microprocesador es mejorado con el uso de un FPGA o un producto estándar de aplicación específica (ASSP).

Los sistemas embebidos generalmente cuentan con uno o más microprocesadores, memorias, y periféricos dedicados, realizan tareas específicas y pueden formar parte de sistemas más complejos de hardware/ software. En el diseño de sistemas embebidos hay que considerar diferentes factores los cuales son el bajo costo, reducido número de componentes, bajo consumo de potencia y respuesta en tiempo de real. Una metodología general usada en el diseño de sistemas embebidos se muestra en la tabla 1 [1].

Tabla 1: Flujo de diseño de un sistema embebido.

Fase de diseño.	**Detalles de la fase de diseño.**
Requerimientos	Requisitos funcionales y requisitos no funcionales (tamaño, peso, consumo de energía y costo).
Especificaciones	Detalles de la interfaz de usuario junto con las operaciones necesarias para satisfacer la solicitud del usuario.
Arquitectura	Componentes de hardware (procesador, periféricos, lógica programable y ASSP), componentes de software (programas principales y sus operaciones)
Diseño de componentes	Componentes prediseñados, componentes modificados y nuevos componentes
Integración del sistema (hardware y software)	Esquema de verificación para descubrir errores rápidamente

La decisión sobre el tipo de plataforma digital que se utilizará se produce durante la fase de arquitectura del sistema, ya que cada aplicación embebida está vinculada a sus limitaciones operacionales únicas. Algunas de las restricciones de un controlador digital del hardware del sistema embebido incluyen (sin un orden particular) lo siguiente:

- Tasa de muestreo en tiempo real.
- Potencia
- Costo
- Solución en un solo chip
- Facilidad de programación
- Portabilidad del código
- Librerías de código reutilizable.
- Herramientas de programación

Hasta la década de 1970, los diseños de sistemas electrónicos se basaban en componentes analógicos discretos, como transistores, amplificadores operacionales, resistencias, condensadores e inductores. Estos circuitos ofrecían procesamiento simultáneo pero tenían problemas de variación de parámetros con la temperatura y el envejecimiento. La llegada de los componentes basados en TTL sentó las bases del diseño digital. El microprocesador Intel 4004 se convirtió en la primera plataforma digital que se podía configurar con software. La tabla 2 enumera los principales diseños digitales contemporáneos junto con su relativo mérito [2].

Tabla 2: Plataformas de diseño digital.

Plataforma de diseño digital	**Mérito**
Microprocesador	Reconfigurable usando software. Bueno para cálculos
Microcontroladores, controladores de señal digital.	Combinación de periféricos y un CPU
Producto estándar de aplicación específica (ASSP)	Un periférico especializado con la capacidad de comunicarse con un procesador host.
FPGA	Capacidad para combinar las fortalezas, del procesador, controlador y ASSP

El microprocesador ha cambiado la metodología de diseño digital como ningún otro componente digital. Comenzó como una CPU programable de 4 bits en 1971 y sigue siendo el controlador digital de elección en varias áreas de aplicación. El microprocesador trajo el concepto de arquitectura de conjunto de instrucciones (ISA), ensamblador y compilador. Hay muchas aplicaciones en tiempo real, con velocidades de actualización rápidas que requieren programar el microprocesador en su lenguaje ensamblador nativo. Esto se hace generalmente cuando el tamaño de la memoria disponible es una restricción. A pesar de que la mayoría de los microcontroladores comerciales utilizados hoy en día se adaptan a aplicaciones centradas en datos, hay microprocesadores embebidos en microcontroladores para aplicaciones de control en tiempo real.

El microcontrolador representa la próxima generación de controladores para sistemas integrados. Permite la creación de sistemas con menor número de componentes mediante la incorporación de periféricos que anteriormente estaban interconectados externamente con el procesador de propósito general. Al igual que el microprocesador, las tareas en un entorno de diseño de microcontroladores se dividen según las tasas de actualización requeridas. Para las tareas que requieren bajas tasas de actualización, la codificación se realiza utilizando un lenguaje de programación de software como C. Las tareas que necesitan tener altas tasas de actualización se codifican utilizando el lenguaje ensamblador nativo para un microcontrolador particular.

En la actualidad los FPGA (Field Programmable Gate Array) integran diseño de lógica digital, procesadores e interfaces de comunicación en un solo chip. Los microcontroladores actualmente tienen una ventaja sobre el FPGA en términos de potencia y costo. Pero los FPGA se están poniendo al día al ofrecer la portabilidad de código en varios proveedores de FPGA. Los dispositivos programables que tradicionalmente eran dispositivos que contaban con un reducido número de compuertas están ahora en una posición para soportar grandes partes de la lógica del sistema digital. Hoy el diseñador digital tiene una opción viable de usar solo el dispositivo FPGA como controlador del sistema embebido. La disponibilidad de dispositivos FPGA de alta densidad y bajo costo les ha dado a los diseñadores digitales mucha flexibilidad para diseñar arquitecturas digitales personalizadas usando FPGA y HDL (Hardware Description Language). Los dispositivos FPGA han evolucionado desde su predecesor lógico a un dispositivo que ahora contiene una gran variedad de componentes digitales incorporados (memorias, multiplicadores, transceivers y mucho más). La densidad de los dispositivos FPGA ha aumentado con el paso de los años y, al mismo tiempo, su costo ha hecho que sea económicamente viable para uso en varias aplicaciones. Los FPGAs actuales ofrecen:

- **Programación en sitio:** Los dispositivos programables de campo pueden reconfigurarse en cualquier momento. Los diseñadores pueden integrar modificaciones o hacer cambios completos.
- **Diseño definido por software**: El hardware está definido por lenguajes como software (HDL). Los diseñadores pueden desarrollar, simular y probar un circuito completamente antes de ejecutarlo en un dispositivo programable en campo.
- **Paralelismo:** Los circuitos definidos en el FPGA pueden diseñarse de manera paralela. Esto es similar a usar dispositivos analógicos de ruta múltiple. Un usuario puede crear instancias múltiples de hardware en un mismo chip sin interferencia entre módulos o carga de computación.
- **Alta velocidad**: Debido a que un FPGA es una implementación de hardware que funciona con velocidades de bloqueo rápidas, los diseñadores pueden alcanzar velocidades muy altas. Junto con el paralelismo, la implementación del FPGA puede superar los sistemas basados en procesador.

- **Confiabilidad:** Los diseñadores pueden esperar confiabilidad de hardware verdadera de los FPGA, por que no aquí no hay sistema operativo o capa de controlador que pueda afectar el tiempo de actividad del sistema.
- **Protección y reutilización de IP (Intellectual Property):** Una vez compilados y descargados en la implementación de hardware del FPGA es difícil realizar ingeniería inversa.

Con el aumento en el número de compuertas de los dispositivos FPGA, muchos proveedores de FPGA ofrecen un procesador que ya existe en silicón como un "Hard-Core" o puede incorporarse dentro del dispositivo programable como un "Soft-Core". El propósito de tener un procesador junto con los componentes lógicos digitales convencionales es proporcionar flexibilidad de combinar software y control basado en hardware en un chip.

La capacidad de admitir un procesador lógico ha traído una nueva dimensión al uso de dispositivos FPGA. Ha proporcionado a los diseñadores la libertad de dividir sus diseños ya sea para un flujo de software de subproceso único o para usar la lógica digital concurrente. Una búsqueda rápida en internet muestra que varios procesadores patentados de 8 y 32 bits son ofrecidos por vendedores líderes de FPGA. Casi todos los proveedores principales de dispositivos programables en campo proporcionan procesadores, para su uso con sus respectivos dispositivos. La tabla 3 se muestra una lista parcial de proveedores y los procesadores que ofrecen.

Un Soft-core es la descripción de un procesador específico que puede ser implementado en un dispositivo FPGA particular. Los proveedores de FPGA ofrecen soft-cores para aplicaciones de 8 y 32 bits. El procesador de 8 bits ocupa una pequeña superficie en el dispositivo FPGA y utiliza la memoria incorporada del FPGA para el almacenamiento de programas y datos. Entre los soft-cores de 32 bits, dos de los principales procesadores son NIOS de Altera y MicroBlaze de Xilinix. Estos procesadores usan una parte de los recursos del FPGA. La parte restante del FPGA se puede usar para incorporar otra lógica digital.

El procesador soft-core MicroBlaze de 32 bits desarrollado por Xilinix, para sus FPGAs de las familias Spartan y Virtex. Sigue una arquitectura Harvard con bus de memoria de datos e instrucciones separados. Es compatible tanto con el chip BlockRAM y/o memoria externa. Todos los periféricos se aplican sobre la estructura del FPGA. En la figura 1 se muestra un diagrama a bloques de un procesador MicroBlaze. Una de las principales características es que es muy confiable, pudiendo incluir o excluir una serie de elementos del microprocesador según las necesidades de la aplicación objetivo, permitiendo una gran variedad de configuraciones más o menos rápidas y que ocupa más o menos área en el FPGA. Entre las características más destacadas de este procesador tenemos 32 registros de propósito específico, instrucciones de 32 bits, unidad de punto flotante FPU, bus de direcciones de 32 bits, controladores de memoria SRAM, SDRAM, DDR y DDR2, dispositivos de comunicación serie I2C, UART y pipeline de 3 o 6 capas [3].

Tabla 3: Lista parcial de proveedores de procesadores basados en FPGA.

Processor name	Type/Bits	Interface bus	FPGA vendor
MicroBlaze™	Soft/32	IBM Coreconnect	Xilinx
NIOS®	Soft/32	Avalon	Altera
LatticeMico32	Soft/32	Wishbone	Lattice
CoreMP7	Soft/32	APB	Actel
ARM Cortex-M1	Soft/32	AHB	Vendor independent
LatticeMico8	Soft/8	Input/Output ports	Lattice
Core8051	Soft/8	Nil	Actel
Core8051s	Soft/8	APB	Actel
PicoBlaze™	Soft/8	Input/Output ports	Xilinx
PowerPC	Hard/32	IBM Coreconnect	Xilinx
AVR	Hard/8	Input/Output ports	Atmel

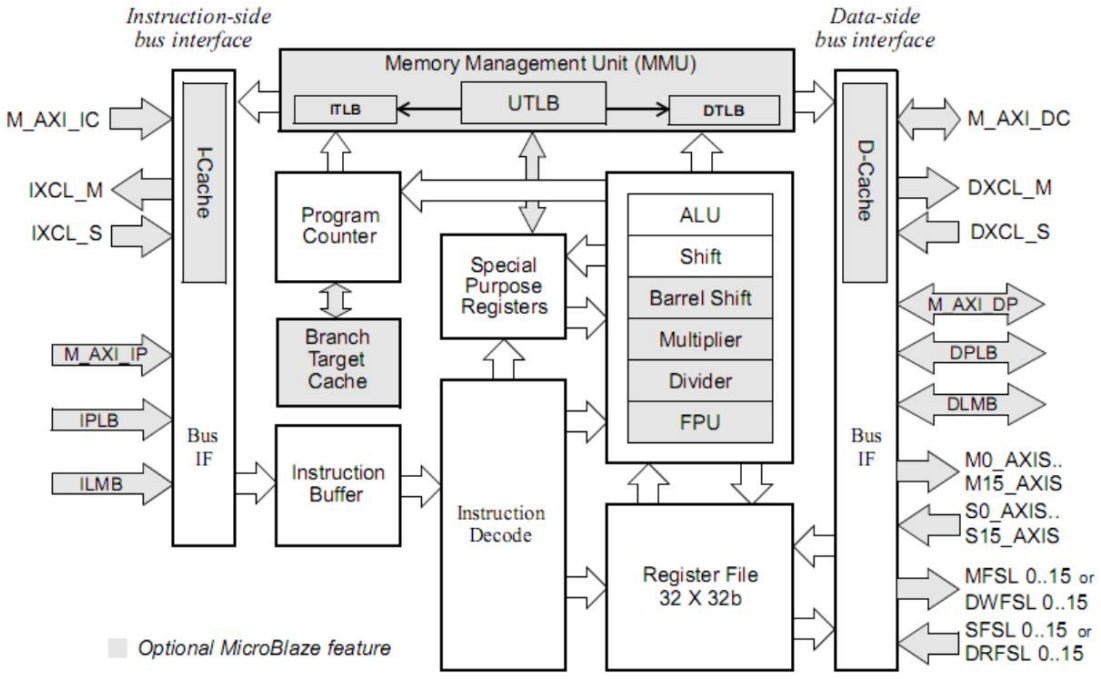


Figura 1: Diagrama a bloques de un procesador MicroBlaze.

El procesador Nios II es un soft-core específicamente diseñado por Altera para ser implementado en sus dispositivos FPGA de las familias Stratix y Cyclone. Es un

procesador RISC (Reduce Instruction Set Computer) de 32 bits con arquitectura Harvard. Entre las características más destacadas de este procesador tenemos 32 registros de propósito general 32 bits, operaciones de punto flotante de precisión simple, bus de direcciones de 32 bits, posibilidad de añadir lógica a la ALU para crear nuevas instrucciones de propósito específico, cache de instrucciones y datos opcionales [4]. En la figura 2 se muestra el diagrama a bloques del procesador Nios II.

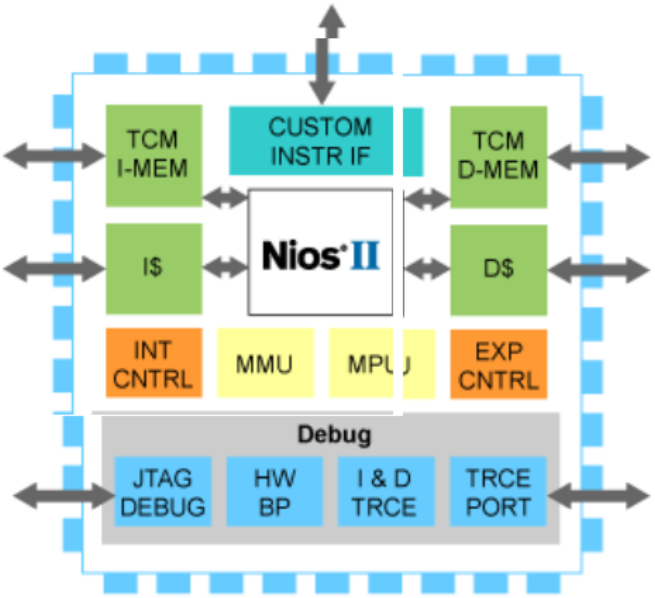


Figura 2: Diagrama a bloques de procesador Nios.

En sistemas embebidos basados en soft-core como principales ventajas se tienen la característica de diseño hardware-software, circuitos reconfigurables, integración de hardware adicional, el software del procesador puede ser actualizado sin modificar el hardware y los sistemas basados en hardware programable por lo general son versátiles ya que el usuario es quien define la lógica del sistema. Como desventajas del uso de un soft-core es que sus diseños no son completamente libres, ya que el procesador define ciertas características a respetar y el procesador puede llegar a consumir gran cantidad de recursos, dependiendo de las características con las que cuente. Debido a los inconvenientes que se presentan por el uso de un soft-core en este trabajo de tesis se optó por diseñar nuestro propio procesador embebido de 32 bits con algunas características de los procesadores de Xilinix y Altera ya mencionados como es la arquitectura Harvard, un conjunto reducido de instrucciones de 32 bits basadas en MIPS, una unidad aritmética de punto flotante de simple precisión y administradores de memorias externas, esto con el objetivo de desarrollar un sistema de código abierto que se ajuste a los requerimientos de cada proyecto y no requiera la compra de licencias de uso que impliquen altos costos. Para la instrumentación del microprocesador se diseña y desarrolla una tarjeta electrónica basada en FPGA con memorias externas y un módulo de comunicación inalámbrica. Mediante el módulo de interfaz inalámbrica WiFi se establece la comunicación entre el sistema

embebido y una computadora, para la carga de programas a la memoria ROM, la ejecución de programas, él envió de parámetros del sistema a controlar y la lectura de los resultados arrojados por nuestro programa desde la memoria RAM, mediante un software desarrollado en LabView. Para mostrar las ventajas que ofrece este sistema embebido basado en FPGA, se presentan tres aplicaciones del sistema embebido desarrollado, las aplicaciones son el control de una mano mecatrónica de 5 grados de libertad, el control de un sistema mecatrónico que emula los movimientos del cuello humano y el control de un robot de 3 grados de libertad. Está aplicación son solo un ejemplo de algunas aplicaciones.

Objetivo general.

"***Desarrollar un sistema embebido con RAM y ROM externas con la finalidad de automatizar diversos procesos***"

Objetivos específicos.

1. Investigar el estado del arte de sistemas embebidos como el propuesto.
2. Realizar el diseño electrónico del sistema embebido.
3. Realizar el diseño de la tarjeta de circuito impreso.
4. Diseño de CPU básico.
5. Implementación del sistema.
6. Instrumentación de la comunicación Wi-Fi.
7. Realizar pruebas del sistema completo.
8. Publicación de resultados.
9. Escritura de la tesis.

Respecto al contenido de esta tesis, en el capítulo uno se presenta la descripción de sistemas embebidos como el nuestro, se revisan los requerimientos del sistema embebido, así como las características de la tarjeta y del microprocesador embebido.

En el capítulo dos se presenta el diseño del sistema embebido tomando en cuenta los requerimientos y especificaciones del sistema.

En el capítulo tres se presenta el diseño del controlador del sistema embebido, que se compone de un microprocesador, administradores de memoria, administradores de puerto y administradores de WiFi y la descripción de cada una de las partes que lo conforman y se expone el formato y el conjunto de instrucciones utilizadas.

En el capítulo cuatro se exponen tres trabajos de control de robots manipuladores en los que se utilizó el sistema desarrollado en este trabajo, con el fin de mostrar la practicidad y velocidad de desarrollo que ofrece el uso de un microprocesador de 32 bits con instrucciones basadas en MIPS en FPGA.

Capítulo 1. Características del sistema embebido

El diseño de tarjetas de adquisición y control tiene un alto grado de complejidad, por lo que en su mayoría son desarrollados por compañías transnacionales en las cuales un equipo multidisciplinario realiza los diseños, estos son de gran valor para las compañías que los realizan como para su país. En este capítulo se presentan las características del sistema embebido a desarrollar, el cual consta de una tarjeta electrónica que tiene como base un FPGA y cuenta con memorias RAM y ROM externas y un módulo de comunicación inalámbrica WiFi. Este desarrollo es con la finalidad de tener un sistema más robusto que nos permita controlar y monitorear diversos procesos desde un equipo de cómputo de manera inalámbrica y la de seguir creando infraestructura tecnológica para en un futuro cercano tener un sistema que permita la solución a un problema de nuestro entorno. Los sistemas basados en FPGA están ganando aceptación en el terreno de los sistemas embebidos ya que ofrecen la plataforma para la integración de diseño de lógica digital, procesadores, memorias e interfaces de comunicación en un solo chip, lo que reduce costos y espacio en una tarjeta [5]. A continuación se presentan las tarjetas electrónicas basadas en FPGA que precede a la que se desarrolla en este trabajo.

1.1 Tarjeta FCE-BUAP con Cyclone II

En la figura 3 se muestra una tarjeta electrónica que tiene como base un FPGA Cyclone II de la marca Altera y corresponde al primer diseño de sistema embebido basado en FPGA desarrollado en la MCEA (Maestría en Ciencias de la Electrónica opción Automatización) de FCE-BUAP. La tarjeta cuenta con 135 terminales de usuario y 8256 elementos lógicos. Esta tarjeta es usada para prácticas de laboratorio y se ha utilizado para automatizar un CNC de 4 grados de libertad desde una PC mediante una tarjeta PCI [6]. Este fue un diseño de 4 capas y tiene componentes en la parte superior e inferior.

En la parte superior consta de:

(1) FGPA Cyclone II modelo EP2C8Q208C7.
(2) Un puerto de programación compatible con USB-BLASTER.
(3) Un conector de alimentación externa.
(4) Un botón de reinicio.
(5) 4 Conectores Header macho de 50 pines.

La parte inferior consta de:

(1) Memoria de configuración (EPC2)
(2) Convertidores DC-DC de 3.3V a 2.5 y de 3.3V a 1.2V.
(3) Oscilador de cristal de 100MHz
(4) Resistencias y capacitores.

Figura 3: Tarjeta FCE-BUAP con Cyclone II.

El empleo de sistemas embebidos basados en FPGA, ofrece ciertas ventajas, entre ellas la flexibilidad que se obtiene al desarrollar un prototipo, el FPGA puede ser reprogramado, en poco tiempo y con las funciones específicas que requiera una aplicación. En la mayoría de las aplicaciones con el paso del tiempo se tiene que hacer uso de nuevas tecnologías e implementar en el FPGA circuitos digitales cada vez más complejos que demandan mayores recursos como lo son los elementos lógicos, memoria, puertos de entrada/salida y periféricos, por ello en la MCEA se diseñó una siguiente tarjeta con un FPGA Cyclone III.

1.2 Tarjeta FCE-BUAP con Cyclone III

Esta segunda tarjeta basada en un FPGA Cyclone III, ofrece los recursos para la implementación de sistemas más completos al contar con 10000 elementos lógicos, 160 terminales de usuario y analizador lógico integrado. Este diseño fue de cinco capas y tiene componentes en la parte superior e inferior.

La parte superior, consta de:

(1) Un FPGA Cyclone III con matrícula EP3C10F256C6 de Altera.
(2) Un puerto de programación compatible con el programador USB-BLASTER
(3) Un puerto JTAG.
(4) Un conector para alimentación externa.
(5) Un botón de reinicio.
(6) 4 Conectores header macho de 50 pines.

La parte inferior, consta de:

(1) Memoria EEPROM (EPC2).

(2) Convertidores DC-DC de 3.3V a 2.5 y de 3.3V a 1.2V.
(3) Oscilador de cristal de 100MHz.
(4) Resistencias y capacitores.

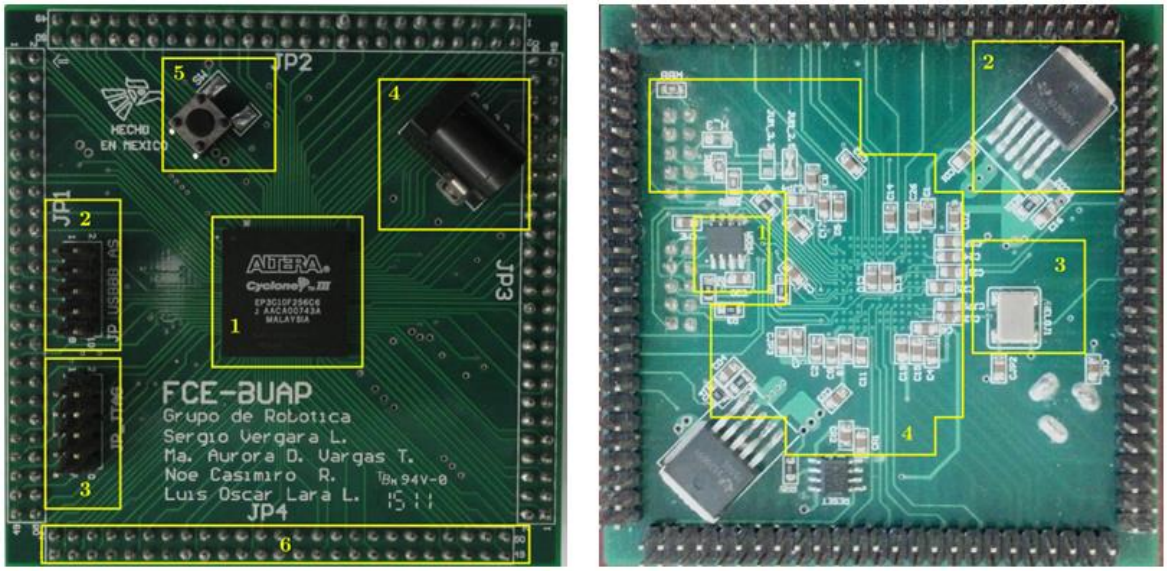


Figura 4: Tarjeta FCE-BUAP con Cyclone III, capa superior e inferior.

La tarjeta de la figura 4 es utilizada para prácticas de laboratorio y se ha utilizado para controlar el mismo CNC de 4 grados de libertad pero ahora vía WiFi [7], sintetizando en el FPGA un microprocesador, protocolos de comunicación UART y SPI, controladores de memoria, y demás circuitos digitales útiles para el control del robot. El FPGA Cyclone III de esta tarjeta cuenta con un mayor número de elementos lógicos respecto al anterior, pero con el aumento de requerimientos de las aplicaciones con las que se trabaja en la MCEA estos recursos ya no son suficientes por lo que se ha visto la necesidad de desarrollar una sistema embebido más robusto que cuente con memorias RAM y ROM externas que liberen al FPGA del almacenamiento de datos y sus recursos se utilicen únicamente para la implementación de circuitos digitales y que además cuente con un módulo de comunicación inalámbrica WiFi para controlar y monitorear de manera inalámbrica procesos desde una PC.

1.3 Tarjeta con FPGA, RAM y ROM externas

Se requiere diseñar un sistema que a diferencia de los diseños anteriores cuente con un FPGA de la marca Altera de la familia Cyclone, de mayor número de elementos lógicos, numero de I/O, memorias externas ROM y RAM para el almacenamiento de programas y datos respectivamente, un módulo WiFi para comunicar al sistema embebido con equipos de cómputo y lo principal que caracteriza a este trabajo de tesis es la implementación en el FPGA de un microprocesador RISC de 32 bits con arquitectura Harvard con un conjunto de 16 instrucciones de 32 bits basadas en MIPS y 16 registros de propósito general. Asegurándonos siempre de mantener en todo momento la compatibilidad electrónica y

mecánica con las tarjetas que le preceden, a continuación se muestra una lista de requerimientos del sistema que se comparten con los actuales y las nuevas características.

Características que se mantienen:

- 4 conectores (header) macho de 50 pines.
- Convertidores DC-DC de 3.3V a 2.5 y de 3.3V a 1.2V.
- Un botón de reinicio.
- Oscilador de cristal de 100 MHz.
- Puerto de programación para usb-blaster.
- Puerto JTAG.

Nuevas características.

- Memorias externas RAM y ROM.
- Un módulo de comunicación inalámbrica WiFi.
- Memoria de configuración para FPGA.
- FPGA Altera, de más de 25 mil elementos lógicos y más de 160 I/O de uso general.

Características embebidas en el FPGA.

- Microprocesador RISC de 32 bits con arquitectura Harvard.
- Administradores de memoria.
- Administrador WiFi.

1.4 Diagrama a bloques del sistema

En la figura 5 se muestra el diagrama a bloques del sistema embebido a desarrollar. Cabe mencionar que entre sus principales características se encuentra el diseño de la tarjeta que integra un FPGA, memorias externas, puertos de entrada/salida y un módulo de comunicación inalámbrica WiFi (Hardware), el diseño e implementación en el FPGA de un microprocesador RISC embebido de 32 bits con arquitectura Harvard, un administrador de memoria RAM, un administrador de memoria ROM, un módulo de interfaz inalámbrica WiFi, un módulo administrador de comandos y un módulo administrador de puertos de entrada y salida (Firmware), y el diseño de una interfaz gráfica de usuario diseñada en LabView que permite el grabado de programas en la ROM, grabar o leer datos de la RAM, ejecución y paro de programas y el envío de parámetros al sistema a controlar (Software).

Para la operación del sistema embebido, el usuario cuenta con un software de interfaz gráfica desarrollada en LabView que se comunica con el sistema embebido por medio de una interfaz WiFi, el cual recibe comandos y datos del software y los envía a un módulo administrador de WiFi, que serializa la información recibida y la envía al módulo administrador de comandos, para ser ordenada, decodificada y determinar si la información corresponde a un dato que tiene que ser enviado a las memorias o al microprocesador, o corresponde a señales de activación que permiten la carga de programas, lectura de memorias o el arranque y paro del microprocesador.

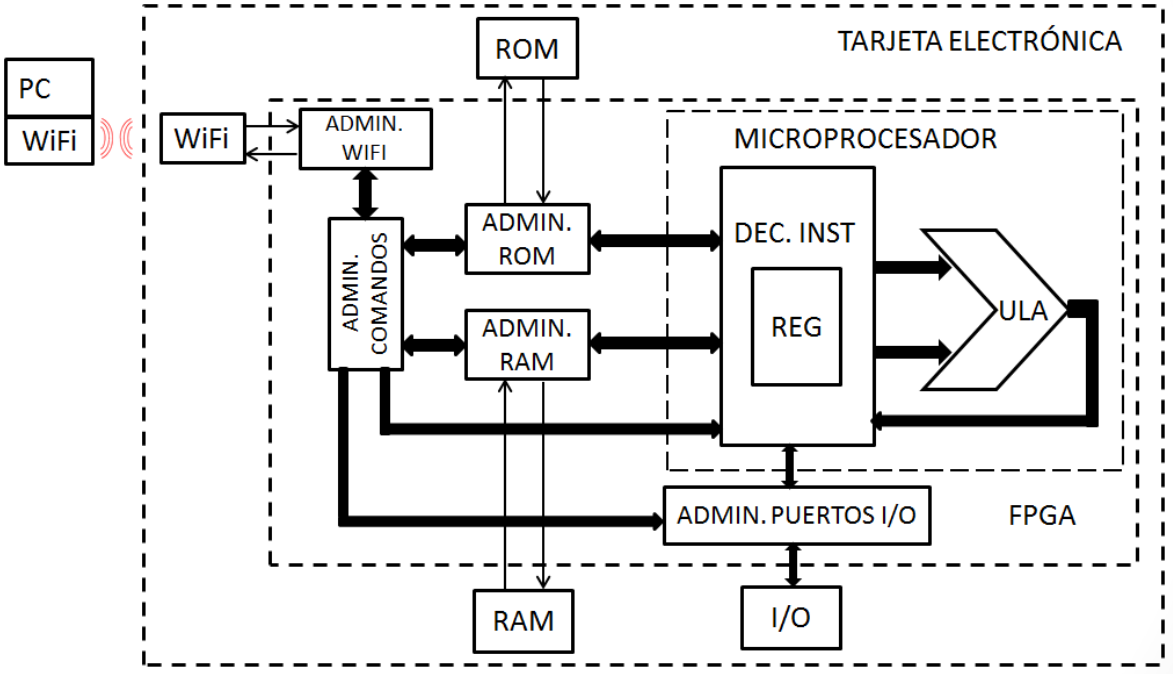


Figura 5: Diagrama a bloques general del sistema embebido.

Las memorias tipo RAM y ROM son dispositivos externos al FPGA, la memoria ROM se destina para el almacenamiento del conjunto de instrucciones que interpreta y ejecuta el microprocesador y la RAM al almacenamiento de datos a procesar. Se busca que estos circuitos integrados sean de comunicación serial, esto con la finalidad de ocupar el menor número de pines I/O del FPGA. Para leer y escribir datos en estos dispositivos en una localidad de memoria determinad se implementan en el FPGA administradores de memoria, los cuales tienen la función de determinar si los datos, direcciones y comandos para la lectura o escritura de las memorias se toman de las señales enviadas del microprocesador o del software de la PC. Además tienen la función de generan las señales correspondientes para establecer comunicación con los circuitos externos, así como de enviar los comandos para realizar los procesos de escritura, lectura y borrado de memoria. Estos módulos reciben los datos, direcciones y comandos del microprocesador en paralelo y los convierten a un tren de pulsos que indica a las memorias la localidad de memoria en la que deben leer o escribir.

El administrador de puertos de entradas y salidas es un módulo controlado por el microprocesador mediante instrucciones de lectura y escritura de puertos. Este módulo controla el acceso y salida de datos del microprocesador. Los puertos I/O permiten la interacción del microprocesador con dispositivos externos, para la automatización de procesos.

El elemento principal de este trabajo un microprocesador embebido en FPGA de 32 bits, con arquitectura Harvard que cuenta con un conjunto de 16 instrucciones de 32 bits (aritméticas, de transferencia y condicionales) basadas en MIPS, una unidad aritmética de

punto flotante de simple precisión IEEE 754, un módulo comparador de punto flotante para la ejecución de las instrucciones condicionales, 16 registros de propósito general de 32 bits cada uno. Este módulo es el encargado de realizar cálculos con gran rapidez obedeciendo un conjunto de instrucciones específicas y elementales. En breve se presentan el tipo y las instrucciones a implementar, así como también las características de la unidad aritmética de punto flotante de simple precisión IEEE754.

Mediante un software en LabView en la PC se graban instrucciones en la memoria de programa de la tarjeta (ROM), se ejecutan algoritmos y leen los datos arrojados por el procesador descargando el contenido de la memoria de datos (RAM) a una hoja de cálculo de Excel para su manipulación.

1.5 Instrucciones MIPS

El MIPS (Microprocessor without Interloked Pipelines Stage, o lo que es lo mismo, microprocesador sin enclavamiento de estados de tuberías) hace referencia a la gama de microprocesadores desarrollados por MIPS Technologies [8]. El MIPS es un procesador de 32 bits con arquitectura RISC (Esta arquitectura es un tipo de diseño de CPU que se caracteriza por instrucciones de tamaño fijo y presentadas en un reducido número de formatos además sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos su objetivo es permitir el paralelismo en la ejecución de instrucciones y reducir los accesos de memoria) y un banco con 32 registros de 32 bits. El conjunto de instrucciones permite realizar instrucciones de carga y almacenamiento desde y hacia memoria, tiene la capacidad de desarrollar programas que resuelven problemas aritméticos y lógicos, y ofrece la posibilidad de controlar el flujo de la ejecución del programa mediante instrucciones de comparación y salto. En este último caso dispone de instrucciones de salto condicional como incondicional. Todas las instrucciones MIPS tienen el mismo tamaño (32 bits). Se pueden clasificar en función de los elementos que utiliza la instrucción se debe especificar en una serie de bits. Los distintos tipos de instrucción constan de diferentes tamaños para los espacios reservados para esos bits (campos) es decir, utilizan diferentes formatos para codificar sus campos. Por lo que se tienen tres tipos de formatos de instrucción.

- Formato R o de registro.
- Formato I o de tipo inmediato.
- Formato J o de salto incondicional.

1.5.1 Formato de instrucciones MIPS

Para entrar de lleno en el formato de instrucciones MIPS veamos un ejemplo de instrucción representada simbólicamente: add $t0, $s1, $s2 en binario como se muestra en la figura 6.

Esta instrucción se representa en el lenguaje MIPS como campos de números binarios de la siguiente forma:

000000	10001	10010	01000	00000	100000

Figura 6: Representación de instrucción MIPS en binario 32 bits

Que en decimal se corresponde a lo mostrado en la figura 7.

000000	17	18	8	0	32

Figura 7: Representación de instrucción MIPS en decimal 32 bits.

Esta distribución de la instrucción se denomina formato de instrucción. Es importante saber que en MIPS, el compromiso elegido por los diseñadores de la arquitectura fue el de guardar todas las instrucciones con la misma longitud. Como consecuencia de ello, el número de bits de una instrucción MIPS siempre es de 32 bits, el mismo tamaño que el de una palabra. A los campos MIPS se les da una serie de nombres para su rápida identificación:

- OP: Operación básica de la instrucción, tradicionalmente llamada código de operación.
- RS: Primer registro operando fuente.
- RT: Segundo registro operando fuente.
- RD: Registro operando destino, donde se almacena el resultado de la operación.
- SHAMT: Tamaño de desplazamiento.
- FUNCT: Función. Este campo selecciona la variante específica de la operación del campo

En MIPS todas las instrucciones se guardan con la misma longitud, pero se da el caso de requerirse diferentes clases de formatos de instrucciones para diferentes clases de instrucción. En MIPS se distinguen tres: tipo R, tipo I y tipo J [9]. Ver figuras 8, 9 y 10.

COD. OP	REGISTRO FUENTE 1	REGISTRO FUENTE 2	REGISTRO DESTINO		FUNCT
XXXXXX	RS	RT	RD	SHAMT	FUNCT
6	5	5	5	5	6
31-26	25-21	20-16	15-11	10-6	5-0

Figura 8: Formato tipo R, utilizado por instrucciones aritméticas y lógicas.

COD. OP	REGISTRO BASE	REGISTRO DESTINO	DESPLAZAMINETO
XXXXXX	RS	RT	INMEDIATO
6	5	5	16
31-26	25-21	20-16	15-0

Figura 9: formato tipo I, utilizado por las instrucciones de transferencia, las de salto condicional y las instrucciones con operandos inmediatos.

COD. OP	Dirección destino
XXXXXX	Dirección
6	26
31-26	25-0

Figura 10: Formato tipo J, utilizado por las instrucciones de salto incondicional.

Los formatos se distinguen por el valor del primer campo: a cada formato se le asigna un conjunto de valores distintos en el primer campo y por lo tanto la circuitería sabe si ha de tratar la mitad de la instrucción como tres campos, es decir como tipo R, o como un campo simple tipo I, o si la instrucción es tipo J.

1.5.2 Juego de instrucciones MIPS

El conjunto de instrucciones que especificaremos permite realizar operaciones de carga y almacenamiento desde y hacia a memoria, tendrá capacidad de desarrollar programas aritméticos en punto flotante de simple precisión y ofrecerá la posibilidad de controlar el flujo de la ejecución del programa mediante instrucciones de comparación y salto, tanto condicionales, como incondicionales. En resumen tendremos: Instrucciones aritméticas para punto flotante, instrucciones aritméticas para números enteros, instrucciones de carga/almacenamiento (o transferencia), instrucciones de comparación, instrucciones de salto condicional e instrucciones de salto incondicional. En la tabla 4 se muestran las instrucciones MIPS que se implementaran en el microprocesador a desarrollar.

Tabla 4: Lista de instrucciones MIPS a utilizar.

No.	INST.	FUNCIÓN.	TIPO
1	**MOV**	Mueve el contenido de un registro a otro.	I
2	**MOVC**	Carga una constante que se encuentra en la siguiente dirección de la memoria ROM.	I
3	**MOVRX**	Asigna el contenido de una localidad de memoria de la RAM a un registro.	I
4	**MOVWX**	Guarda el contenido de un registro a una localidad de memoria de la RAM.	I
5	**RIO**	Asigna el contenido de un puerto de entrada a un registro.	I
6	**WIO**	Asigna el contenido de un registro a un puerto de salida.	I
7	**ADD.S**	Suma el contenido de dos registros en punto flotante de simple precisión formato IEEE 754 y almacena el resultado en un tercer registro.	R
8	**SUB.S**	Resta el contenido de dos registros en punto flotante de simple precisión formato IEEE 754 y almacena el resultado en un tercer registro.	R
9	**MUL.S**	Multiplica el contenido de dos registros en punto flotante de simple precisión formato IEEE 754 y almacena el resultado en un tercer registro.	R
10	**DIV.S**	Obtiene el inverso del contenido de un registro en punto flotante de simple precisión formato IEEE 754 y almacena el contenido en un segundo registro.	R
11	**ADD.I**	Suma el contenido de dos registros en entero y almacena el resultado en un tercer registro	R
12	**SUB.I**	Resta el contenido de dos registros y almacena el resultado en	R

		un tercer registro.	
13	**C.EQ.S**	Compara el contenido de dos registros en punto flotante de simple precisión con formato IEEE 754, si son iguales se salta a una instrucción definida, si no son iguales se sigue con la siguiente instrucción.	I
14	**C.LE.S**	Compara el contenido de dos registros en punto flotante de simple precisión con formato IEEE 754, si un registro es menor o igual que otro, se salta a una instrucción definida, si no son iguales se sigue con la siguiente instrucción.	I
15	**C.LT.S**	Compara el contenido de dos registros en punto flotante de simple precisión con formato IEEE 754, si un registro es menor que otro, se salta a una instrucción definida, si no son iguales se sigue con la siguiente instrucción.	I
16	**JUMP**	Permite el salto a cualquier localidad de la memoria de programa	J

1.6 Unidad Aritmética

Para este trabajo de tesis se diseña una unidad aritmética haciendo uso de módulos de propiedad intelectual útiles para operaciones de coma flotante que proporciona el fabricante del FPGA de Altera [10], estos módulos operan con números de punto flotante de simple y doble precisión bajo el estándar IEEE-754, para el caso de este trabajo de tesis se opera únicamente con números de punto flotante de simple precisión (32 bits). A continuación se expone sobre la representación numérica de punto flotante, el estándar IEEE-754 y la lógica aritmética de punto flotante para realizar la suma, resta multiplicación y división.

1.6.1 Punto flotante

Existen varias formas de representar números no enteros. Una de ellas es usando un punto fijo. Este tipo de representación ubica siempre un punto o como en alguna posición a la derecha del digito menos significativo (ver tabla 5). Otra alternativa común mente usada es la que se le conoce como representación en punto flotante. Los números de punto flotante decimales normalmente se expresan en notación científica con un punto explicito siempre entre el primer y segundo digito. El exponente se escribe explícitamente incluyendo la base, o se usa una "e" por separado de la mantisa.

Tabla 5: Ejemplos de punto flotante decimal expresado en notación científica.

Mantisa	Exponente	Notación científica	Valor en punto fijo
1.5	4	$1.5\ x\ 10^{4}$	15000
-2.001	2	$-2.001\ \text{x}\ 10^{2}$	-200.1
5	-3	$5\ x\ 10^{-3}$	0.005
6.667	-11	$6.667\ \text{x}\ e^{-11}$	0.0000000000667

Dado que un número en punto flotante puede expresarse de distintas formas que son equivalentes, es necesario establecer una única representación. Es por ello que se trabaja con números normalizados. Decimos que un número esta normalizado si el digito a la izquierda del punto está entre 0 y la base (0<digito a la izquierda del punto<b)

$1.00x10^2$ Normalizado (1)

$0.01x10^2$ Des normalizado (2)

En particular, decimos que un número binario esta normalizado si el digito a la izquierda del punto es igual a 1.

1.6.2 Formato de punto flotante de simple precisión IEEE-754

El estándar IEEE-754 es un estándar técnico para el cálculo de punto flotante, define los formatos aritméticos, los formatos de intercambio, las reglas de redondeo, las operaciones y el manejo de excepciones [11]. El formato con punto flotante de simple precisión se representa en la aritmética binaria con un bit de signo (31), (Sx), 8 bits de exponente (30 al 23), (Ex) y 23 bits de mantisa (22-0), (Mx) como se muestra en la ecuación 3. Para obtener una mayor precisión, se omite el bit correspondiente a la parte entera de la mantisa y se supone "1" cuando es un número normal y "0" cuando es un subnormal obteniendo una precisión real de 24 bits de mantisa. Ver figura 11.

$$(-1)^{S_x} \text{x } M_x \text{ x } 2^{E_x} \quad (3)$$

31	30		23	22		0
Signo	Exponente			Mantisa		
1	8 bits			23 bits		

Figura 11: Representación de punto flotante de simple precisión.

Para representar el exponente en punto flotante se utiliza una representación en exceso N de forma que el exponente más negativo posible para simple precisión, quede en 0000 0001 y el más grande de los positivos en 1111 1110 . El estándar IEEE 754 usa como exceso 127 en simple precisión, el exponente está situado en un rango de -126 a +127 y es desplazado mediante la suma de 127 para obtener un valor en el rango 1 a 254.

Para valores de exponente desde 1 hasta 254, se representan números de punto fijo normalizados. El exponente esta en exceso, siendo el rango del exponente de -126 a +127. Un número normalizado debe contener un 1 bit a la izquierda del punto binario; este bit está implícito, dando una mantisa efectiva de 24 bits. Un exponente cero junto con una parte fraccionaria cero representa el cero positivo o negativo, dependiendo del bit de signo. Ver tabla 6 representaciones de tipo de datos.

Tabla 6: Representación de tipo de datos de punto flotante de simple precisión.

Tipo de dato	Signo	Exponente	Mantisa
Cero positivo	0	E=0	M=0
Cero negativo	1	E=0	M=0
Subnormal	S	E=0	M>0
Normal	S	0<E<255	M=X
Infinito	S	E=255	M=0
No un número (NaN)	S	E=255	M>0

La representación de punto flotante tiene un número de dígitos limitado, por consiguiente no es posible representar todos los reales de forma precisa: cuando hay más dígitos de los que permite el formato, los que sobran se omiten, y se redondean. El método de redondeo por defecto del estándar IEEE 754 es el redondeo mitad al par: si la fracción truncada es mayor que la mitad de la base, se incrementa el último digito. Si es igual a la mitad de la base, se incrementa solo si el resultado es par. Esto minimiza el error y el sesgo.

1.6.3 Aritmética en punto flotante

Suma y Resta: Para realizar la suma y resta de dos operandos en representación de punto flotante se realiza previamente la separación de los exponentes y de las mantisas para su posterior manejo y después realizar los siguientes pasos [12]:

1. Seleccionar el número con menor exponente y desplazar su mantisa a la derecha cuantas veces lo indique la diferencia en módulo de los exponentes.
2. Hacer que el exponente resultado sea igual al mayor de los exponentes.
3. Realizar las operaciones de suma o resta con las mantisas.
4. Normalización del resultado. Una vez realizada la suma se debe normalizar desplazando los bits a la a mantisa hacia la izquierda o la derecha con la cual habrá que cambiar el valor del exponente.
5. Comprobar las condiciones de rebose.

Multiplicación y división: La multiplicación y la división en punto flotante son más sencillas de realizar, los pasos que nos permitirán realizar estas operaciones son las siguientes [12]:

1. Realizar la suma de los exponentes.
2. Multiplicar/dividir las mantisas y determinar el signo del resultado.
3. Normalizar el resultado, si es necesario.
4. Comprobar las condiciones de rebose.

En la unidad aritmética a desarrollar se implementan las operaciones básicas ya mencionadas, pero dependiendo de la aplicación del procesador se pueden agregar módulos IP para el cálculo de funciones trigonométricas, raíz cuadrada, exponenciales, etc.

1.7 Conclusiones

Al principio del capítulo se presentaron las tarjetas basadas en FPGA con las que actualmente cuenta la MCEA, las cuales ofrecen un limitado número de recursos lógicos, que impide la evolución e implementación de algoritmos de mayor complejidad, con el sistema embebido propuesto se ofrece un FPGA con el doble de recursos lógicos, un microprocesador embebido, memorias externas y una interfaz inalámbrica WiFi, todo en una sola tarjeta que además es compatible con las ya existentes, esto permitirá la actualización y uso de la infraestructura con la que ya se cuenta.

Después de haber realizado un análisis y descripción de requerimientos para el sistema embebido a desarrollar pasamos a la selección de componentes y diseño de la tarjeta electrónica (hardware).

Capítulo 2. Hardware

EL Hardware es la parte física y visible de un sistema de cómputo. Esto puede ser un conjunto de circuitos electrónicos y quizá algunos componentes mecánicos. En pocas palabras, está hecho de material y es tangible. En este capítulo se presenta el diseño electrónico y el diseño del circuito impreso (PCB, Printed Circuit Board) del sistema embebido propuesto, para eso se utilizó el software de diseño Altium Designer, debido a la facilidad que ofrece para la creación de nuevas librerías.

2.1 Diseño electrónico

Una vez revisados los requerimientos y características del sistema propuesto, se procede a la selección de componentes y al diseño electrónico del sistema embebido. La figura 12 muestra la constitución por componentes del sistema:

1. FPGA: Circuito electrónico en el que se implementa un microprocesador de 32 bits, este componente está constituido por bloques de entrada/salida (1a), bucles de enganche de fase (Phase-Locked Loops, PLLs) (1b), pines reloj dedicados (1c), bloque de programación (1d), bloque de voltajes de alimentación (1e) y bloques de conexión a tierra (1f).
2. Circuito de suministro de energía: Circuito conformado, por un conector para alimentación externa de la tarjeta (2a) y reguladores de voltaje DC (2b), para la generación de voltajes que requiere el FPGA, y cada uno de los circuitos que conforman la tarjeta electrónica.
3. Circuito reset: Este circuito tiene la finalidad de mantener al FPGA en estado de reset (3a) hasta que los niveles de suministro de energía se hayan estabilizado durante el encendido del dispositivo y la de reiniciar el dispositivo.
4. Circuito de reloj: La tarjeta incluye un oscilador de 100 MHz (4a). El oscilador es conectado directamente a un pin de reloj dedicado del FPGA. Esta señal de reloj puede ser ocupada como una fuente de reloj o manejase con un PLL.
5. Circuito de memoria externas: Este circuito se compone de una memoria del tipo RAM y ROM síncronas (5a), las cuales se comunican con el FPGA por medio de un protocolo de comunicación SPI (Serial Peripheral Interface), la memoria ROM se destina para el almacenamiento del conjunto de instrucciones que interpreta y ejecuta el microprocesador y la RAM al almacenamiento de datos a procesar.
6. Módulo WiFi: Le corresponde la iteración del usuario con el sistema, no solamente como visualización de la información obtenida si no como el periférico de entrada y salida de datos: Se eligió un módulo de comunicación inalámbrica WiFi (6a) para la comunicación con una PC con el fin de brindar al usuario un entorno grafico versátil e intuitivo.
7. Circuito de distribución de entrada/salidas y relojes: La tarjeta cuenta con 4 conectores (header) macho de 50 pines (7a) que conectan directamente con las terminales I/O y relojes del FPGA, además proporcionan terminales de alimentación VCC y pines a GND.

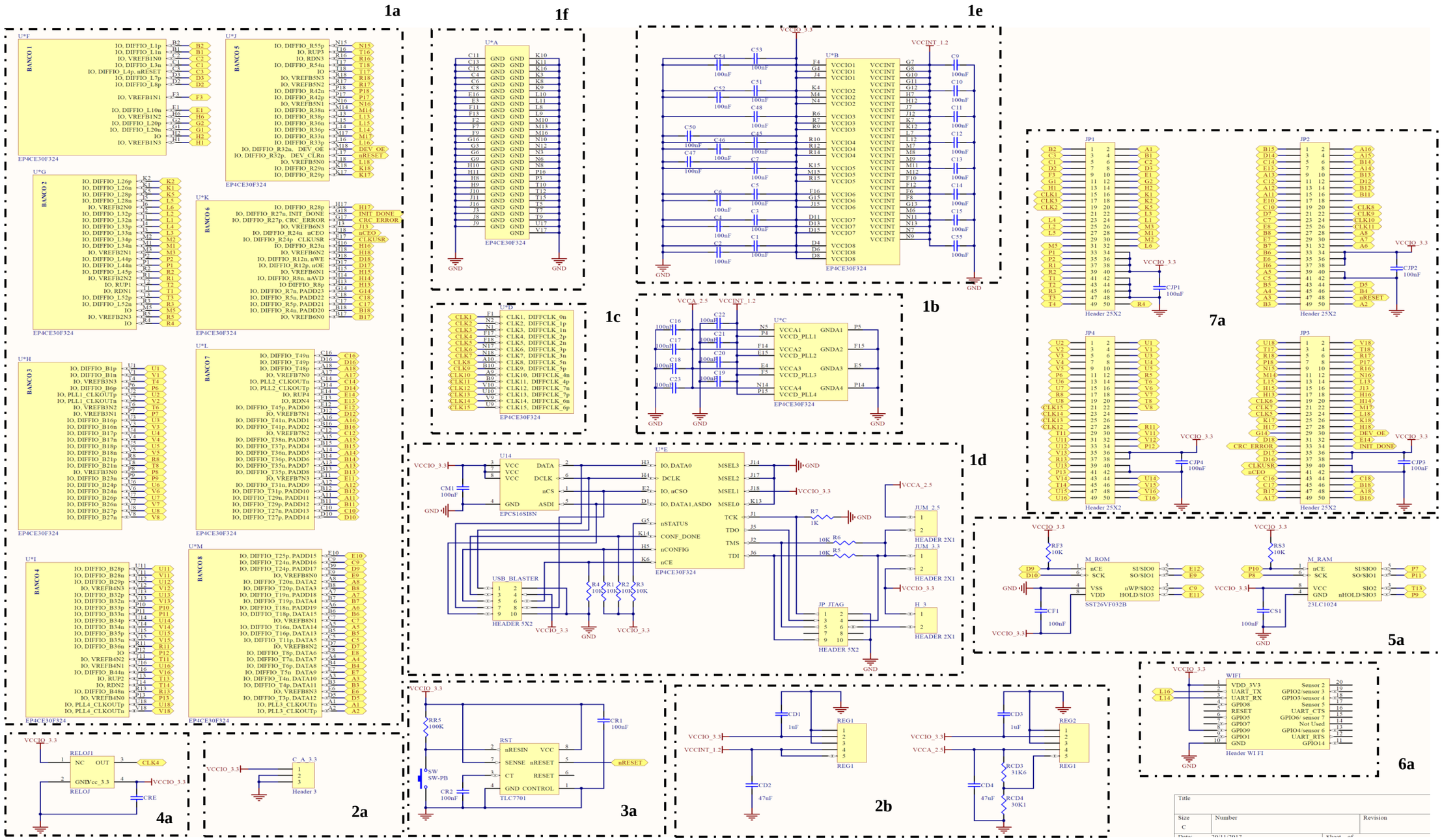


Figura 12: Diseño electrónico de tarjeta electrónica con FPGA, RAM y ROM externas.

El desarrollo de un sistema embebido usando un FPGA, nos permite realizar la solución de ingeniería en una sola pastilla SOC (System On a Chip), con las ventajas de: utilizar menos espacio sobre la tarjeta, menores requerimientos de potencia, procesos de ensamble más rápidos, menos costosos y de mayor confiabilidad, dado que se tiene menos circuitos integrados, así como menos conexiones de circuito donde puedan ocurrir fallas. El diseño de una tarjeta, además de la selección de componentes, tiene que ver con las emisiones electromagnéticas que pueden afectar el correcto funcionamiento de los dispositivos, por lo que en este trabajo se presentan las técnicas usadas en el diseño de la PCB (Printed Circuit Board), con las cuales se pretende minimizar las RF o EMI, que afectan a los componentes (CI) o a las señales.

2.2 Selección de componentes del sistema embebido

En base a las características del sistema, el hardware es seleccionado con el fin de obtener un balance entre desempeño y funcionalidad en los circuitos integrados. Los circuitos integrados de montaje superficial presentan la posibilidad de ser montados en un plano libre de ruido interno, por lo que se obtiene un mejor funcionamiento de los CI, además de las ventajas de pequeño volumen, fiabilidad, economía y prestaciones. Realizando un análisis del sistema y de las señales necesarias se eligieron los CI.

2.2.1 FPGA

La selección de un FPGA para una aplicación se dificulta usualmente por la existencia de una gran variedad de opciones y características del dispositivo y otros factores de difícil ponderación como son: existencia en el mercado, documentación, apoyo técnico y disponibilidad. Los criterios considerados para determinar el FPGA de la tarjeta son:

- Encapsulado FBGA de 256 pines de preferencia.
- Numero de pines I/O de usuario, entre 160 a 200.
- Disponibilidad y precio por unidad.

El dispositivo idóneo para el desarrollo de este proyecto es el **FPGA EP4CE30F19A7N** de la familia Cyclone IV serie E [13], ya que proporciona mejores características a la tarjeta. En la siguiente lista se muestran las características más importantes de dispositivo seleccionado.

- Encapsulado de FPGA de 19x19 con 324 pines.
- 28,848 elementos lógicos.
- Temperatura de operación, grado automotriz (A).
- Grado de velocidad 7.
- 594 Kbits de memoria integrada
- 66 multiplicadores integrados de 18x18
- 193 pines I/O.
- 4 PLL.

Ya seleccionado el FPGA, procedemos a conocer las condiciones de operación optimas del dispositivos de acuerdo a las especificaciones que marca el fabricante. Los parámetros que determinan las condiciones de operación son el grado de velocidad que influye en el rendimiento y consumo de energía. Para conocer dichos parámetros hacemos uso de la matrícula del FPGA seleccionado (EP4CE30F19A7N), el cual se desglosa de la siguiente manera:

Dónde: **EP4C-E-30-F-19-A-7-N**

- **EP4C**: Indica que la familia Cyclone IV.
- **E**: Indica a sub-familia.
- **30**: Indica el número de elementos lógicos (LEs).
- **F**. Indica el encapsulado FBGA (19x19).
- **19**: Indica los pines del dispositivo (324 pines).
- **A**: Temperatura de operación. (Grado Automotriz -40ºc a 125ºc.)
- **7**: Grado de velocidad.
- **N**: indica que el dispositivo está libre de plomo.

Los dispositivos Cyclone IV están disponibles en grado comercial (C), industrial (I), y automotriz (A). Como también en los grados de velocidad -6 (el más veloz), -7, -8, -8L, y -9L para los dispositivos comerciales, -8L para dispositivos industriales y -7 para dispositivos automotrices. Los dispositivos Cyclone IV, con grado de velocidad A7 operan con un "**core voltaje**" (Voltaje de alimentación para la lógica interna) de 1.2 VDC. En la tabla 7 se muestra las condiciones de operación recomendados (TIPO) para dispositivos Cyclone IV de acuerdo al fabricante para los parámetros de voltaje y temperatura, así como el valor mínimo y máximo aceptable de cada uno

Tabla 7: Condiciones de operación recomendadas para dispositivos Cyclone IV E con grado de velocidad A7.

SIMBOLO	PARÁMETRO	MIN	TIPO	MAX	Ud.
VCCINT	Voltaje de alimentación para la lógica interna, 1.2V.	1.15	1.2	1.25	V
VCCIO	Voltaje de alimentación para buffers de salida, 3.3V.	3.135	3.3	3.465	V
VCCA	Voltaje de alimentación (analógico) para regulador PLL.	2.375	2.5	2.625	V
VCCD_PLL	Voltaje de alimentación (digital) para PLL.	1.15	1.2	1.25	V
V_I	Voltaje de entrada.	-0.5	-	3.6	V
V_O	Voltaje de salida.	0	-	VCCIO	V
T_J	Temperaturas de operación.	-40		125	ºC
t_{RAMP}	STANDARD Power-On-Reset (POR)	50us		50ms	S
t_{RAMP}	FAST POR	50us		3ms	S

Ya que se conocen los voltajes de operación recomendados por el fabricante, se procede a diseñar y seleccionar los componentes para el circuito de suministro de energía, pero antes se revisa el tema del circuito de configuración del FPGA.

2.2.2 Memoria de configuración

En los dispositivos FPGA la tecnología utilizada para los recursos configurables son de tecnología RAM. La utilización de tecnología RAM permite la implementación de dispositivos FPGA de muy alta escala de integración. Como contrapartida, la tecnología RAM de los dispositivos FPGA pierde su configuración cada vez que el dispositivo se des energiza y debe ser reconfigurado cada vez que el dispositivo FPGA entra el funcionamiento. Para este propósito, cuando se utilizan dispositivos FPGA, se disponen conjuntamente con una memoria flash en la cual se almacena la configuración que el dispositivo FPGA lee automáticamente cuando es energizado para auto configurarse. Los esquemas de configuración con los que puede ser configurado el dispositivo Cyclone IV, por medio de un dispositivo de memoria, son los siguientes:

- Active serial (AS).
- Active parallel (AP).

Active serial: En este esquema de configuración los dispositivos Cyclone IV son configurados con dispositivos de configuración serial. Estos dispositivos de configuración con memoria no volátil son de bajo costo, cuentan con una sencilla interfaz de cuatro pines y dimensiones reducidas. Estas características hacen de estos dispositivos de configuración una solución ideal de bajo costo.

Active parallel (AP): El esquema de configuración AP solo está disponible para dispositivos Cyclone IV E. En el esquema de configuración AP, los dispositivos Cyclone IV E, son configurados usando memorias flash paralelas. Estos dispositivos de configuración con memoria no volátil externas proporcionan una rápida interfaz para acceder a datos de configuración. Como se ha mencionado, el esquema Active parallel, utiliza memorias flash paralelas para la configuración de dispositivos, por lo que involucra un mayor número de pines para su conexión, contrario al esquema de configuración AS, que utiliza dispositivos de configuración que solo requieren de cuatro pines para su conexión. Dado a que el FPGA seleccionado no cuenta con bastantes pines como para conectar una memoria paralela y en la tarjeta no se dispone del espacio suficiente para un dispositivo de este tipo, se considera idóneo utilizar el esquema de configuración AS. A continuación se muestra en la tabla 8 una lista de selección de memorias de configuración serial para distintos dispositivo Cyclone IV.

La memoria de configuración serial que le corresponde al FPGA seleccionado es una memoria con 16 Mbits de almacenamiento EPCS16 [14], en la tabla 8 se mencionan algunas de sus características principales.

- Compatible con esquema de configuración (AS) x1.
- Voltaje de alimentación de 2.7V a 3.6 V.
- Dispositivo de pocos pines, de bajo costo y memoria no volátil.
- Fácil interfaz de cuatro pines.
- Encapsulado de 8 pines (SOIC 8)

Tabla 8: Memorias de configuración serial para FPGAs Cyclone IV.

FPGA	Numero de bits de configuración.	Dispositivo de configuración serial
EP4CE10	2,944,088	EPCS4
EP4CE15	4,086,848	EPCS4
EP4CE30	**9,534,304**	**EPCS16**
EP4CE40	9,534,304	EPCS16

En la tabla 9 se muestra la organización de las memorias de configuración serial.

Tabla 9: Organización de memoria de los dispositivos de configuración serial.

Details	EPCS128	EPCS64	EPCS16	EPCS4	EPCS1
Bytes	16,777,216 bytes (128 Mb)	8,388,608 bytes (64 Mb)	2,097,152 bytes (16 Mb)	524,288 bytes (4 Mb)	131,072 bytes (1 Mb)
Number of sectors	64	128	32	8	4
Bytes per sector	262,144 bytes (2 Mb)	65,536 bytes (512 Kb)	65,536 bytes (512 Kb)	65,536 bytes (512 Kb)	32,768 bytes (256 Kb)
Pages per sector	1,024	256	256	256	128
Total number of pages	65,536	32,768	8,192	2,048	512
Bytes per page	256 bytes	256 bytes	256 bytes	256 bytes	256 bytes

Como se mencionó estos dispositivos utilizan una sencilla interfaz de cuatro pines que son los siguientes:

- Serial Data output (DATA)
- AS data input (ASDI)
- Serial clock (DCLK)
- Chip select (nCS)

La comunicación que se establece entre la memoria de configuración y el FPGA, se realiza por medio del protocolo de comunicación SPI, que requiere de dos señales de control (selección y reloj) y uno para enviar (DATA) y otro para recibir datos (ASDI). A continuación en la figura 13 se muestra el circuito que se implementar en el sistema embebido a desarrollar. Que ofrece la posibilidad de programar al FPGA por un puerto JTAG, y programar la memoria de configuración por medio de un puerto compatible con el programador usb-blaster.

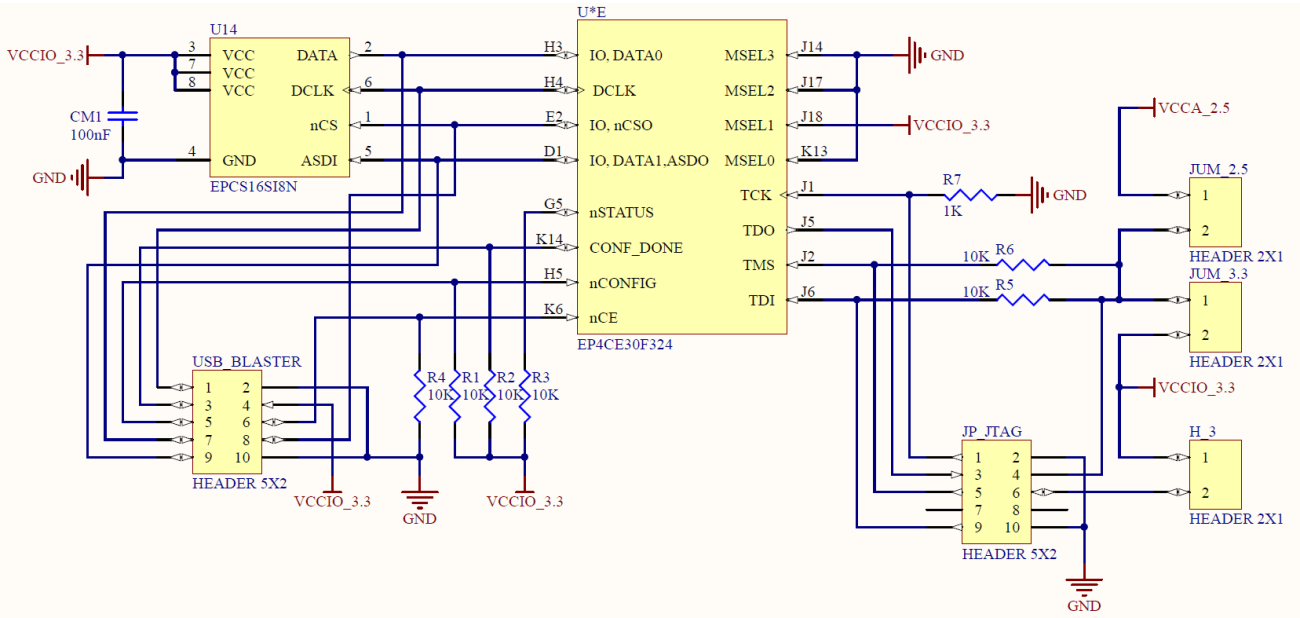


Figura 13: Esquema de configuración que combina JTAG y AS.

En la figura 13 se observan las conexiones entre la memoria de configuración **EPCS16** y los pines de programación del FPGA. Este circuito permite la configuración del FPGA por JTAG o la configuración por usb-blaster de la memoria de configuración dado a que se eligió una memoria de configuración serial compatible con el esquema de configuración AS entonces, elegimos un retardo POR (Power-On Reset) estándar, con MSEL0=GND, MSEL1=VCCIO, MSEL2=GND, y MSEL3=GND, tal como se muestra en la tabla 10.

Tabla 10: Esquemas de configuración para dispositivos Cyclone IV E.

Configuration Scheme	MSEL3	MSEL2	MSEL1	MSEL0	POR Delay	Configuration Voltage Standard (V) (1)
AS	1	1	0	1	Fast	3.3
	0	1	0	0	Fast	3.0, 2.5
	0	0	1	0	Standard	3.3
	0	0	1	1	Standard	3.0, 2.5
AP	0	1	0	1	Fast	3.3
	0	1	1	0	Fast	1.8
	0	1	1	1	Standard	3.3
	1	0	1	1	Standard	3.0, 2.5
	1	0	0	0	Standard	1.8

2.2.3 Circuito de suministro de energía

El FPGA requiere de distintas fuentes de alimentación externas, un voltaje a 3.3V para alimentar los bancos I/O (VCCIO), 2.5V para la alimentación analógica de los PLL (VCCA) y 1.2V para la alimentación digital de los PLL (VCCD_PLL) y alimentación de la lógica interna del dispositivo (VCCINT), los cuales deberán ser generados a partir de una

única fuente de alimentación externa de 3.3V DC, utilizando el circuito TPS75501KTT que es un regulador de voltaje lineal DC, que puede regular el voltaje de salida aun cuando el voltaje de alimentación está muy cercano al de salida. A partir de estos datos se genera el diagrama a bloques de la figura 14 que describe al sistema de distribución de energía que se implementara a la nueva tarjeta.

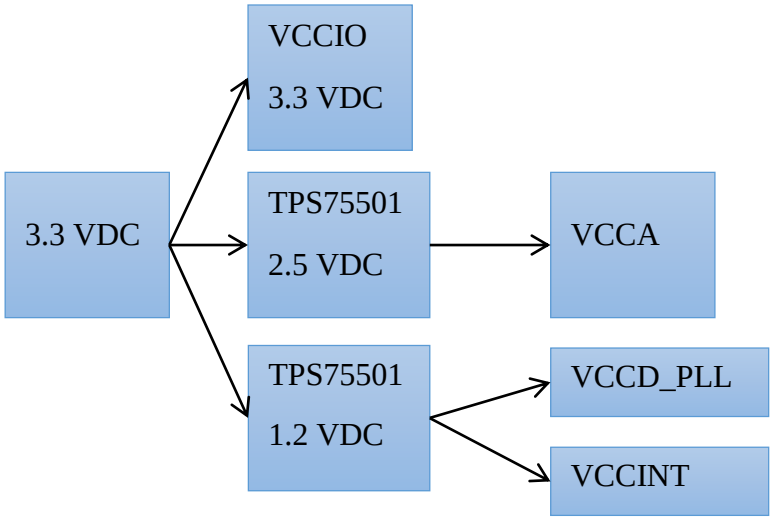
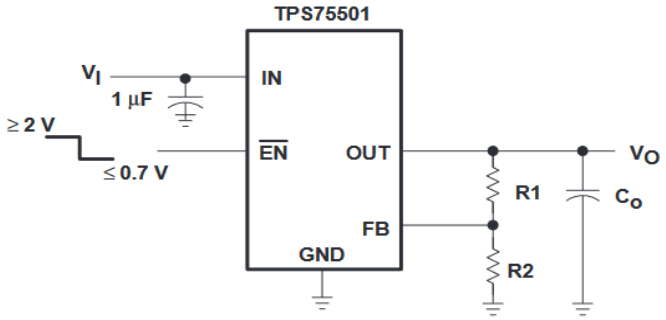


Figura 14: Sistema de distribución de energía a implementar en diseño de tarjeta

Como se observa en la figura 14, el sistema de distribución de energía se compone de dos reguladores de voltaje lineal, que proporcionan 2.5V DC y 1.2V DC respectivamente a partir de un voltaje de entrada de 3.3V DC. Las características más importantes del regulado TPS75501 son [15]:

- Es un regulador de voltaje Low Dropout.
- Voltaje de salida ajustable de 1.22V a 5V DC.
- Rápida respuesta transitoria.
- Encapsulado TO-220 de 5 pines.
- Protección de apagado térmico.

Para obtener los voltajes de 2.5V y 1.2V se utilizó la configuración de ajuste de voltaje, donde el voltaje de salida se programa por medio de unas resistencias externas R1 y R2, tal como se muestra en el siguiente circuito de la figura 15.



OUTPUT VOLTAGE PROGRAMMING GUIDE

OUTPUT VOLTAGE	R1	R2	UNIT
2.5 V	31.6	30.1	kΩ
3.3 V	51	30.1	kΩ
3.6 V	58.3	30.1	kΩ

Figura 15: TPS75501 en configuración de voltaje de salida ajustable.

El voltaje de salida se calcula mediante la siguiente formula:

$$V_o = V_{ref}\left(1 + \frac{R1}{R2}\right) \tag{4}$$

Donde Vref = 1.224V (voltaje de referencia interno). El procedimiento de diseño recomendado es elegir a R2=30.1K, para fijar una corriente en el divisor de 40µA y calcular R1, por medio de la siguiente ecuación.

$$R1 = \left(\frac{V_o}{V_{ref}} - 1\right) R2 \tag{5}$$

Para obtener el voltaje de 1.2V se conecta directamente el pin FB a OUT (ver figura 9), y para obtener 2.5V se toma a R2=30.1KΩ y de acuerdo a la ecuación (2), R1 queda como 31.6 KΩ. El capacitor de entrada es de 1µF y el de salida como es de 47 µF como mínimo tal como se muestra en el diagrama eléctrico de la figura 16. Con esto terminamos de definir lo elementos que conforman al sistema de distribución de energía de la nueva tarjeta a desarrollar. Por lo que continuamos con la selección de elementos que integran el circuito de reset.

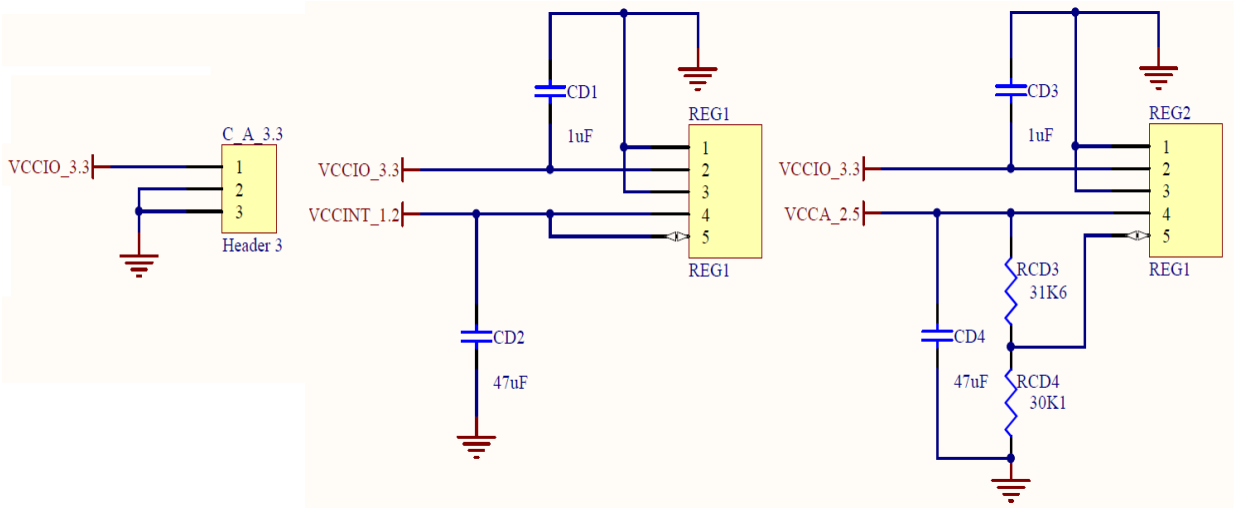


Figura 16: Diagrama eléctrico de circuito de suministro de energía.

En la figura 16 tenemos a la izquierda la conexión a GND y 3.3.VDC, que tomamos del exterior mediante una fuente regulada y que se distribuyen directamente a la tierra del FPGA y a la alimentación de los bancos I/O (VCCIO), al centro tenemos un regulador de voltaje que nos proporciona 1.2VDC para la alimentación de la lógica interna del FGPA VCCINT, y a la derecha se muestra el regulador de voltaje que nos proporciona a la salida el voltaje de 2.5V para la alimentación analógica de los PLL (VCCA). Las conexiones del FPGA a los tres voltajes y GND disponibles en la tarjeta se muestran en la figura 17 y 18.

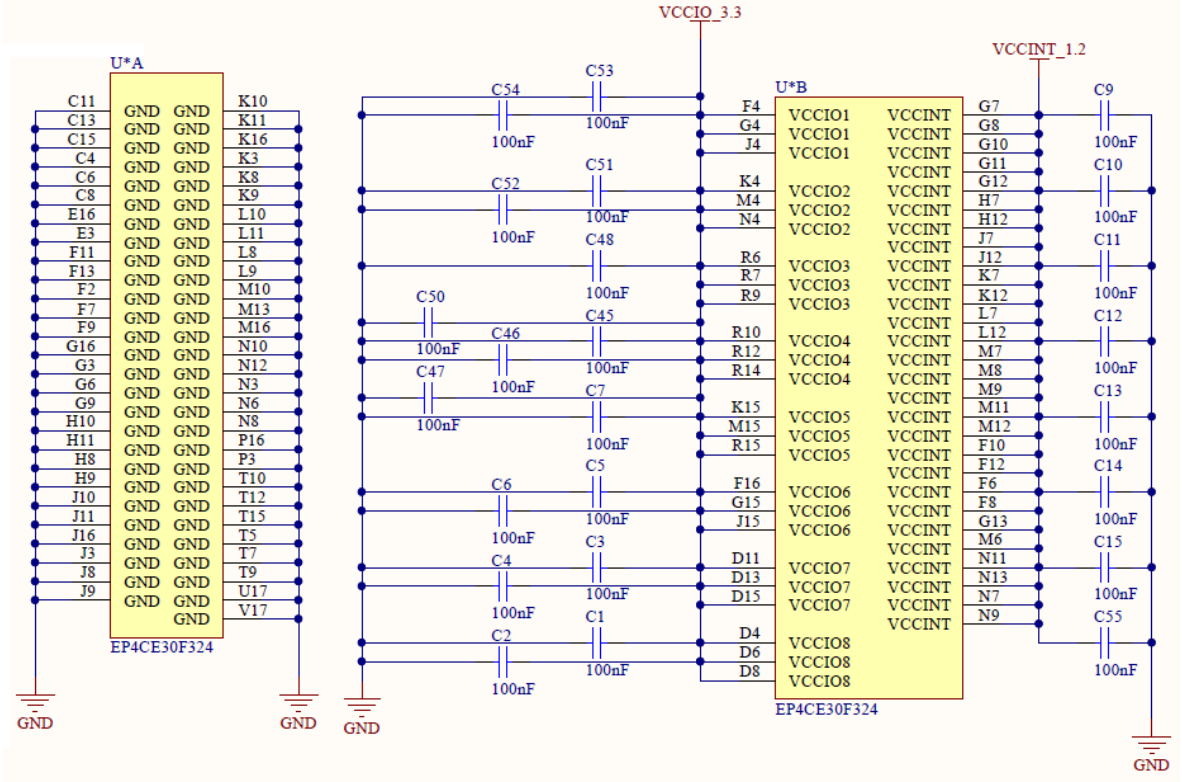


Figura 17: Circuito eléctrico para GND, VCCIO y VCCINT del FPGA.

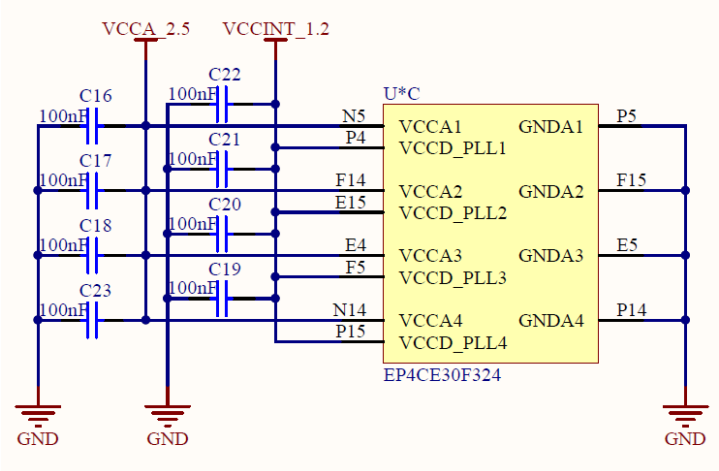


Figura 18: Circuito eléctrico para VCCA y VCCD del FPGA.

2.2.4 Circuito de reset

El circuito de reset o circuito Power-On (POR), mantiene el dispositivo en estado de reset hasta que los niveles de suministro de energía se hayan estabilizado durante el encendido del dispositivo, el circuito que se considera es el TLC7701 [16]. A continuación se mencionan las principales características del dispositivo.

- Generador POR.
- Generador un reset automático, tras la caída de voltaje.
- Sensor de voltaje de precisión.
- Voltaje de alimentación de 2V a 6V.
- Retardo programable por capacitor externo.
- Máxima corriente de alimentación de 16µS.

En la figura 19 se muestra un diagrama de tiempo de la respuesta del circuito de reset, ante las variaciones del voltaje de alimentación, así como el retardo td.

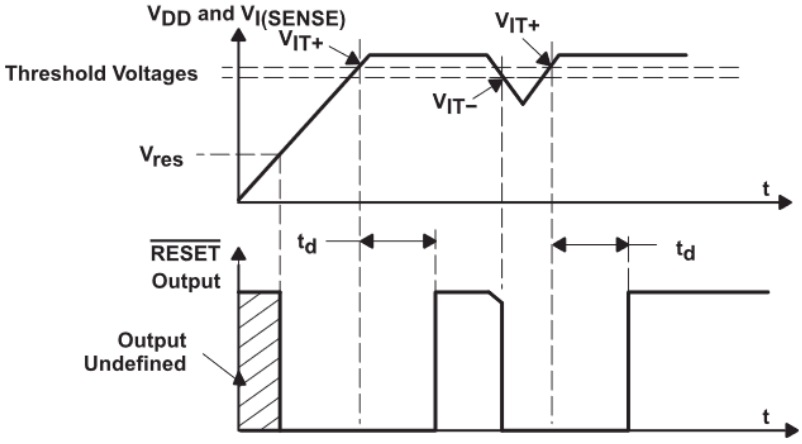


Figura 19: Diagrama de tiempos del circuito de reset.

Como en se observa en la figura 19, VDD es el voltaje a monitorear (SENSE), VIT+ es VDD y VIT- es VDD menos el voltaje de umbral que para el dispositivo TLC7701ID es de 1.1V. Sobre el funcionamiento del circuito podemos que en el instante en que VDD es igual o mayor a VIT+ se inicia un retardo td, y una vez que termina se manda a reset el dispositivo controlado (FPGA o microcontrolador, etc.). En la figura 20 se muestra la configuración utilizada para el control de reset de nuestro FPGA, se puede observar las resistencias, capacitores e interruptor que requiere el circuito de reset, para su correcto funcionamiento como controlador de reset en un microordenador. El valor del capacitor conectado a la terminal determina el tiempo de retardo td que se calcula mediante la siguiente fórmula.

$$td = 2.1 \times 10^{4} \times CT \quad (6)$$

Donde CT está en faradios y td en segundos.

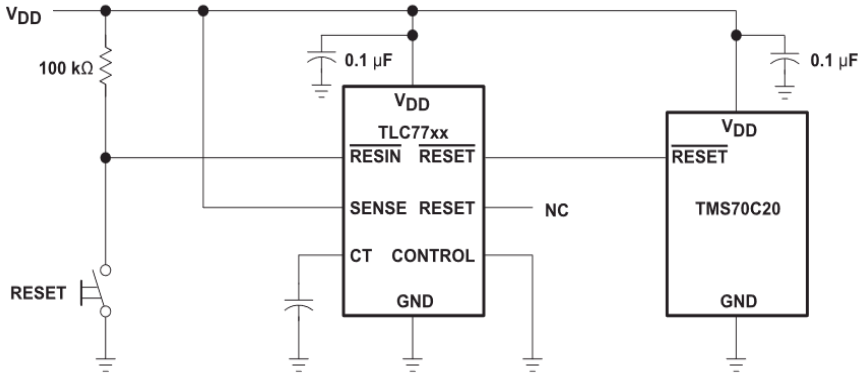


Figura 20: Controlador de reset en un sistema microordenador

En la figura 21 se puede ver la implementación de capacitores Bvpass, de 100 nf, para la eliminación de ruido en las líneas de alimentación, que se genera por la carga y descarga de los capacitores internos de los circuitos.

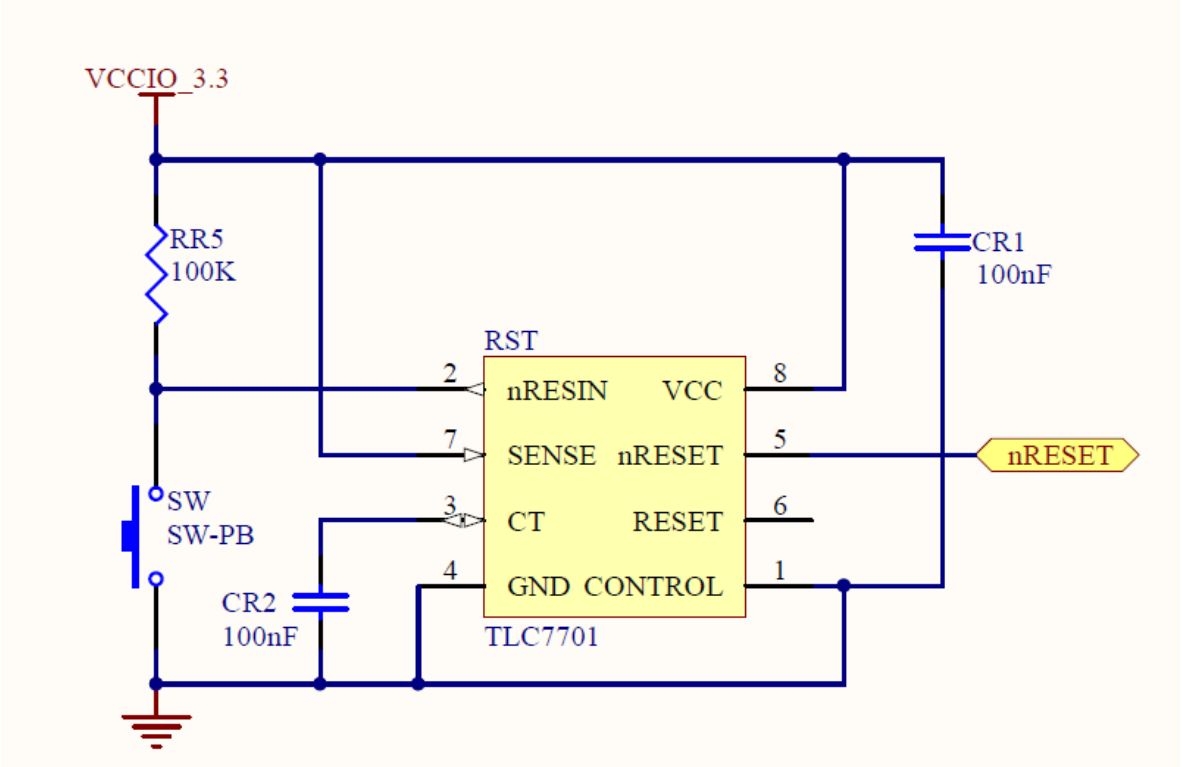


Figura 21: Diagrama eléctrico de circuito reset.

La terminal nRESET, de la figura 21, está conectada al banco 5 de entradas/salidas del FPGA y a la terminal 48 del conector 2 (header de 50 pines).

2.2.5 Circuito de reloj

La tarjeta incluye un oscilador de 100 MHz (4a). El oscilador es conectado directamente a un pin de reloj dedicado del FPGA. Esta señal de reloj puede ser ocupada como una fuente de reloj o manejase con un PLL para la generación de múltiples señales de reloj. En la figura 22 se muestra es circuito eléctrico para el oscilador de cristal, que tiene como alimentación los 3.3V de VCCIO y su salida CLK4 es conectada al reloj 4 del FPGA.

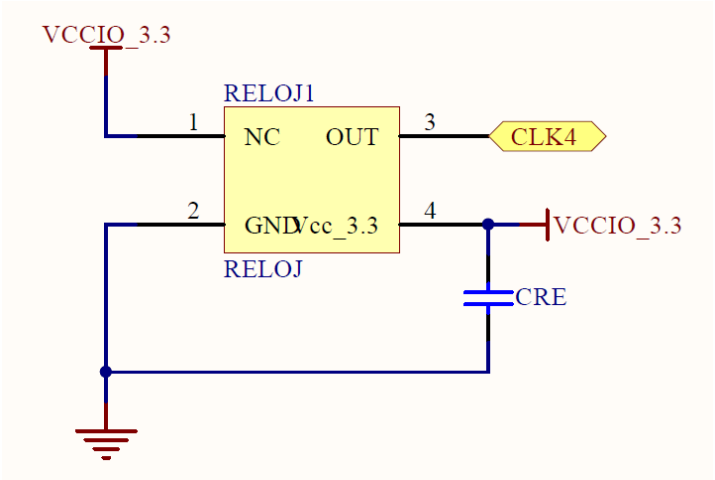


Figura 22: Diagrama eléctrico de circuito de reloj.

El FPGA, seleccionado cuenta con 15 entradas de reloj, estas entradas pueden ser utilizadas para sincronizar al FPGA con dispositivos externos. En la figura 23 se muestra el bloque del FPGA con las conexiones de reloj disponibles.

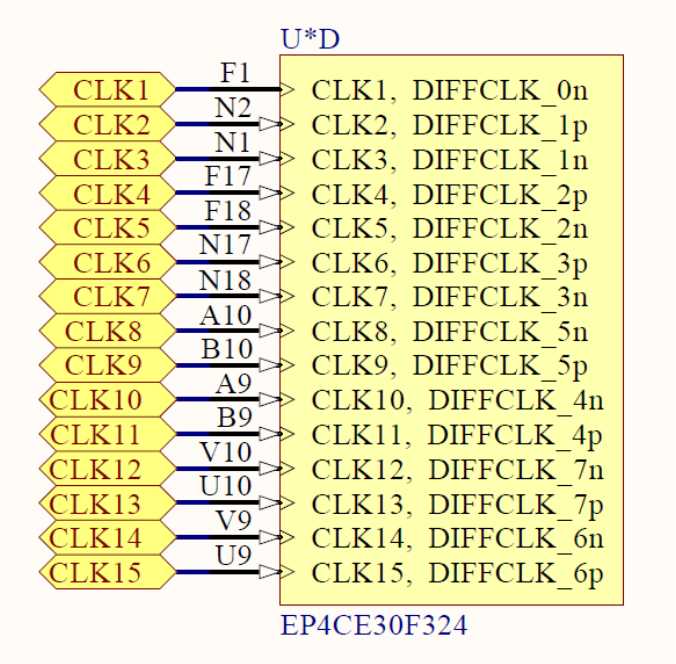


Figura 23: Conexiones relojes (solo entradas) del FPGA.

2.2.6 Circuito de memorias externas

Las memorias RAM y ROM, del sistema embebido deben tener como principales características el encapsulado SOIC-8 y la comunicación serial. Para destinar el mínimo de terminales de entrada/salida del FPGA a la comunicación con las memorias. Para conocer sobre el encapsulado al que nos referimos, en la figura 24, se muestran diferentes encapsulados de memorias EEPROM de comunicación serial de la marca Microchip.

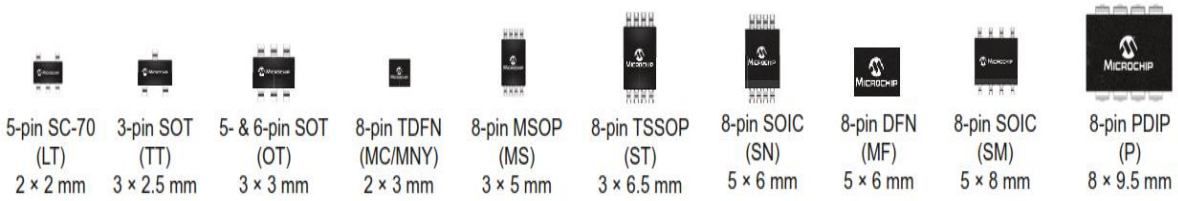


Figura 24: Encapsulados de memorias EEPROM Microchip.

Ya que se conocen los encapsulados disponibles para las memorias de comunicación serial, se desea conocer las el tipo de memorias disponibles que cumplan con los requisitos ya mencionados, para esto consultamos el catálogo de dispositivos electrónicos de "Digi-Key ELECTRONICS" que es uno de los principales proveedores de dispositivos electrónicos a nivel mundial. Al buscar las memorias por el encapsulado "SOIC-8", tenemos como memorias disponibles los siguientes tipos:

- **EEPROM:** Electrically Erasable Programmable Read-Only Memory (ROM programable y borrable eléctricamente). Es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente. Son memorias no volátiles y suelen comunicarse mediante protocolos I^2C, SPI y Microwire.
- **FLASH-NOR:** Derivada de la memoria EEPROM, permite la lectura y escritura de múltiples posiciones de memoria en una misma operación, lo que permite velocidades de funcionamiento muy superiores frente a la tecnología EEPROM. Las basadas en compuertas lógicas NOR permiten la lectura de acceso aleatorio.
- **SRAM:** Static Random Access Memory, que significa memoria estática de acceso aleatorio (o RAM estática), para denominar a un tipo de tecnología de memoria RAM basada en semiconductores, capaz de mantener los datos, mientras siga alimentada, sin necesidad de circuito de refresco. Este concepto surge en oposición al de memoria **DRAM** (RAM dinámica), con la que se denomina al tipo de tecnología RAM basada en condensadores, que sí necesita refresco dinámico de sus cargas.

Microchip proporciona en su cartera de productos, estos tres tipos de memorias, por lo que hacemos uso de la información que proporciona este proveedor al respeto de estos dispositivos. A continuación se muestra en la figura 25 una gráfica comparativa entre los tres tipos de memoria respecto a protocolos de comunicación soportados y capacidad de almacenamiento.

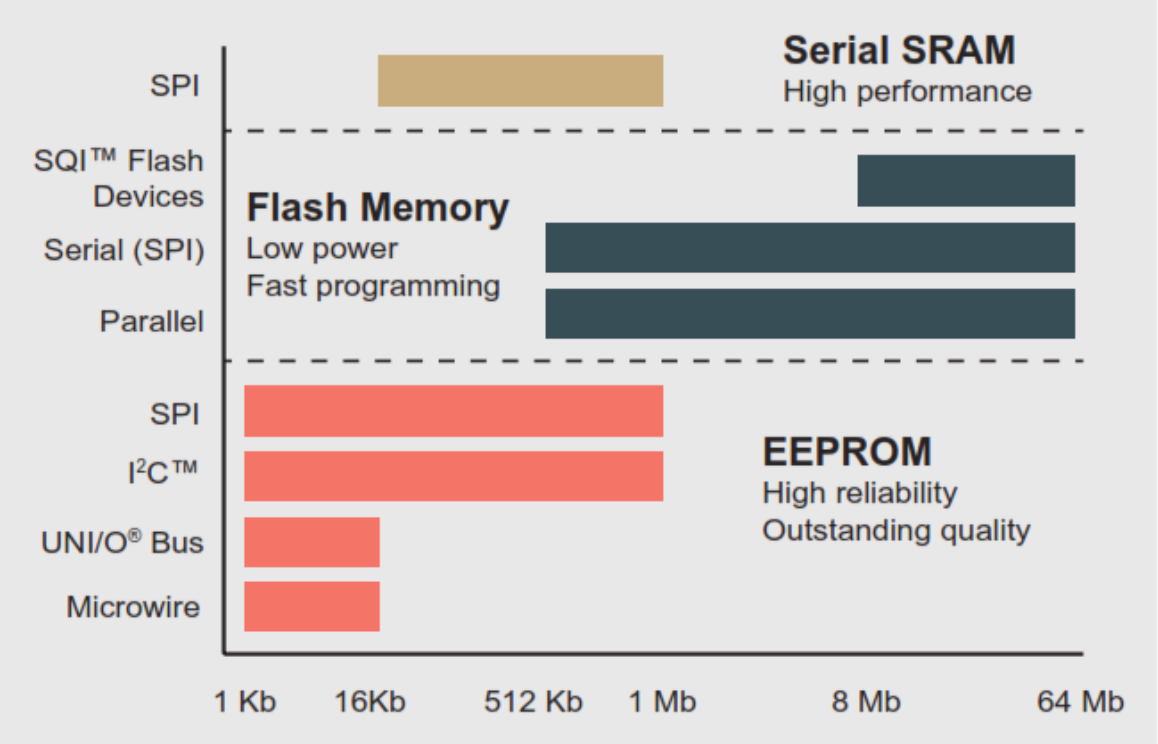


Figura 25: Comparativa entre memorias SRAM, EEPROM y FLASH.

A partir de la figura 25 se determinan los tipos de memoria a utilizar en el sistema embebido, **para la memoria RAM, se utilizara un tipo SRAM**, ya que no requiere un circuito de refresco, p**ara la memoria ROM, se utilizara un tipo FLASH**, ya que ofrece mayor capacidad de almacenamiento de 8 Mb hasta 64Mb y soportar los protocolos de comunicación SPI y SQI (disponible para memorias FLASH, que permite la transferencia de hasta 4 bits por cada ciclo de reloj lo que cuadriplica la velocidad de transferencia de datos), que hacen de la memoria FLASH la mejor opción. Para entender el tema del protocolo de comunicación serial SPI, a continuación se describen los ya mencionados:

SPI: Serial Peripheral Interface (interfaz de periféricos serial) es un estándar de comunicaciones, usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. El bus de interfaz de periféricos serie o bus SPI es un estándar para controlar casi cualquier dispositivo electrónico digital que acepte un flujo de bits serie regulado por un reloj (comunicación sincrónica).

Muchos sistemas digitales tienen periféricos que necesitan existir pero no ser rápidos. La ventajas de un bus serie es que minimiza el número de conductores, pines y el tamaño del circuito integrado. Esto reduce el coste de fabricación, montaje y pruebas electrónicas. Un bus de periféricos serie es la opción más flexible cuando se tiene tipos diferentes de periféricos serie. El hardware consiste en señales de reloj **(SCLK)**, data in **(MOSI)**, data out **(MISO)** y chip select **(SS)** para cada circuito integrado que tiene que ser controlado. A continuación se muestra en la figura 26 un diagrama esquemático de conexiones entre dispositivos mediante el estándar de comunicación serial SPI.

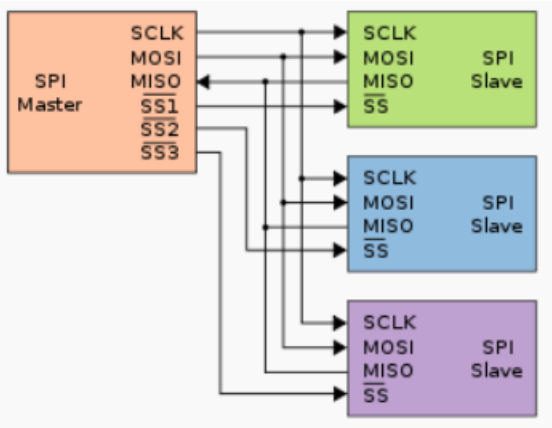


Figura 26: Comunicación SPI de un circuito maestro y tres esclavos.

El SPI es un protocolo síncrono. La sincronización y la transmisión de datos se realizan por medio de 4 señales:

- SCLK (Clock): Es el pulso que marca la sincronización. Con cada pulso de este reloj, se lee o se envía un bit.
- MOSI (Master Output Slave Input): Salida de datos del Master y entrada de datos al Slave. También llamada SIMO.
- MISO (Master Input Slave Output): Salida de datos del Slave y entrada al Master. También conocida por SOMI.
- SS/Select: Para seleccionar un Slave, o para que el Master le diga al Slave que se active. También llamada SSTE.

SQI/SDI I/O: Serial Quad/Serial Dual I/O, son estándares derivados del SPI, que permiten a un dispositivo SPI aumentar su ancho de banda desde un único dispositivo.

- El estándar SDI I/O, (bus de datos de dos bits) la interfaz permite duplicar la velocidad de transferencia en comparación con los dispositivos de memoria flash con el estándar SPI.
- El estándar SQI I/O, (bus de datos de cuatro bits), mejora el rendimiento de la interfaz cuatro veces y se abre una amplia gama de aplicaciones que requieren un mayor rendimiento.

La figura 27 muestra las conexiones para un dispositivo de memoria SPI, en modo SQI (bus de datos de 4 bits). Donde IO0-IO3, son terminales bidireccionales para entrada y salida de señales del master al slaves. SCLK es la señal de reloj generada por el master que sincroniza la transferencia de datos. Y /SC se utiliza para seleccionar o activar un Slave.

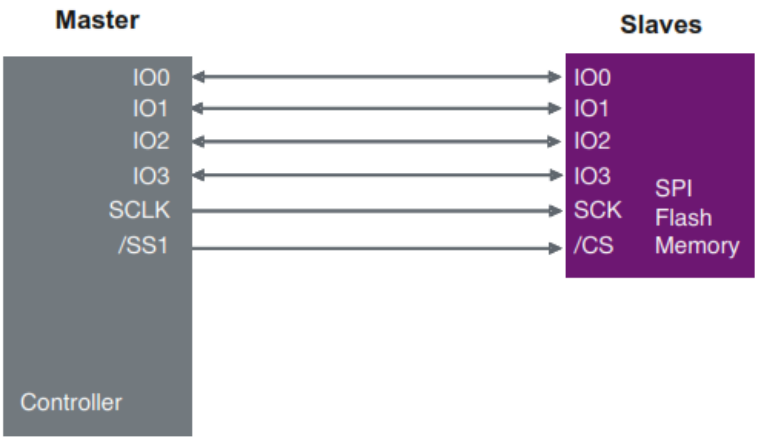


Figura 27: Memoria flash en modo SQI.

Como se mencionó anteriormente la memoria flash, soportan los estándares de comunicación SPI, SDI, SQI, siendo los dos últimos los que permiten aumentar el ancho de banda de la memoria dos veces o cuatro veces en un solo dispositivo. Hoy en día las memorias flash SPI, operan a velocidades de hasta 80 MHz, con un solo bit, Cuando se utiliza el dispositivo SPI en modo Quad, los 80 MHz equivalen a correr la flash a una frecuencia efectiva de 320MHz con hasta 40MB/segundo de velocidad de transferencia. Ya definido el tipo de memorias, para continuar con la selección de los dispositivos, se toma en cuenta el voltaje de alimentación y operación que debe de ser de 2.3V a 3.6V DC. Las memorias se seleccionaran del catálogo WEB de Microchip, se termina de elegir la memora SRAM y la FLASH-NOR, ya que influye principalmente la cantidad mínima de compra y existencia de cada modelo. A continuación se presentan, los modelos de las memorias seleccionadas:

Memoria SRAM marca Microchip, modelo 23LC1024-I/SN [17]:

- Capacidad de memoria: 1Mb (128K x 8 bit)
- Velocidad 20MHz.
- Interfaz: SPI, SDI y SQI.
- Voltaje de alimentación: 2.5V a 5.5V.
- Encapsulado: SOIC-8 (Ancho 3.9mm).

Memoria FLASH marca Microchip, modelo SST26VF032B104I/SM [18]:

- Capacidad de memoria: 32Mb (4M x 8 bit)
- Velocidad de 80 a 104MHz.
- Interfaz: SPI y SQI.
- Voltaje de alimentación: 2.7V a 3.6V.
- Encapsulado: SOIC-8 (Ancho 3.9mm).

En la figura 28 muestra el diagrama eléctrico implementado para las memorias SRAM y NOR-FLASH, para operar bajo os protocolos SPI, SDI o SQI, la diferencia entre estos protocolo de comunicación, es el número de datos transmitidos por ciclo de reloj y pines requeridos para la comunicación. Los pines de la memoria van conectados a los puertos I/O del FPGA.

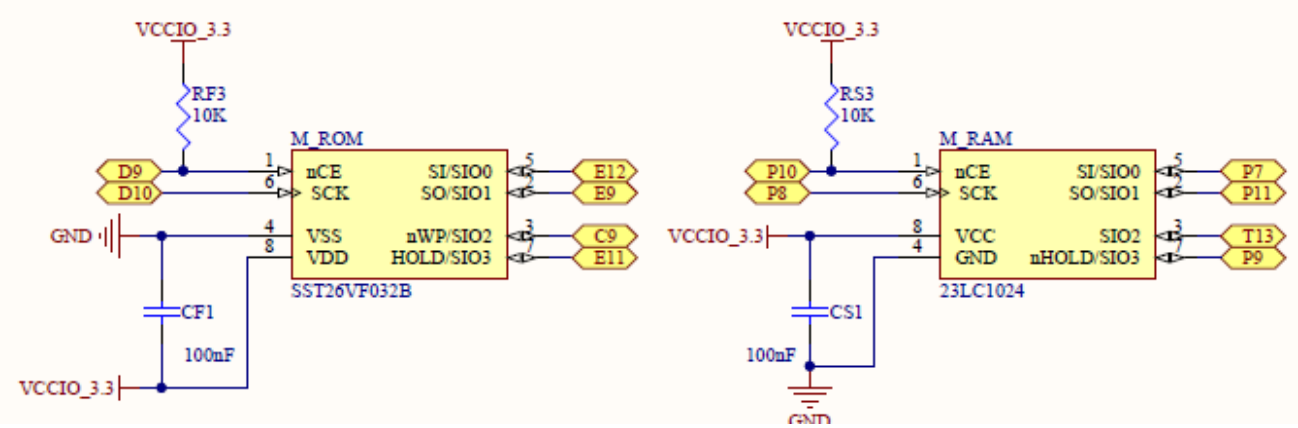


Figura 28: Diagrama eléctrico de memorias externas.

2.2.7 Modulo WiFi

Para la interfaz de usuario se pensó en un medio de comunicación inalámbrico que nos permitiera enviar y recibir datos desde un equipo de cómputo, para ello se eligió al módulo RN-XV que está basado en el módulo RN-171 de Roving Networks. Básicamente es el módulo RN-171 adaptado a una PCB con el fin de ser compatible con el socket XBee. Este diseño facilita su uso ya que el RN-171 por si sólo es de montaje superficial y su método de soldadura es delicado, además de requerir el diseño de una PCB para su montaje. También cuenta con una antena integrada a la PCB y LEDs indicadores como se muestra en la figura 29.

Figura 29: Módulo de comunicación inalámbrica RN-XV.

El módulo cuentan con un firmware integrado el cual facilita su configuración, la configuración puede ser mediante un método alambrado (serial) o inalámbrico mediante Telnet [19]. El módulo RN-XV tiene montado el módulo RN-171, el cual cuenta con dos

protocolos seriales de configuración los cuales son el Serial Peripheral Interface (SPI) y el Universal Asynchronous Receiver Transmitter (UART). Sin embargo se usa el protocolo UART como medio de comunicación serial entre el módulo y el FPGA. En la figura 30 se muestran las conexiones del módulo RN-XV con el FPGA, las cuales corresponden únicamente a señales TX y RX.

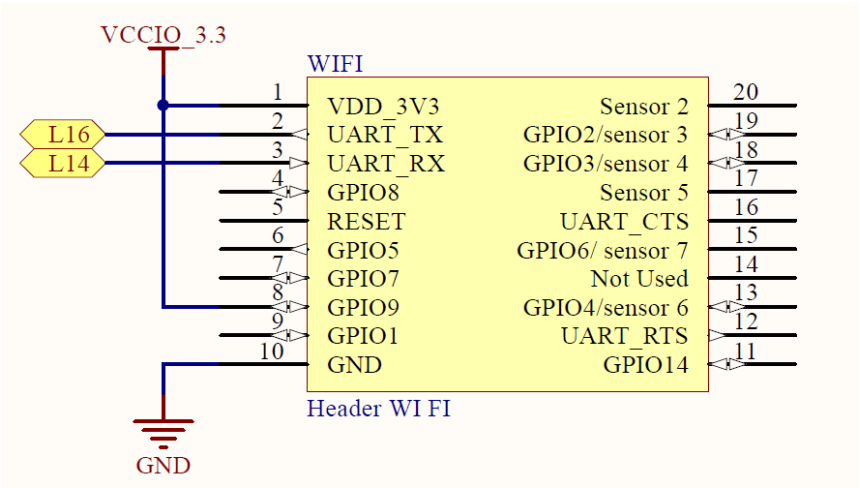


Figura 30: Conexión de módulo WIFI con FPGA.

2.2.8 Circuito de distribución I/O y relojes

La tarjeta cuenta con 4 conectores (header) macho de 50 pines que conectan directamente con las terminales I/O y relojes del FPGA, además proporcionan terminales de alimentación VCC y pines a GND. A continuación en las figuras 31 y 32 se muestran los bancos de I/O que conforman al FPGA seleccionado y en la figura 33 se muestran los 4 conectores con 50 pines cada uno y sus conexiones a pines I/O y relojes del FPGA, como también sus conexiones a VCCIO y GND.

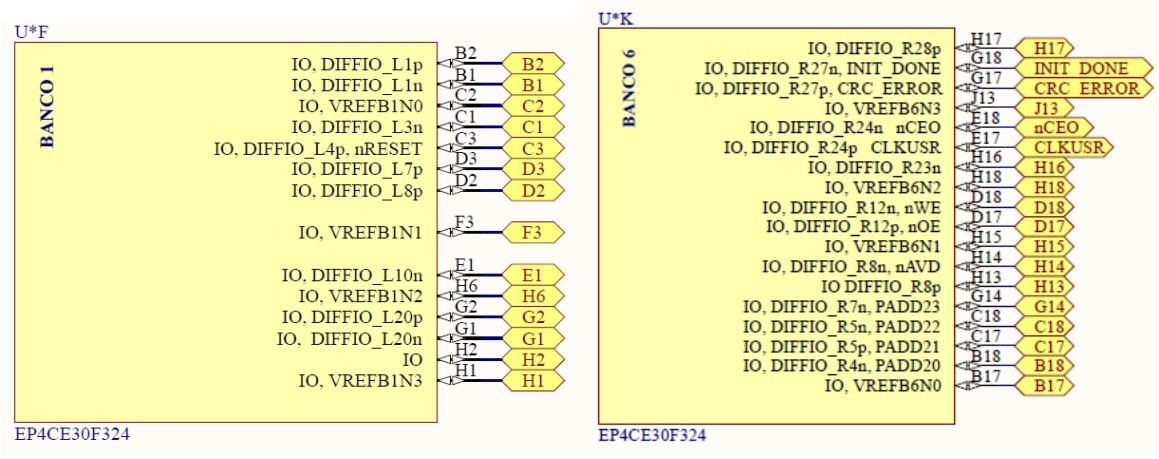


Figura 31: Bancos de entradas/salidas 1 y 6 del FPGA.

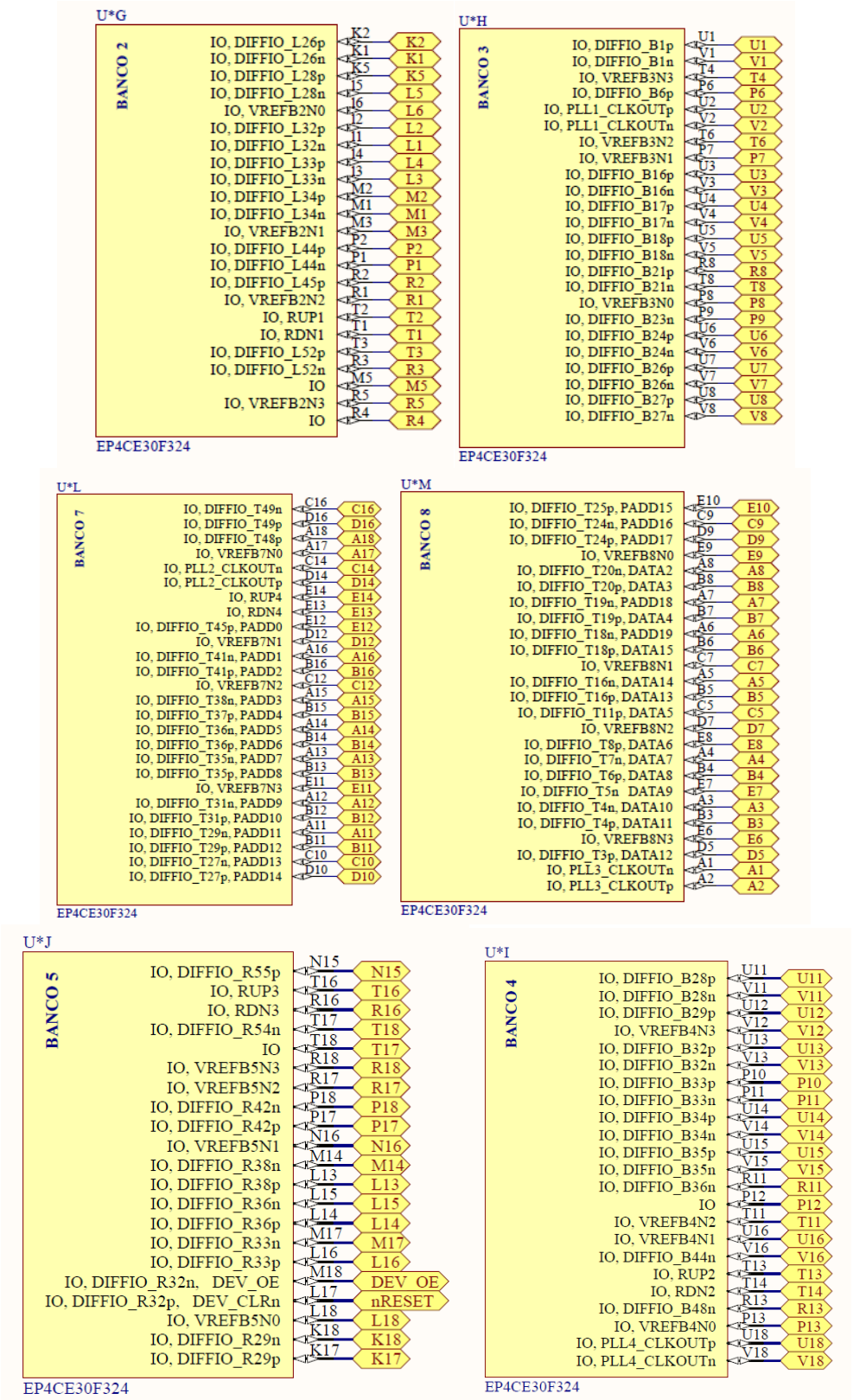


Figura 32: Bancos de entradas/salidas 2, 3, 4, 5, 7 y 8 del FPGA.

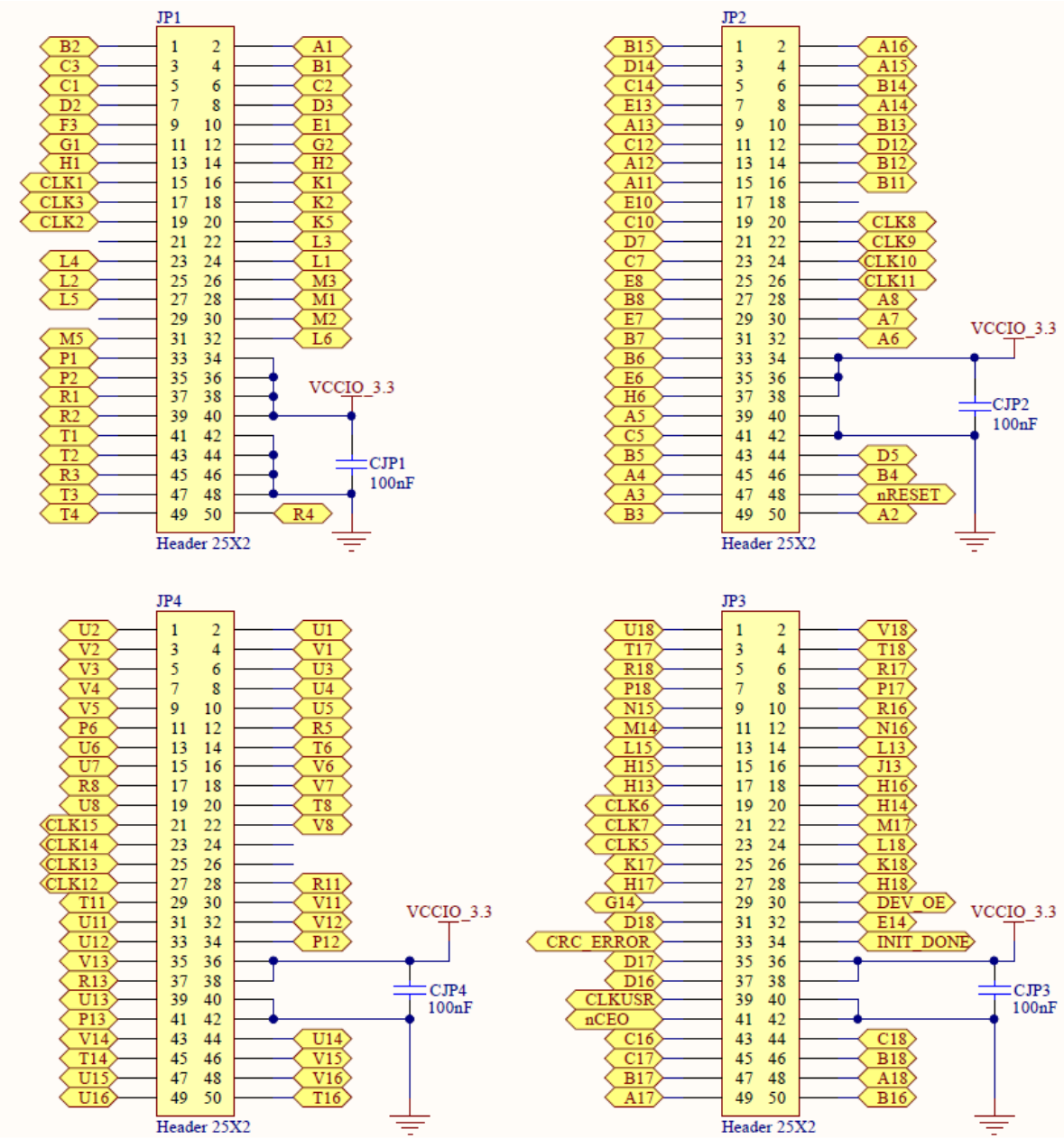


Figura 33: Distribución de pines I/O y relojes del sistema embebido.

En la tabla 11 se presenta a forma de resumen el listado de componentes principales que van a conformar al sistema embebido.

Tabla 11: Lista de principales componentes seleccionados.

Componente.	**Dispositivo.**	**Descripción.**
FPGA	**EP4CE30F19A7N**	FPGA de 30K LE.
Memoria de configuración	**EPCS16**	Memoria de 16Mb.
Memoria SRAM	**23LC1024-I/SN**	Memoria de datos de 1Mb.
Memoria Flash-NOR	**SST26VF032B104I/SM**	Memoria de programa 32Mb.
Módulo WIFI	**RN-XV**	Módulo WIFI. Protocolo 802.11g

2.3 Diseño del circuito impreso PCB

De acuerdo a la norma UNE 20-621-84, el **CIRCUITO IMPRESO** se define como un modo de conexión de los elementos o componentes electrónicos por medio de pistas de cobre, normalmente adheridas a un soporte aislante rígido o flexible [20]. La *placa* o *tarjeta* de circuito impreso PCB (printed circuit board) suele ser una superficie plana de un espesor variable y normalmente de forma rectangular o cuadrada; está constituida por un material base o sustrato de tipo laminado rígido o flexible que sirve de soporte físico aislante para la colocación y soldadura de los componentes y el trazado de las pistas conductoras de cobre. El soporte base tiene que ser muy buen aislante eléctrico y muy resistente al fuego. Actual mente los materiales más utilizados son: fibra de vidrio, politetrafluoretileno, PTFE-fibra de vidrio, PTFE-fibra de cerámica, termoplásticos, resina epoxidica, resina de silicona, resina melaminica, etc. Y diferentes mezclas entre ellos para mejorar las propiedades finales de substrato. Los tipos de PCB que actualmente se fabrican son:

- Monocapa o simple capa.
- Bicapa o doble cara.
- Multicapa o más de dos caras.
- Flexible - multicapa.
- Rígido-flexible multicapa.
- Tridimensional o MCB.

2.4.1 Estrategias de diseño

Para el diseño del PCB debemos de tomar en cuenta un factor muy importante y que para cualquier tarjeta de circuito impreso es fatal, el ruido que no es más que una señal no deseada y que podría interferir con el correcto funcionamiento de nuestro sistema. Existen dos tipos de ruido: interno y externo [21]. A continuación se expone a detalle cada uno.

2.4.1.1 Ruido interno

Como ruido interno tenemos el generado por la interferencia electromagnética, por las fuentes de alimentación y la interferencia entre pistas.

2.4.1.2 Interferencia electro magnética en circuitos electrónicos

Las interferencias electromagnéticas EMI se pueden definir como señales electromagnéticas que perturban (sin intención) el funcionamiento normal de un sistema eléctrico o electrónico.

Los tipos de interferencia, métodos de medida, y los limites tolerados, están especificados por normas intencionales y se establecen en función de la banda de frecuencia de interferencias.

- Perturbaciones en baja frecuencia: (f<10KHz), emitidas por la red y fuentes de alimentación.
- Perturbaciones en banda de 10-150 KHz: Por impulsos de intensidad y fenómenos transitorios de tensión interruptores, etc.

- Perturbaciones en banda de 150 KHz a 30 MHz: Por impulsos de intensidad y fenómenos transitorios de tensión interruptores, la propagación es por radiación y acoplamiento.
- Perturbaciones en banda de 30 MHz a 300 MHz: Propagación por radiación.
- Perturbaciones en banda de 500 MHz a 18 GHz: Se propaga por radiación y suelen aparecer en los equipos de comunicaciones o los circuitos lógicos de conmutación.

Por una EMI se pude entender como la presencia de voltaje o corriente no deseada que puede aparecer en un equipo como resultado de la operación de otro aparato o fenómeno natural. El ruido en las líneas y las fuentes que las causan son múltiples como los son:

- Interrupción de carga.
- Transmisión de estaciones de comunicación.
- Puesta a tierra de los equipos pobre o nula.
- Descargas atmosféricas (rayos).
- Operaciones de equipo pesado.
- Motores eléctricos funcionando cerca del equipo electrónico sensible.

Algunos métodos para reducir las interferencia en un circuito digital en una tarjeta impresa, consiste en usar un plano o una rejilla a tierra, cada conector de entrada y salida debe tener conexiones a tierra, además de intercalar líneas de tierra entre las líneas de señal.

2.4.1.3 Interferencia electro magnética por fuentes de alimentación

La interferencia electromagnética producida por fuentes de alimentación surge debido a que todas las fuentes de voltaje se rectifican, y estas idealmente deben proporcionar un voltaje continuo sin embargo se produce un voltaje rizo a causa de que en realidad se presenta una carga y descarga como se muestra en la figura 34.

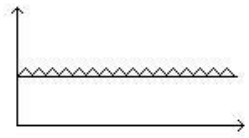


Figura 34: Voltaje de rizo de una fuente de alimentación.

Esto nos transforma nuestra fuente ideal de corriente directa en dos fuentes, una con voltaje de corriente directa y otra con el voltaje de corriente alterna como se muestra en la figura 35, entonces cuando alimentamos cualquier circuito, cualquier integrado en realidad lo estamos alimentando con corriente alterna.

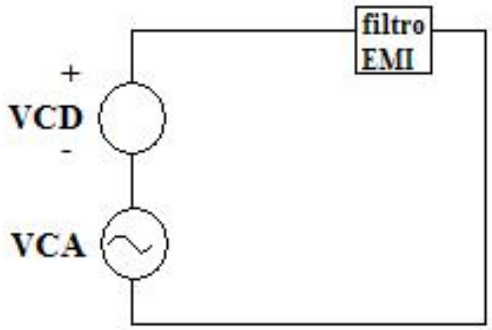


Figura 35: Fuente de voltaje con filtro EMI.

Este fenómeno se combate con los filtros de interferencia electromagnética (EMI) figura 36, hay muchos tipos de filtros EMI, un arreglo de capacitores. El rizo se va a tierra y es absorbido por el filtro. Es decir, los filtros EMI reducen el ruido proveniente de las conexiones a la fuente de alimentación de línea.

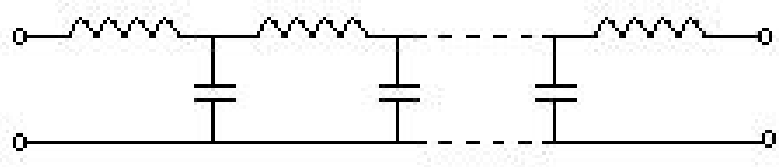


Figura 36: Arreglo de filtro EMI.

2.4.1.4 Interferencia entre pistas

En la tarjeta de circuito impreso se generan guías de onda, para este diseño la tarjeta contendrá en toda su área pistas que nos conducen a los dispositivos y unirán cada una de las etapas de la tarjeta para su correcto funcionamiento, a continuación de estas pistas que pertenecen a la primera capa, se encuentra la segunda capa que corresponde al plano de tierra, cuando la tarjeta está en funcionamiento, el campo electromagnético en un 90 o 95% se va al plano de tierra pero un 5 o 10% interfiere con la siguiente pista, ver figura 37 y es el momento en que se produce la interferencia entre pistas, esto es absolutamente normal en la tarjeta, este fenómeno se le llama ruido por interferencia entre pistas con el plano de tierra. Cabe mencionar que este fenómeno se presenta cuando trabajamos con frecuencias a partir de los 50MHz.

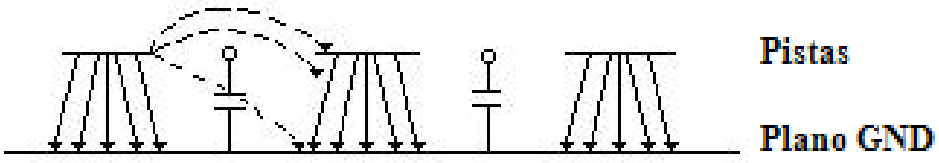


Figura 37: Interferencia entre pistas

Debemos tomar en cuenta que entre más pistas y entradas digitales tengamos, más se va a contribuir a la interferencia electromagnética y por lo tanto va haber más inducción de ruido, es por esto que cada vez que se coloca un dispositivo debemos de colocar capacitores de 100nF de VCC a tierra y deben ser distribuidos en toda el área de la tarjeta para reducir la interferencia electromagnética entre pistas (suficiente para sistemas digitales CMOS, LVTTL y TTL). Para reducir la interferencia entre pistas aterrizamos los componentes electrónicos utilizados y algunos otros puntos con capacitores de 100nF para poner cortos en corriente alterna.

2.4.1.5 Ruido externo

Nuestro entorno está inundado de ruido externo provocado por: las computadoras, ondas electromagnéticas de radio, televisión, telefonía, etc.

En un metal, a temperatura constante, la densidad de corriente $\boldsymbol{J}$ es linealmente proporcional al campo eléctrico $\boldsymbol{E}$.

$$\boldsymbol{J} = g\boldsymbol{E} \tag{7}$$

Donde g es la conductividad.

La densidad de corriente $\boldsymbol{J}$, caracteriza a un punto interno del conductor. Si la corriente i se distribuye uniformemente a través del área $\boldsymbol{A}$ de la sección transversal de un conductor, la magnitud de la densidad de corriente de los puntos en esa sección transversal es:

$$\boldsymbol{J} = \frac{i}{\boldsymbol{A}}, \Rightarrow \frac{i}{\boldsymbol{A}} = g\boldsymbol{E} \Rightarrow \boldsymbol{E} = \frac{i}{\boldsymbol{A}} g, \tag{8}$$

La conductividad idealmente tiende a infinito y por lo tanto

$$\boldsymbol{E} = 0. \tag{9}$$

Por esta razón, en una cierta área alrededor de la tarjeta no existe ningún factor externo incidiendo, así, se usan componentes superficiales de tal manera que se ubiquen en el área carente de ruido externo

2.5 Conceptos básicos de fabricación de PCBs

Los paquetes de diseño de PCBs están compuestos por tres herramientas, el editor de componentes, el editor de esquemas y el postprocesador. Cuando iniciamos el diseño de una nueva PCB, las medidas están impuestas por restricciones de diseño, normalmente se trabaja con medidas inglesas, se emplea la milésima de pulgada, donde una 1 in = 2.45 cm.

La librería de componentes es la parte más importante del programa de diseño. Un programa de captura de diagramas con herramientas de diseño de PCB, tiene enlaces que definen como un símbolo del diagrama se relaciona con un símbolo de PCB. El procesado del diagrama genera un fichero de netlist para el programa de ruteo. Hay que prestar atención entre la relación de la numeración de las terminales del símbolo en el diagrama y en el encapsulado del componente. Altium Designer permite diseñar encapsulados

especiales. Toda la información de la lista de materiales se puede almacenar con cada componente si creamos un PCB a partir de un diagrama.

De manera general los pasos para hacer un diseño de PCB consisten en:

- Primera fase: Consiste en definir las dimensiones exteriores de la placa y los agujeros de montaje necesarios; luego, se lee el netlist creado por el software de captura de diagrama. Este proceso lee componentes de las librerías y los coloca uno encima de otros en el origen del dibujo.
- Segunda fase: Es la correcta colocación de los componentes. Cuando se mueve un componente a una zona libre de la placa, observamos líneas que lo unen a otros componentes, estas líneas sirven para encontrar la disposición más adecuada con respecto a los demás componentes. Colocar los condensadores de desacoplo cerca del extremo de los circuitos integrados (IC, *Integrated Circuit*) para mantener la longitud de las pistas lo más cortas posible. Los componentes de la etapa analógica deben permanecer separados de la digital.
- Tercera fase: Consiste en colocar las pistas de cobre que unen los componentes, esta fase puede ser automática o manual.
- Cuarta fase: Es generar los archivos necesarios para la fabricación; en el caso de hacer este proceso de manera manual, entonces imprimir mediante una impresora de alta resolución el diseño final en un papel especial que sea transparente a la luz actínica.

Altium Designer incluye varias características de automatización para facilitar el ruteo manual de la placa, utiliza un algoritmo de conectividad que analiza continuamente el diseño a medida que se rutea, optimizando y ajustando las líneas de conexión a medida que se coloca cada pista. Esto nos permite rutear las conexiones a lo largo de cualquier camino, sin tener en cuenta la línea de conexión actual. Una rejilla eléctrica tiene preferencia sobre la rejilla snap, que se usa para la visualización, permitiéndonos rutear conexiones de componentes que no se ajusten a la rejilla. Altium Designer incluye varios modos de colocación de pistas: en 45 grados, ortogonal, oblicuo o en arco. Asimismo, se puede alternar entre capas mientras se hace el ruteo. Altium Designer permite definir modos de ruteo; El modo de ruteo “Evitar obstáculo” monitorea de forma automática el ruteado e impide la violación inadvertida de cualquier regla de diseño al colocar una pista en un objeto que no se encuentre en la misma conexión. El modo “Empujar obstáculo” permite mover pistas a medida que se rutea.

Existen tres tipos de capas usadas en el editor PCB:

1. Electrical Layers (Capas eléctricas). Estas incluyen 32 capas de señales y 16 capas planas internas. Las capas son añadidas o eliminadas desde Layer Stack Manager.

2. Mechanical Layers (Capas mecánicas). Hay 16 capas mecánicas de propósito general para definir los límites de la tarjeta, ubicar las dimensiones, incluir los detalles de fabricación o cualquier detalle mecánico que requiera el diseño. Estas capas pueden seleccionarse o incluirse dentro de la impresión y los archivos Gerber.

3. Special Layer (Capas especiales). Se encuentran tanto en la parte superior e inferior de las capas de serigrafía o silkscreen de soldadura de perforación, de keep out (usado para establecer los límites de las conexiones eléctricas) de conexión de error y entre las multicapas.

2.6 Diseño de PCB multicapa

El diseño del PCB del sistema embebido, consta de cinco capas de cobre y cuatro de material dieléctrico. Las pistas para las señales se localizan en la capa superior (TOP LAYER), intermedia (MID LAYER) e inferior (BOTTON LAYER); El plano de alimentación se localiza entre la capa intermedia e inferior y la capa de tierra se localizan entre las capas superior e intermedia tal como se muestra en la figura 38.

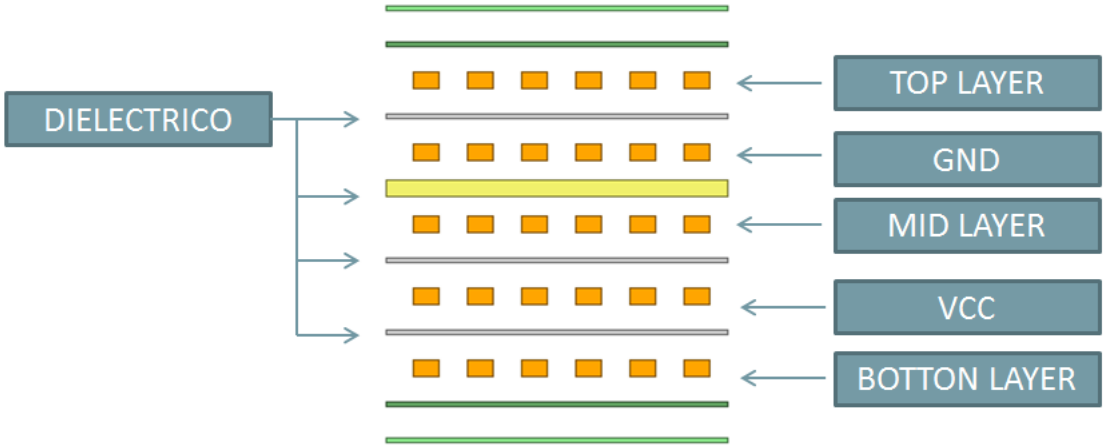


Figura 38: Distribución de 5 capas del PCB

Para insertar cada dispositivo, Altium posee una gran cantidad de librerías, sin embargo si no son suficientes, el programa permite diseñarlos; para el diseño de esta tarjeta, se crearon algunas librerías, como las que contienen las memorias, el FPGA y el módulo Wi-Fi, etc.

Primeramente se crea un proyecto, posteriormente se crean las librerías del circuito de los componentes es decir el *schematic library* asignándole el nombre del dispositivo es importante asignar una numeración o nombre a cada pin del dispositivo ya que es la conexión que el programa seguirá en cuando se diseñe la *pcb library* del dispositivo, una vez que se ha diseñado el *schematic library* de cada uno de los dispositivos se deben crear las huellas (*footprint*), las cuales serán las que se plasmen en la tarjeta electrónica, por lo que para diseñar el *pcb library* es necesario realizar una medición física correcta de cada dispositivo, una vez que se tiene la huella, se le asigna un nombre y se da de alta en las librerías, así puede ser utilizado para cualquier dispositivo al que se le agregue dicha huella.

Ya que se tienen los componentes y las huellas, se acomodan los dispositivos y se hace el ruteo. Altium tiene la propiedad de realizar un ruteo automatizado, pero se recomienda hacerlo manualmente ya que la experiencia que se tuvo, consistió en reducir la longitud de las pistas para disminuir ruidos, para un circuito el proceso de ruteo automático en Altium

se vuelve muy lento y tardado, aunque no se desprecia en algunos casos esta bondad que el programa ofrece.

2.6.1 Diseño de capa superior

Mediante el diseño electrónico del sistema embebido, se contemplan los circuitos integrados a utilizar y sus requerimientos de alimentación y de conexión de entradas y salidas con los demás dispositivos. Con el uso de Altium Designer se compila el diagrama esquemático mostrado en la figura 6 y se genera la primera capa del PCB, el cual debe ser trazado en un espacio previamente definido de acuerdo al tamaño de las tarjetas que le preceden a nuestro diseño que corresponde a 75.325mm X 75.6 mm, esto con el fin de asegurar la compatibilidad entre tarjetas respecto a la distribución y ubicación de los pines de los 4 headers de propósito general. En la capa superior o Top Layer están colocados los siguientes componentes: Un FPGA modelo EP4CE30F19A7N de la marca Altera, un puerto de programación JTAG y otro para programación por USB-BLASTER, un botón de reinicio, un conector de alimentación externa, cuatro conectores de propósito general de 50 pines cada uno y un módulo Wi-Fi. En esta capa se trazó la mayoría parte de pistas de conexión entre los pines I/O del FPGA con los conectores de propósito general. En la figura 39 se muestra la capa superior.

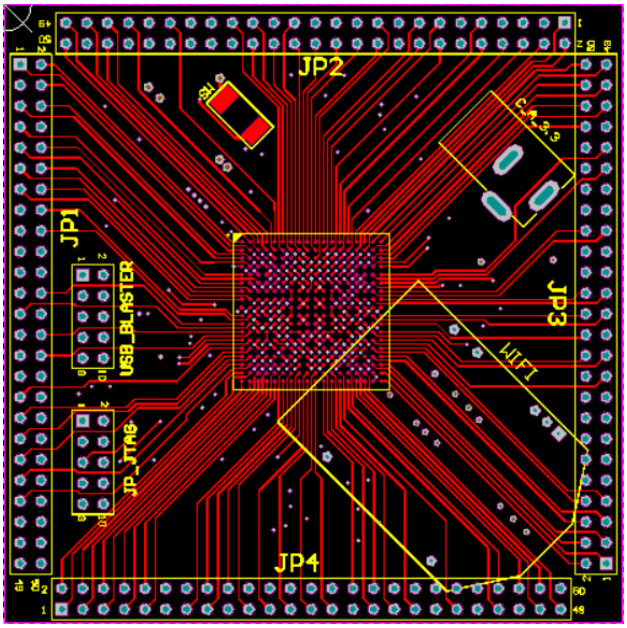


Figura 39: Capa superior o TOP LAYER.

2.6.2 Diseño de capa a GND

Teniendo en cuenta que en la capa superior están el FPGA, el módulo WiFi y la mayoría de pistas del circuito eléctrico, se coloca una segunda capa a GND. Con la cual se reducirá la interferencia electromagnética entre pistas, considerando que las señales del FPGA pueden operar a frecuencias mayores a 50 MHz, además del plano de tierra se recurre a colocar capacitores de 100nF de VCC a tierra, colocándolos en toda el área de la tarjeta y cerca de los circuitos integrados para reducir la interferencia electromagnética entre pistas. En la

figura 40 se muestra el plano de GND que se conecta a las terminales de GND de los headers de propósito y por medio de vías se conectan los distintos circuitos integrados del sistema embebido.

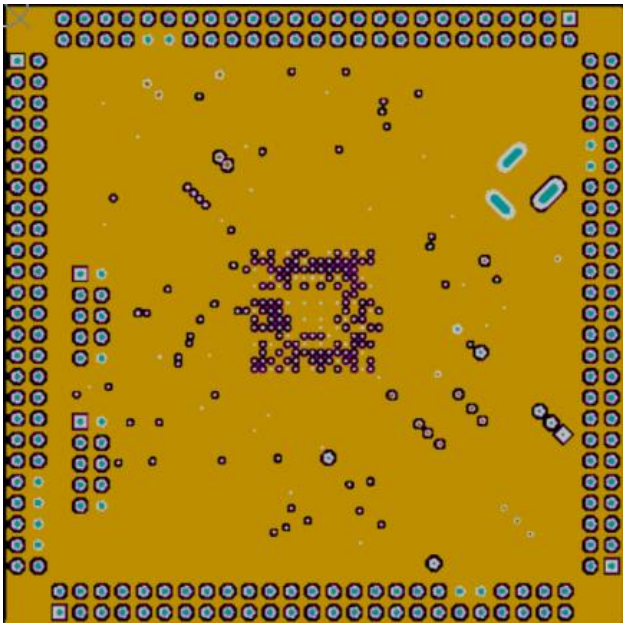


Figura 40: Capa de GND.

2.6.3 Diseño de capa media

El diseño de la tarjeta en si no se compone de un gran número de circuitos integrados, ya que se emplea un FPGA, pero debido a que este cuenta con 324 pines que hay que rutear, el espacio para el trazo de pista se reduce, por componentes y pistas que bloquean el paso de pistas nuevas en los distintos planos utilizados. Debido a esto se crea una tercera capa a la que llamamos capa media o MID LAYER. En esta capa se trazaron las pistas que conectan a los pines I/O y de reloj del FPGA de difícil acceso con los headers de propósito general y las conexiones que conectan al FPGA con el módulo WiFi En la figura 41 se muestra la capa media o MID LAYER.

2.6.4 Diseño de capa a VCC

El plano de alimentación, es colocado en cuarto lugar, de acuerdo al diseño electrónico de la tarjeta electrónica diseñada, en este plano se distribuyen tres distintos voltajes 3.3VDC, 2.5VDC y 1.2VDC a través de vías a los distintos componentes haciendo uso también de semiplanos. Este plano también tiene la finalidad de reducir las interferencias electromagnéticas entre pistas de la capa Mid Layer con la capa Botton Layer creando semiplanos conectados a GND sobre pistas que conectan algunas señales de reloj del FPGA a terminales de los headers de propósito general. En la figura 42 se muestra la capa conectada a VCC de la tarjeta.

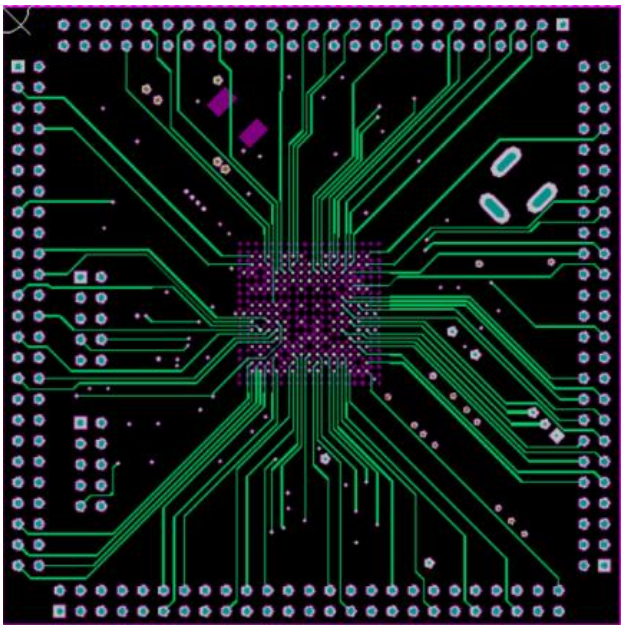


Figura 41: Capa media o MID LAYER.

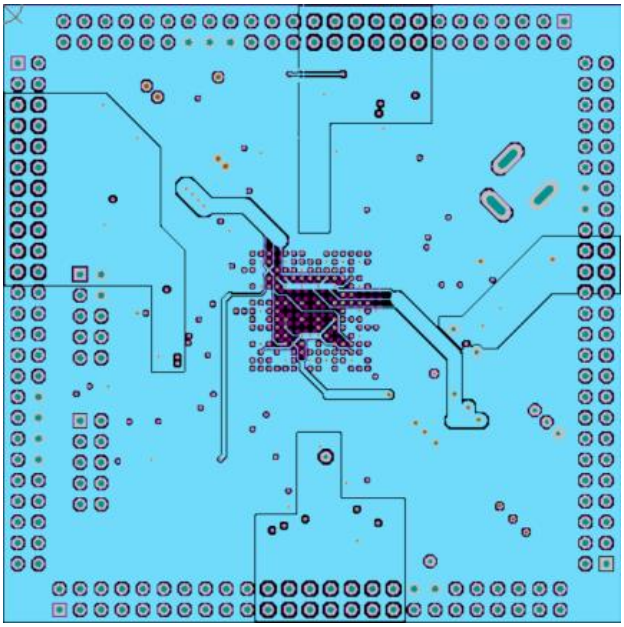


Figura 42: Capa a VCC.

2.6.5 Diseño de capa inferior

La quinta capa que corresponde a la capa inferior o BOTTOM LAYER esta paralela al plano de alimentación por lo que el ruido es más intenso que la capa superior. Para evitar la existencia de cualquier ruido interno en este plano todos los capacitores utilizados en esta tarjeta son colocados en esta capa.

En la capa inferior o Botton Layer están colocados los siguientes componentes: una memoria de configuración EPCS16, dos convertidores DC-DC, un oscilador de cristal, una

memoria FLASH NOR de 32 Mb, una memoria SRAM de 1 Mb, y un conjunto de resistencias y capacitores. En este plano, se trazaron las pistas para las conexiones entre los puertos de programación JTAG, usb-blaster y los dispositivos mencionados con el FPGA. En este plano se colocaron capacitores de 100nf conectados de VCC a GND, lo más cerca de los circuitos integrados, como es en el caso del FPGA y memoria, que los capacitores se colocaron alrededor del dispositivo para reducir las interferencias electromagnéticas entre pistas. En la figura 43 se muestra la capa inferior o BOTTOM LAYER.

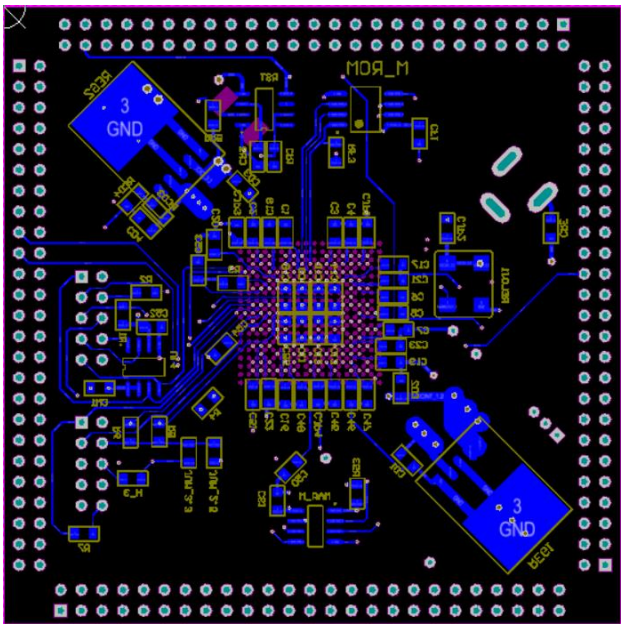


Figura 43: Capa inferior o BOTTOM LAYER.

2.7 Tarjeta con FPGA, RAM y ROM externas

En la figura 44 se muestra la vista 3D del PCB del sistema embebido, en la imagen de la izquierda se muestra la vista superior de la tarjeta y en la imagen de la derecha la vista inferior.

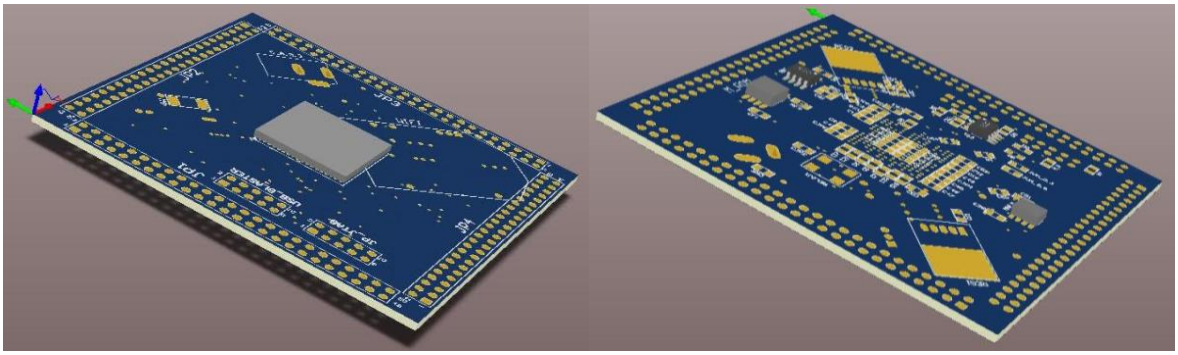


Figura 44: Vista 3d de tarjeta con FPGA, RAM y ROM externas.

Ya trazadas las pistas y realizadas las conexiones entre los dispositivos y los planos utilizados, se procedió a revisar en conjunto con mis asesores el diseño de PCB del sistema

embebido de acuerdo al diagrama eléctrico generado. Tras varias revisiones y correcciones se aprobó el diseño del PCB, y se comenzaron a generar los archivos para su fabricación pero debido a que en momento no se contó con el recuso económico para la fabricación se procedió a utilizar un sistema embebido comercial con características similares, esta tarjeta corresponde a un diseño desarrollado por terasiC, el cual distribuye tarjetas electrónicas basas en FPGAs y CPLDs de la marca Altera. La tarjeta a la que nos referimos es la DEO-Nano que se muestra en la figura 45.

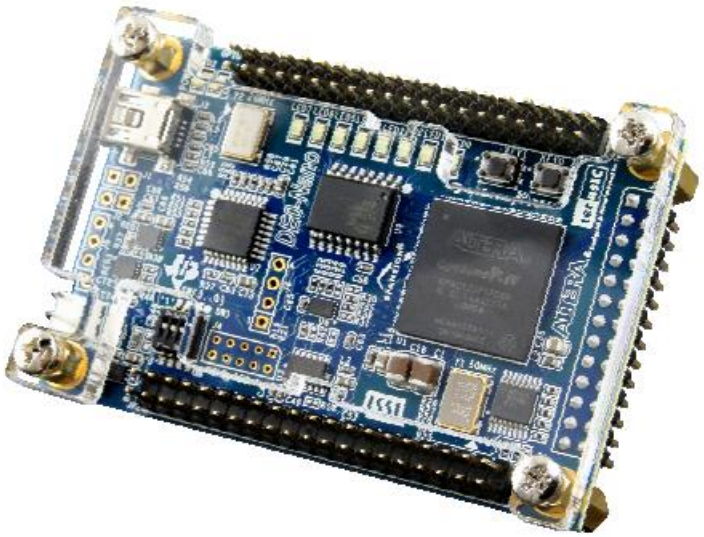


Figura 45: Tarjeta DEO-Nano.

La tarjeta DEO-NANO es ideal para su uso con procesadores embebidos, presenta un potente FPGA Altera Cyclone IV (con 22,320 elementos lógicos), un cristal de 50MHz una memoria SDRAM de 32MB, una memoria EEPROM de 2 Kb y un dispositivo de memoria de configuración serial de 64Mb. Para conectarse a al mundo real, el DEO-NANO incluye un convertidor Analógico Digital de National Semiconductor de 8 canales y 12 bits, y también cuenta con dispositivo acelerómetro de 3 ejes y 13 bits de Analog Device [22].

Con la DEO-NANO, se tiene un FPGA con el número de elementos lógicos que se requieren, faltando únicamente el circuito de las memorias y el del módulo WiFi, para esto se desarrollaron tarjetas prototipo implementando estrategias para disminuir las EMI en las señales, se utilizaron planos de tierra para las memorias ya que esta operan a frecuencias de 20Mhz y 80Mhz, además se colocaron capacitores de 100nf, que conectan las terminales de alimentación VCC de cada dispositivo a GND. Respecto al módulo RN-XV, únicamente se fabricó una tarjeta con zócalos para su sujeción. En la figura 46 se presenta el sistema implementado para la instrumentación del microprocesador a diseñar.

En la figura 46 se muestran las conexiones eléctricas de las memorias y módulo WiFi de acuerdo al diagrama eléctrico generado, para la comunicación del FPGA con las memorias se realizaron las conexiones para usar el protocolos SPI, y para la comunicación con el módulo WiFi se realizaron las conexiones para usar el protocolo UART.

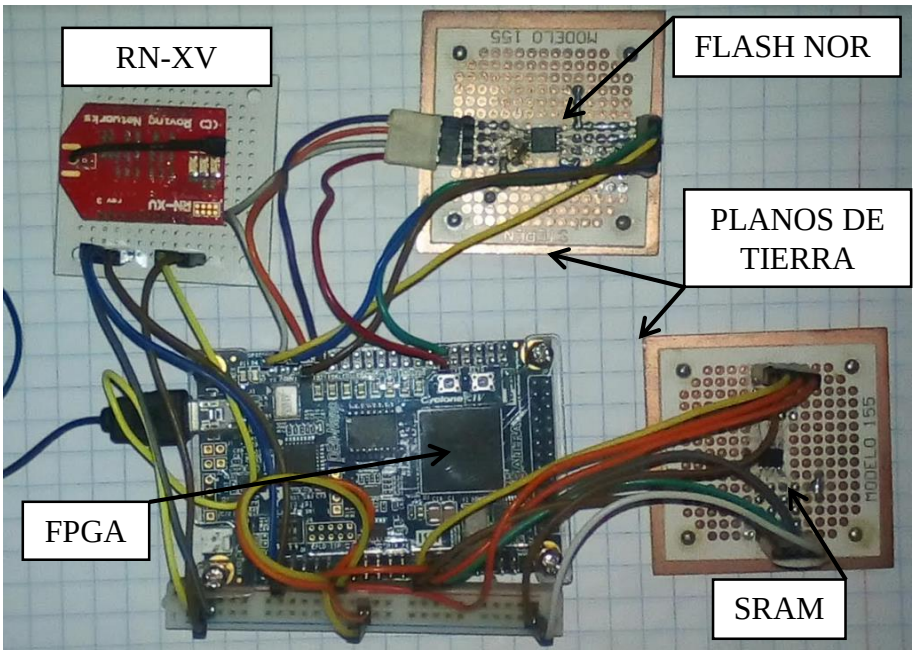


Figura 46: Sistema implementado para la instrumentación de procesador embebido.

En el siguiente capítulo, se presentara el firmware desarrollado para el desarrollo del microprocesador embebido, de los administradores de memoria, la comunicación UART y por último se presenta el desarrollo del software de configuración que nos permitirá grabar programas en la memoria FLASH y datos en la SRAM de la tarjeta, él envió de parámetros de control y la lectura del contenido de la memoria de datos.

2.8 Conclusiones

Se realizó el diseño electrónico de un sistema embebido que cumple con los requerimientos de hardware indicados en el capítulo 1, ya que cuenta con un FPGA Cyclone IV de 30 K elementos lógicos, memorias externas y una interfaz inalámbrica WiFi.

Al diseñar el circuito impreso del sistema embebido fue de suma importancia el contemplar todas las estrategias para la reducción de ruidos, como son los planos de tierra y el uso de capacitores, debido a que se manejan señales de alta frecuencia.

El diseño de circuito impreso del sistema embebido se generó en 5 capas, ya que Altium ofrece las herramientas para el desarrollo de tarjetas multicapa, esto contribuye a la reducción de espacio y del costo en la fabricación de la tarjeta.

Capítulo 3. Firmware

El firmware se define como un bloque de instrucciones de máquina para propósitos específicos, grabado en una memoria no volátil (Flash, EEPROM, ROM, etc.) de solo lectura, que establece la lógica de más bajo nivel que controla los circuitos electrónicos de un dispositivo [23]. En este capítulo se presenta el diseño e implementación en FPGA de los diferentes elementos que conforman el firmware del sistema embebido propuesto utilizando el software de Quartus II versión 13. Además se presenta una interfaz gráfica de usuario diseñada en el software de LabView para la configuración, envío de parámetros y lectura de datos del sistema embebido desde una PC vía WiFi.

3.1 Diagrama general del firmware del sistema embebido

La figura 47 representa el diagrama a bloques del firmware del sistema embebido. Destacan los siguientes componentes que serán explicados más adelante.

- Memoria de programa tipo ROM Flash de 4M x 8 bits.
- Memoria de datos de tipo SRAM de 128K x 8 bits.
- Administradores de memoria, tienen la función de controlar y establecer comunicación entre los dispositivos de almacenamiento externo y el decodificador de instrucciones para la escritura y lectura de datos de una localidad de memoria.
- Puertos de entrada y puertos de salidas para la comunicación con el exterior, cuatro puertos de entrada (PENT) y cuatro puertos de salida (PSAL), cada uno de 32 bits. Los puertos de entrada están conectados a un multiplexor con el cual el decodificador de instrucciones permite el paso y procesamiento de un bus a la vez. Los puertos de salida se obtiene a partir de un demultiplexor que permite el direccionamiento de un único puerto de salida del decodificador de instrucciones hacia cuatro salidas distintas.
- Un microprocesador de 32 bits con arquitectura Harvard el cual se compone de un decodificador de instrucciones y una unidad lógico aritmética (ULA) de 32 bits.
- ULA de 32 bits para la ejecución de operaciones aritméticas y comparación de números en punto flotante de simple precisión bajo el estándar IEEE 754.
- Decodificador de instrucciones, lee y decodifica los bits de los campos que componen las instrucciones de un programa, para generar las señales de control adecuadas para especificar a los demás elementos la operación a realizar. Además determina la dirección de la siguiente instrucción a ejecutar.
- Administrador de comandos, este decodifica los comandos y datos enviados por el software de interfaz, genera y envía señales de control y datos a los módulos de memoria, microprocesador e interfaz WiFi.
- Interfaz WiFi, permite el control y monitoreo del sistema embebido desde un equipo de cómputo. Para su implementación se utiliza el módulo RN XV que emplea el protocolo UART para comunicarse con el firmware del sistema embebido y para conectarse a dispositivo conectados a una red inalámbrica local utiliza el protocolo TCP/IP.

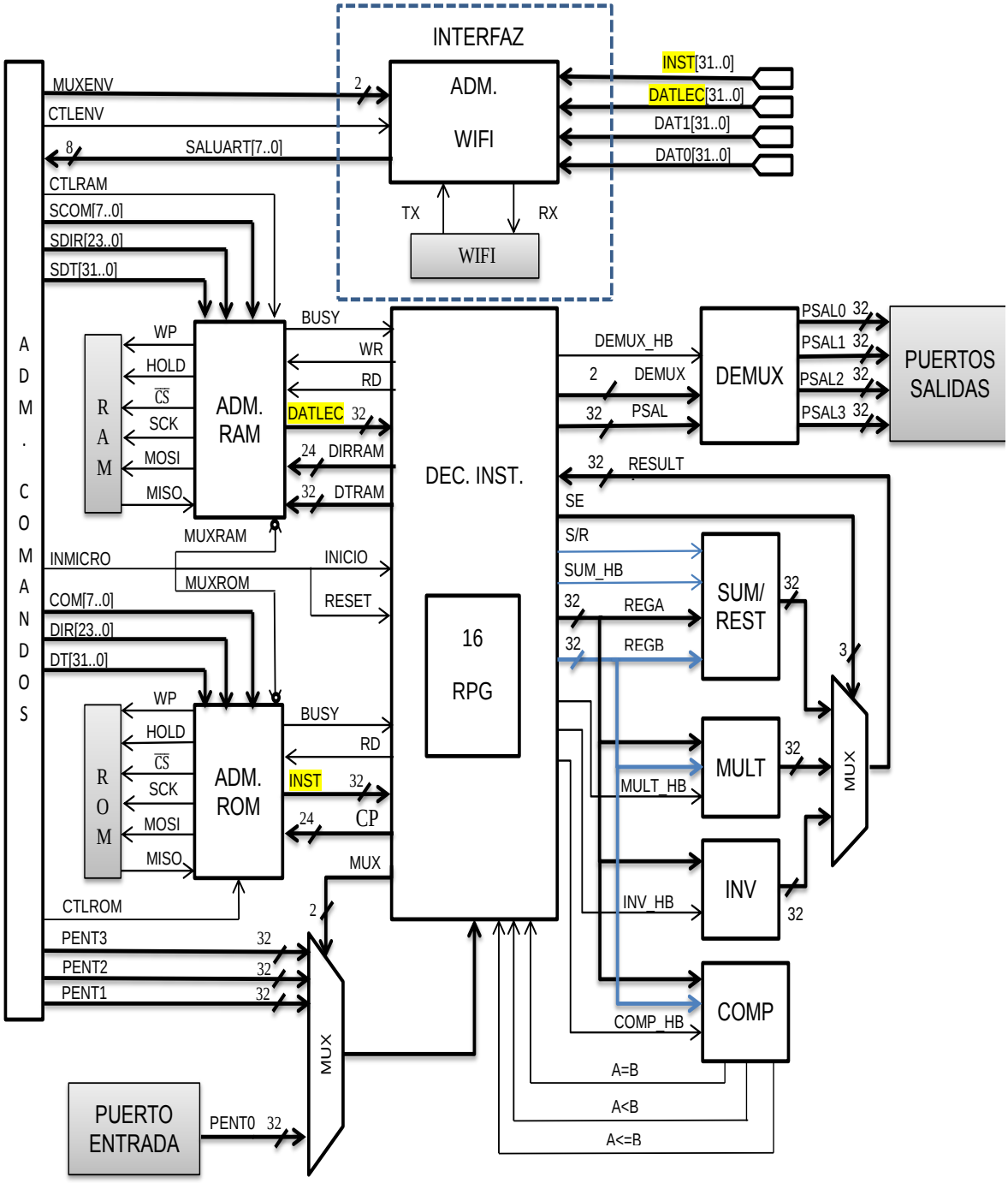


Figura 47: Diagrama general del firmware del sistema embebido.

Los bloques en gris de la figura 47, representan el hardware con el cual el firmware se comunica. Las flechas indican el flujo de datos y el tamaño del bus.

3.1 Mapeo de memoria y puertos del sistema embebido

En el firmware del sistema embebido se distinguen 2 bloques de memoria. En la figura 48. Se muestra el mapeo de memorias y puertos del microprocesador.

Memoria de programa: Almacena el programa con las instrucciones que gobiernan el sistema embebido. Es del tipo no volátil, por lo que el programa se va a mantener aunque se desconecte la alimentación. Para nuestro sistema se utiliza una memoria FLASH NOR con una capacidad de almacenamiento de 32Mb, organizada en 4 Megabytes, por lo que se requiere un contador de programa de 22 bits, para ir de la dirección "000000"H a "3FFFFF"H.

Memoria de datos: Se destina a guardar las variables y datos. Es volátil, es decir, los datos almacenados se borran cuando se desconecte la alimentación. La memoria utilizada para nuestro sistema es del tipo SRAM con un almacenamiento de 1Mb y una organización de 128 Kilobytes, por lo que se requieren 17 bits para ir de la dirección "00000"H a "1FFFF"H.

El microprocesador dispone de puertos de entrada y salida para la comunicación del microprocesador con el exterior.

Puertos de entrada: El microprocesador cuenta con un solo puerto de entrada, por lo que se utiliza un multiplexor para trabajar con más puertos, nuestro sistema se diseñó para trabajar con cuatro por lo que se requiere 2 bits para direccionar la lectura de los puertos que van de "0"H a "3"H.

Puertos de salida: El micro cuenta únicamente con un puerto de salida, por lo que se utiliza un demultiplexor para poder trabajar con más puertos. En el sistema implementado se trabajan con cuatro puertos de salidas, por lo que se requiere 2 bits para direccionar la salida a los puertos "0"H al "3"H.

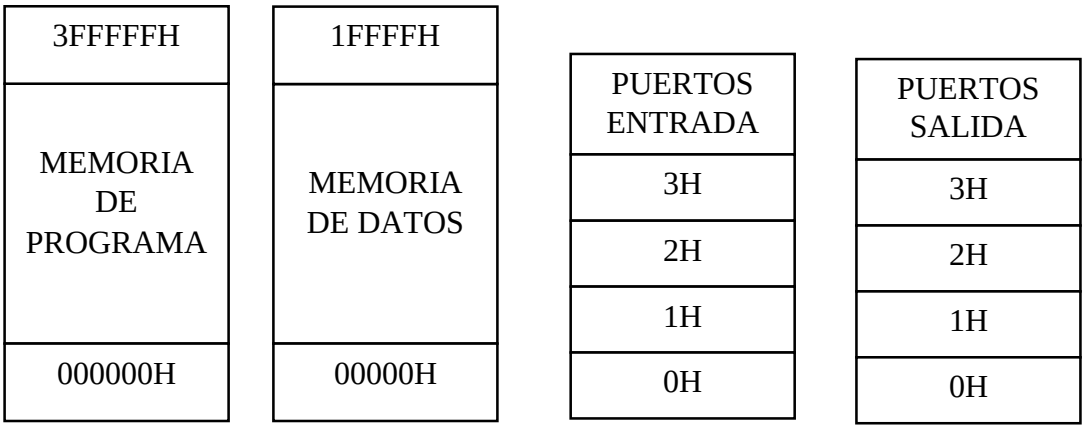


Figura 48: Mapeo de memoria y puertos del sistema embebido.

3.2. Interfaz inalámbrica WiFi

Para la comunicación entre el firmware del sistema embebido y el software de interfaz de usuario se utiliza el módulo WiFly RN-XV, el cual establece comunicación inalámbrica con el software de interfaz bajo el protocolo TCP/IP, y transfiere o recibe datos del FPGA del sistema mediante el protocolo serial asíncrono UART. Para llevar a cabo la comunicación entre el FPGA y el módulo WiFly, se utilizo el firmware administrador WiFi en el cual se implementa el protocolo de comunicación UART [19]. El formato de datos de un UART consta de 10 bits, un bit de inicio +8 bits de datos + 1 bit de paro. La señal está inactiva a un "1" lógico y activa a un "0" lógico. En la figura 49 se muestra el formato de transmisión de bits.

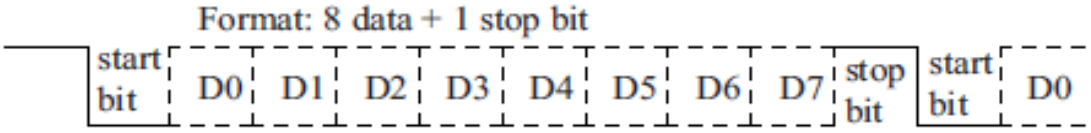


Figura 49: Formato de datos de un UART.

El protocolo UART (Universal Asynchronous Receiver Transmiter) o receptor/emisor asíncrono universal permite la comunicación serial entre dos dispositivos, ver figura 50.

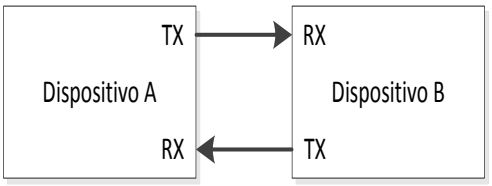


Figura 50: Comunicación UART

En la nomenclatura UART, TX es para la transición y RX para la recepción. La principal característica de este medio de transición es que no se usa una señal de reloj para sincronizar el flujo de datos, caso contrario al protocolo SPI o el USART. El hecho de no usar una señal de reloj simplifica la operación y facilita la implementación de hardware. Sin embargo se tiene un precio a pagar y este es la velocidad de transición. Puesto que no existe una señal de reloj ambos dispositivos deben ser configurados para que funcionen a un flujo de datos predeterminado. Para la recepción de datos se recomienda muestrear las señales al menos 16 veces más rápido que la velocidad de transferencia establecida, en la figura 51 se muestra el muestreo del bit de inicio.

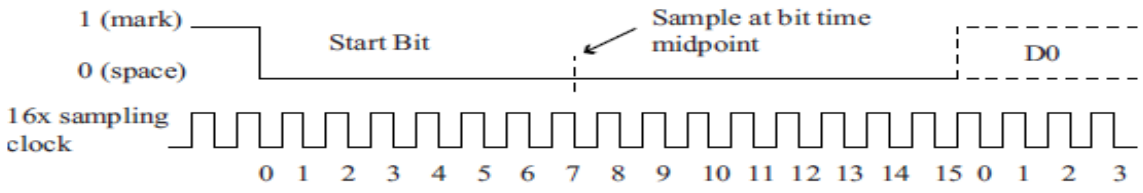


Figura 51: Muestreo de bits.

En la figura 51 se observa el muestro del bit de inicio a la mitad del tiempo que permanece en bajo, esto con la finalidad de capturar las señales en un estado estable y libre de ruido causado por las transiciones de la señal.

Al implementar el administrador WiFi en el FPGA que trabaja a una frecuencia de 100MHz, se puede lograr una frecuencia de muestre adecuada que permita la transmisión y recepción de datos de manera serial, en la figura 52 se muestran las máquinas de estado del firmware correspondiente a los módulos de transmisión y recepción de datos.

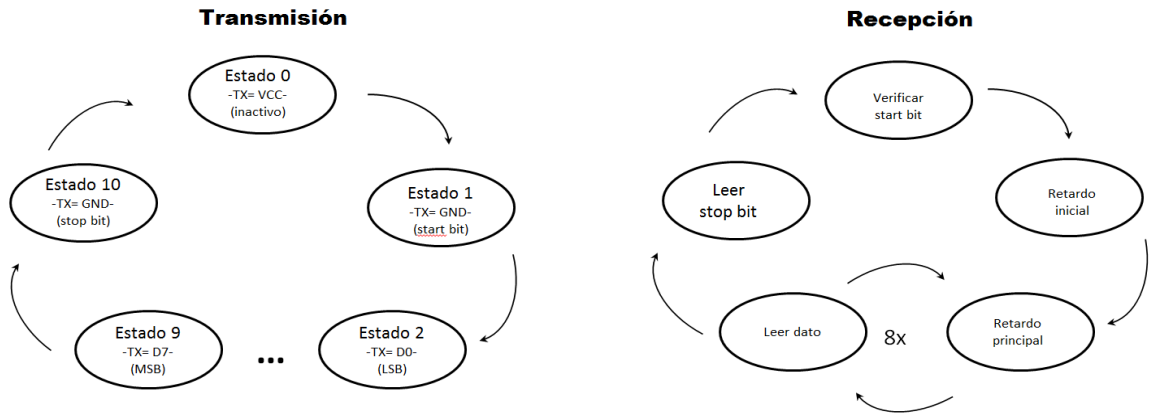


Figura 52: Maquinas de estado de transmisión y recepción de datos UART.

En la figura 53 se muestra un diagrama a bloques de los elementos que conforman la interfaz inalámbrica WiFi, como se puede observar el módulo de transmisión tiene un reloj de 460800 baudios, cuenta con una entrada de habilitación, una entrada de 8 bits y una salida OCP que indica que el modulo está ocupado. El módulo de recepción adquiere los datos de manera serial a 100MHz sincronizándose a los baudios de transmisión mediante retardos y leyendo los datos en la mitad de transmisión de cada bit, la señal de salida es el byte recibido pero en su forma paralela. Los módulos de transmisión (TX) y de recepción (RX), envían y reciben respectivamente datos de 8 bits únicamente, si se requiere trabajar con datos de 16 o 32 bits, se deberá entonces tomar 2 o 4 lecturas para formar el dato, esto en el caso de la recepción de datos. Para el caso de transmitir datos de 32 bits el dato se divide en 7 bytes donde en cada byte enviado los tres bits más significativos corresponden al identificador y los 5 bits menos significativos corresponden a una parte del dato, ver tabla 12.

Tabla 12: Identificación de bytes enviados.

No. Byte	Identificador	Dato	
1	001	[4..0]	LSB
2	010	[9..5]	.
3	011	[14..10]	.
4	100	[19..15]	.
5	101	[24..20]	.
6	110	[29..25]	.
7	111	[31..26]	MSB

Como se observa en la tabla 12, para enviar un dato 32 bits se requieren 7 bytes para enviar este dato al software, cada byte enviado debe contar con un identificador o bandera que el software utiliza para identificar y ordenar el dato recibido. Los bloques remarcados de la figura 53 corresponden a módulos del administrador WiFi.

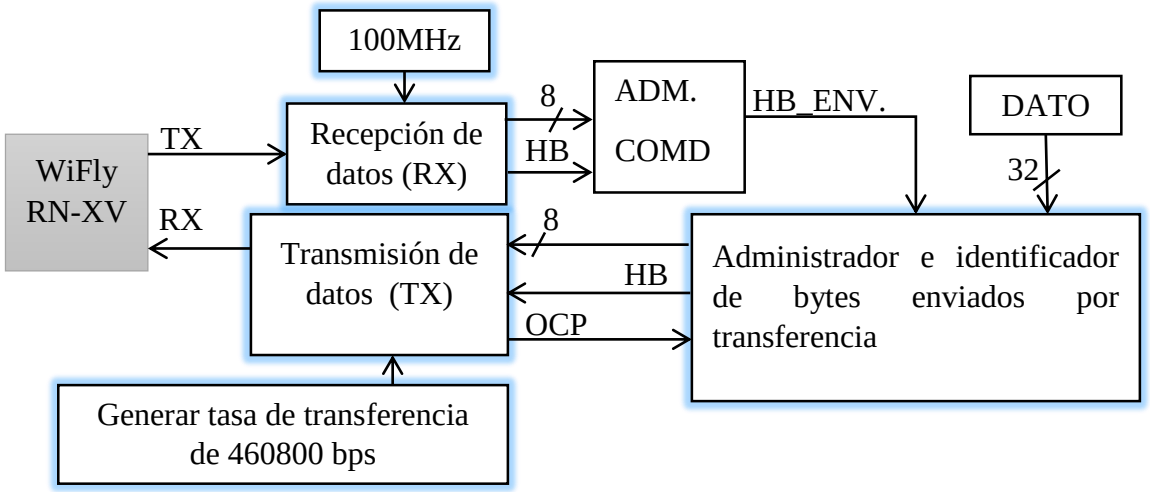


Figura 53: Diagrama a bloques de comunicación UART del sistema embebido.

3.3. Administrador de comandos

Este bloque tiene la función de leer y ordenar los bytes recibidos por el modulo receptor de datos UART (RX), la información que llega a esta parte del firmware del sistema corresponde a los comandos y datos enviados desde el software de interfaz, para realizar tareas como son la escritura o lectura de memorias desde el software, arranque y paro del microprocesador, envió de datos al microprocesador o habilitar él envió de algún dato de 32 bits hacia el software vía WiFi. Los datos enviados desde el software constan de 5 bytes donde el byte más significativo corresponde a una bandera que indica al administrador de comandos las señales a activar o hacia dónde dirigir los 4 bytes menos significativos. En la tabla 13 se muestran las banderas que el modulo maneja y la acción que le corresponde a cada una.

Tabla 13: Lista de banderas manejadas por el administrador de comandos

Bandera (BD) (hexadecimal)	Bandera (BD) (decimal)	Acción
0xF1	241	Datos a comando para ROM
0xF2	242	Dirección para ROM
0xF5	245	Dato para ROM
0xF3	243	Activa ROM
0xF4	244	Inactiva ROM
0xF9	249	Comando para RAM
0xFA	250	Dirección para RAM
0xF0	240	Dato para RAM

0xEF	239	Activa RAM
0xEE	238	Inactiva RAM
0xFB	251	Arranque de microprocesador
0xFC	252	Paro de microprocesador
0XFD	253	Inicio lectura
0xFE	255	Paro lectura.
0xF6	246	Dato a puerto PENT3
0xF7	247	Dato a puerto PENT2
0xF8	248	Dato a puerto PENT1

En la tabla 13 se muestran marcadas las banderas que el administrador de comandos identifica para generar señales de arranque y paro de los módulos administradores de memoria, microprocesador y modulo transmisor UART (TX). Para las banderas sin marcar de la tabla 13 el administrador de comandos determina si los 4 bytes menos significativos se direccionan hacia las memorias o a los puertos de entrada del microprocesador. En la figura 54 se muestra el diagrama a bloques del módulo administrador de comandos.

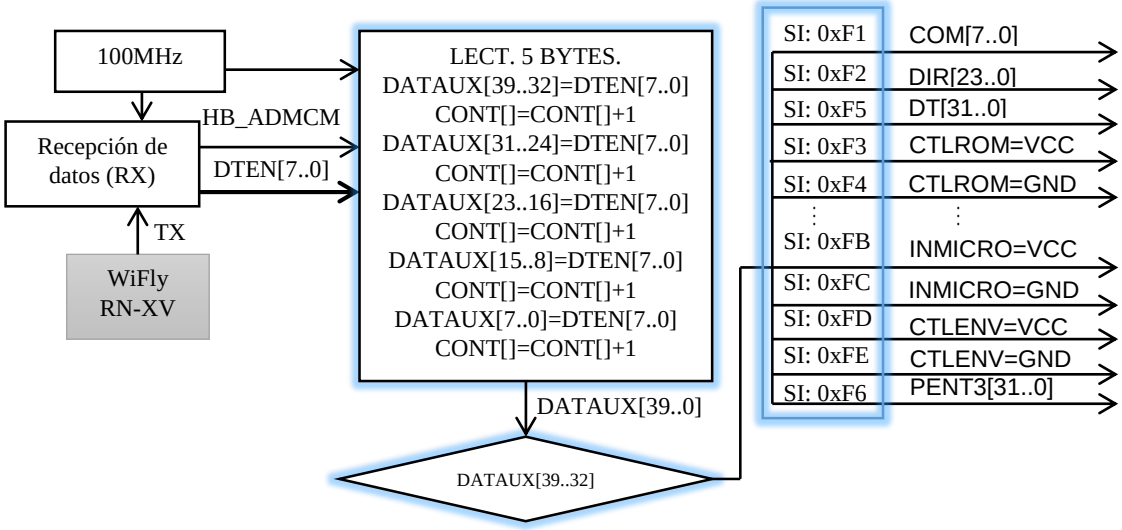


Figura 54: Diagrama a bloques de administrador de comandos.

El administrador de comandos realiza la lectura de los 5 bytes enviados desde el software, almacenando al primer byte recibido en los bits más significativos de la variable DATAUX[39..32] y almacenando al quinto byte en los menos significativos de la variable DATAUX[7..0]. Ya que se tienen cargados los 32 bits de la variable DATAUX[39..0], el administrador de comandos evalúa al byte más significativo de esta variable (bandera), para determinar si el dato se trata de un comando o un dato. El modulo administrador de comandos, junto a los módulos administradores de memoria, interfaz WiFi y software de interfaz, conforman lo que sería el sistema de grabador o programador del sistema embebido, un sistema que nos permite programar a nuestro sistema vía WiFi, sin la necesidad de un programador externo, ya que este los modulo mencionados lo conforman.

3.4 Administrador de memoria ROM

El sistema embebido desarrollado en este trabajo cuenta con una memoria externa del tipo **FLASH-NOR**, con una capacidad de almacenamiento de 32Mb que presenta una organización de memoria de 4MBytes. El circuito integrado al que nos referimos es el **SST26VF032B104I/SN** de la marca Microchip el cual se caracteriza por: operar bajo los protocolos SPI y SQI, por trabajar a frecuencias de 40Mhz, 80Mhz y 104MHz y por su encapsulado que requiere pocos pines para su conexión con el FPGA. Por las características de no volatilidad y capacidad de acceso aleatorio de alta velocidad de la memoria, esta se destina como **memoria de programa** del microprocesador, para almacenar el conjunto de instrucciones (programa de control) del sistema embebido.

3.4.1 Funcionamiento de la memoria FLASH-NOR

El SST26VF032B104I/SN opera bajo el protocolo SPI (Serial Peripheral Interface) o el protocolo de bus multiplexado SQI de 4 bits. El dispositivo inicia con el protocolo SPI, después de un encendido o reinicio y mediante una instrucción se configura al modo SQI. El flujo de datos en el modo SQI es muy similar al modo SPI, excepto que usa un bus multiplexado de cuatro señales de entrada/salida para comandos, dirección y secuencia de datos. La memoria flash opera en el MODO 0 (0,0) y MODO 3 (1,1), la diferencia entre los dos modos es el estado de la señal SCK cuando el bus maestro está en reposo y ningún dato se comienza a transferir. La señal SCK está en bajo para el MODO 0 y en alto para el MODE 3. Para ambos modos los datos seriales son muestreados en el flanco ascendente de la señal de reloj SCK para datos de entrada y en flanco ascendente para datos de salida. El protocolo tradicional SPI tiene la entrada (SI) y la salida (SO) por separado como se muestra en la figura 55. El protocolo SQI usa cuatro señales multiplexadas, SIO [3:0] para los datos de entrada y salida, como se muestra en la figura 56. Esto significa que el protocolo SQI cuadriplica la velocidad de transferencia tradicional con la misma frecuencia de reloj, sin la necesidad de más pines en el encapsulado.

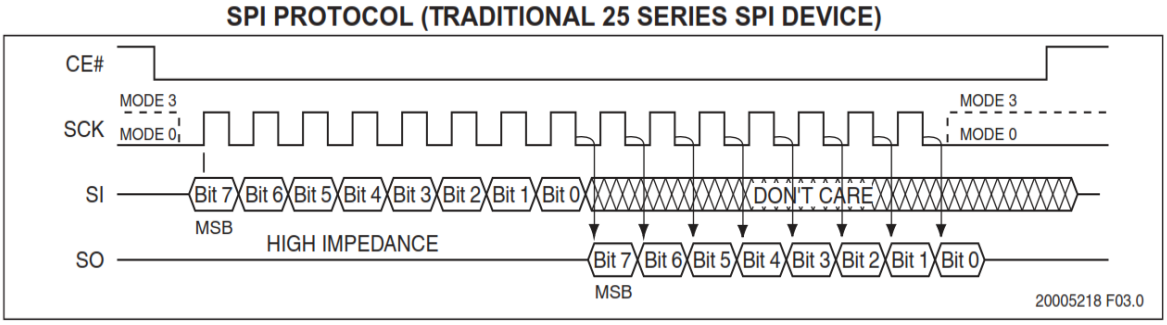


Figura 55: Protocolo SPI.

Como se observa en las figuras 55 y 56, la comunicación comienza cuando la señale CE# se pone en bajo, luego se envía el dato mientras la señal de reloj SCK muestrea sobre los flancos ascendentes para los datos de entrada, y en flanco descendente para los datos de salida, la señal CE# se mantiene en bajo hasta el fin de la comunicación.

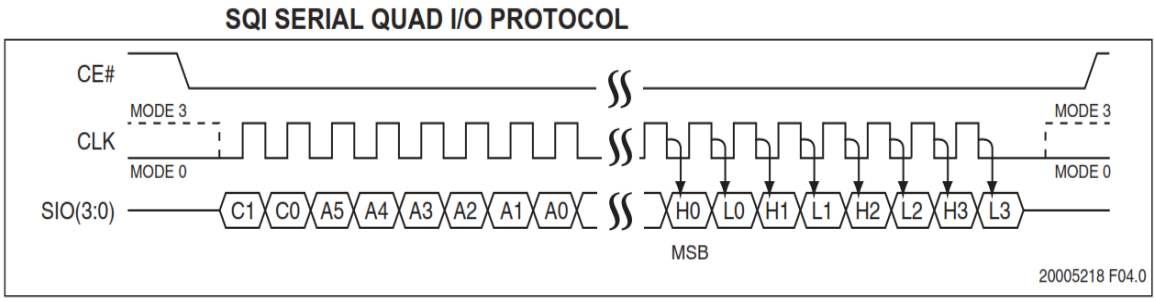


Figura 56: Protocolo SQI.

Ya que se conocen las señales que intervienen en el protocolo de comunicación SPI, se procede a revisar los tiempos en los que debe generarse cada una de las señales y la relación en tiempo que tienen entre cada una. En la figura 57 se muestra el diagrama de tiempos para las señales seriales de entrada. Y en la figura 58 el diagrama de tiempos para las señales de salida.

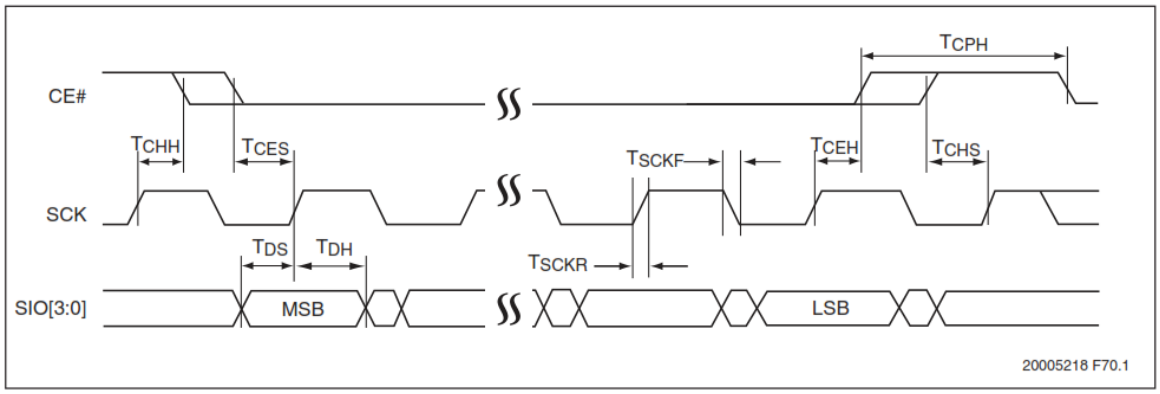


Figura 57: Diagrama de tiempos para datos seriales de entrada.

En la figura 51 se muestra un tiempo **TCES**, que determina el retardo que tendrá la señal SCK, respecto al cambio de estado de CE# y un tiempo **TCEH** que determina el retardo que tiene el cambio de estado de CE# respecto a la señal de reloj SCK.

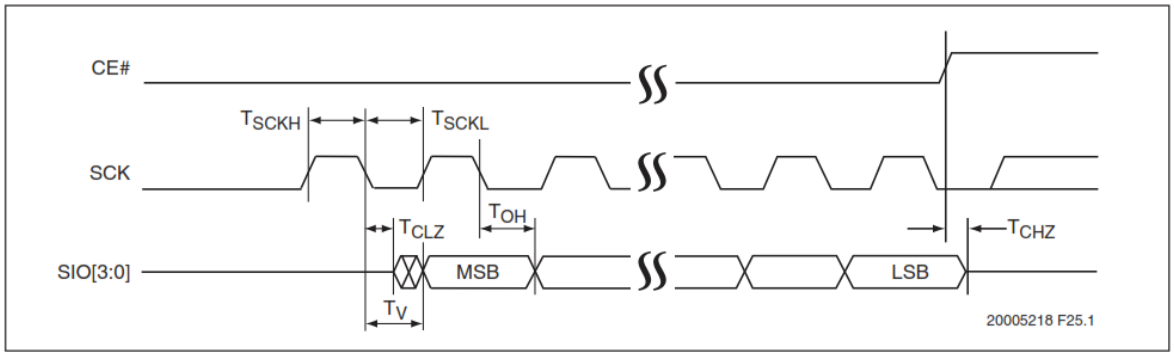


Figura 58: Diagrama de tiempos para datos seriales de salida.

En la gráfica de la figura 58 se toman en cuenta el tiempo **TOH**, este determina el tiempo que se requiere para muestrear la salida por flanco descendente y el tiempo **TCHZ** es el

retardo que tienen las señales de datos respecto a la señal de reloj SCK. En la tabla 14 se muestran los tiempos de operación, necesarios para la implementación de las secuencias de las señales CE#, SCK y SI y SO, necearías para lograr la comunicación con la memoria FLASH.

Tabla 14: Tiempos de operación de memoria FLASH.

Symbol	Parameter	Limits - 40 MHz		Limits - 80 MHz		Limits - 104 MHz		Units
		Min	Max	Min	Max	Min	Max	
F_{CLK}	Serial Clock Frequency		40		80		104	MHz
T_{CLK}	Serial Clock Period		25		12.5		9.6	ns
T_{CES}[2]	CE# Active Setup Time	8		5		5		ns
T_{CEH}[2]	CE# Active Hold Time	8		5		5		ns
T_{CPH}	CE# High Time	25		12.5		12		ns
T_{CHZ}	CE# High to High-Z Output		19		12.5		12	ns

La memoria FLASH en el sistema embebido se cableo con el FPGA para operar tanto con el protocolo SPI como con el protocolo SQI, pero en este trabajo únicamente se diseña e implementa en el FPGA un módulo administrador de memoria en base al protocolo SPI, por lo que se elige 80Mhz como frecuencia de trabajo. Por lo tanto FCLK=80MHz y los tiempos mínimos para TCES y TCEH deben ser de 5ns, y el tiempo entre él envió de un dato y otro debe ser TCPH=12.5ns. La memoria cuenta con un conjunto de instrucciones de 8 bits que permiten la operación del dispositivo, algunos tienen acción directa y otro requieren de instrucciones previas, a continuación se muestran las instrucciones utilizadas para las operaciones de borrado de memoria, lectura y escritura de datos en una localidad de memoria determinada.

3.4.2 Instrucciones de operación del dispositivo FLASH SST26VF032B104I/SN

La memoria flash, cuenta con varias instrucciones para cada uno de los protocolos soportados, las instrucciones son usadas para leer, escribir, borrar y configurar el dispositivo. La lista de las instrucciones utilizadas para la implementación del administrador de memoria se muestra en la tabla 15.

Tabla 15: Instrucciones de operación del dispositivo FLASH-NOR.

INSTRUCTION	DESCRIPTION	COMAND	MODE	MAX FREQ.
WREN	WRITE ENABLE	06H	SPI	104MHz
ULBPR	GLOBAL BLOCK PROTECTION UNLOCK	98H	SPI	104MHz
CE	ERASE FULL ARRAY	C7H	SPI	104MHz
PP	PAGE PROGRAM	02H	SPI	104MHz
HIGH-SPEED READ	READ MEMORY AT HIGH SPEED	0BH	SPI	104MHz

Como ya se mencionó hay instrucciones de operación de la memoria que ejecutan una acción por si solas y otras que requieren instrucciones previas y posteriores. A continuación se muestran las secuencias de instrucciones para realizar las operaciones de borrado, escritura y lectura de memoria.

3.4.2.1 Secuencia de instrucciones para el borrado de memoria.

1. Comando 06H // se habilita la escritura
2. Comando 98H // se desbloquea el bloque global de protecciones
3. Comando 06H // se habilita la escritura nuevamente
4. Comando C7H // borrado de memoria completo

3.4.2.2 Secuencia de instrucciones para la escritura de un dato de 32 bits a partir de una dirección indicada.

1. Comando 06H // se habilita la escritura
2. Comando 98H // se desbloquea el bloque global de protecciones
3. Comando 06H // se habilita la escritura nuevamente
4. Comando 02H // se habilita la programación de una página (256 bytes)
5. Dirección XXXXXXH // se envía la dirección de escritura (24 bits)
6. Dato XXXXXXXXH // se envía el dato a escribir (32 bits)

3.4.2.3 Secuencia de instrucciones para la lectura de un dato de 32 bits a partir de una dirección indicada.

1. Comando 0BH // habilita la lectura en alta velocidad (80Mhz a 104Mhz)
2. Dirección XXXXXXH // se envía la dirección de escritura (24 bits)
3. Dato 00000000H // se deja en ceros

3.4.3 Implementación en FPGA del módulo administrador de memoria ROM

Para leer y escribir datos en la memoria FLASH en una determinada localidad de memoria se implementan en el FPGA el modulo administrador de memoria ROM, el cual tiene la función de determinar por medio de un multiplexor si la dirección, para la lectura o escritura de la memoria se toman de las señal enviada del microprocesador o del software de la PC. Además tienen la función de generan las señales correspondientes para establecer comunicación con el circuito externo, así como de enviar los comandos para realizar los procesos de escritura, lectura y borrado de memoria. Este módulo recibe los datos, direcciones y comandos en paralelo y los convierten en un tren de pulsos que indica a las memorias la localidad en la que deben leer o escribir. Y para los datos que se leen de la memoria el módulo toma el tren de pulsos y proporciona el dato en paralelo para enviarlo al software de interfaz o al microprocesador como una instrucción de 32 bits.

En la figura 59, se muestra en bloques iluminados los módulos que conforman el administrador de memoria ROM, estos elementos son el control de SPI ROM, el módulo SPI ROM, un multiplexor y un inversor. El módulo de Control SPI ROM recibe del software de interfaz la dirección XXXXXXH, el dato XXXXXXH y únicamente los comandos C7H para indicar una operación de borrado de memoria, 02H para la escritura y

0BH para la lectura. Pero como ya se revisó en el tema anterior el único comando que se pasa directamente al siguiente bloque es el de lectura, pero para borra memoria y escribir un dato en una dirección es necesario enviar a la memoria la secuencia de instrucciones correspondiente. La función del módulo de Control SPI ROM, es la enviar la secuencia de instrucciones, una a la vez, al módulo SPI ROM para realizar las operaciones solicitadas.

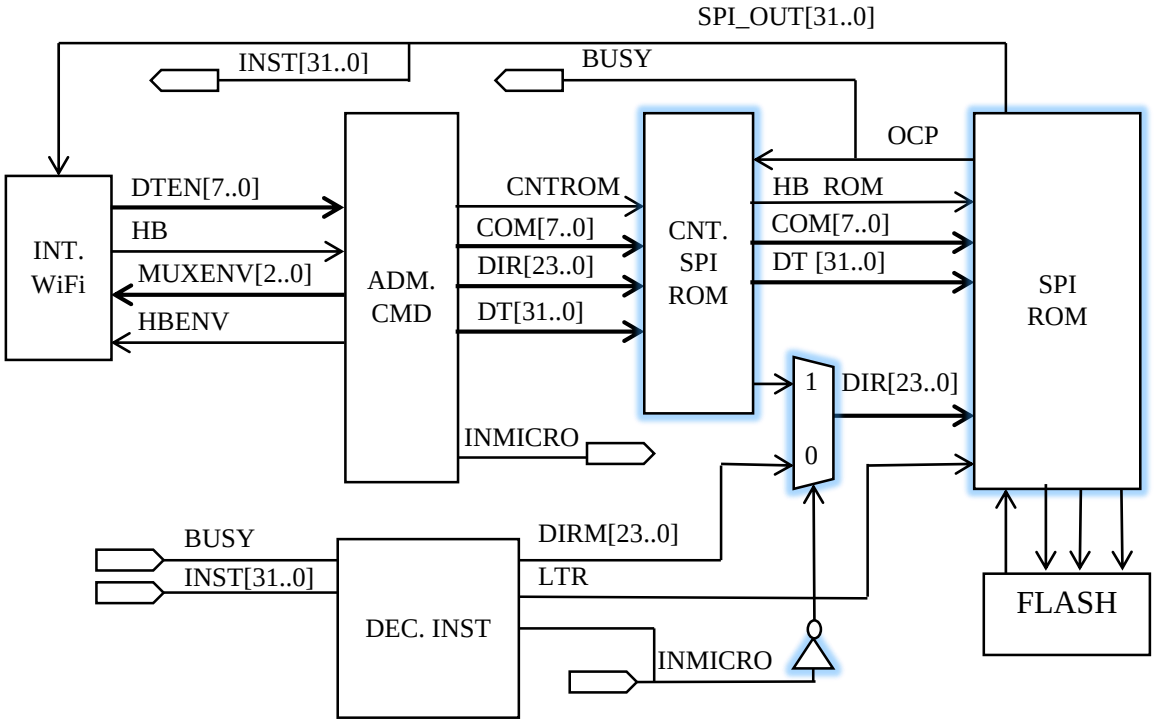


Figura 59: Diagrama a bloques de modulo administrador de memoria ROM.

En la figura 60 se muestra el diagrama de flujo del módulo Control SPI ROM, que tiene como entradas las señales CNTROM, COM[7..0], DIR[23..0], DT[31..0] y HB_ROM, el modulo se activa a partir de que la señal CNTROM se pone en alto, posterior mente se lee el contenido de la entrada COM[7..0] y entonces se envía una serie de comandos al módulo SPI ROM, en el orden que se muestra en el diagrama de flujo. Cabe mencionar que al enviar la secuencia de instrucciones se maneja la señal de ocupado del módulo SPI ROM, con el fin de verificar que esté listo antes de enviarle la siguiente instrucción.

El multiplexor y el inversor tienen la función de seleccionar si la dirección va ser tomada del decodificador de instrucciones o del dato que proviene del software de interfaz. Cuando la señal de inicio de microprocesador se pone en alto el multiplexor toma la dirección del decodificador de instrucciones y cuando se inicia el sistema o se detiene el micro, el multiplexor toma la dirección que viene del administrador de comandos.

El módulo SPI ROM, es el modulo en el que se implementa el protocolo de comunicación serial. El cual recibe los datos de comando, dirección y dato en paralelo y se encarga de serializar en base al protocolo SPI para establecer comunicación con la memoria FLASH-NOR. El protocolo implementado permite la lectura y escritura de datos de 4 bytes, esto debido a que se manejan en este sistema embebido instrucciones y datos de 32 bits. Lo que significa que para cada dato de 32 bits se requieren 4 localidades de memoria y que la

dirección de cualquier dato siempre será múltiplo de 4 por ejemplo 000038H, 00003cH, etc. En el firmware implementado se utiliza la configuración Big-Endian [24], en esta configuración el byte 0 es el más significativo, lo que se acaba de mencionar se le conoce como restricciones de alineamiento. En la figura 61 se muestra diagrama de flujo del SPI ROM.

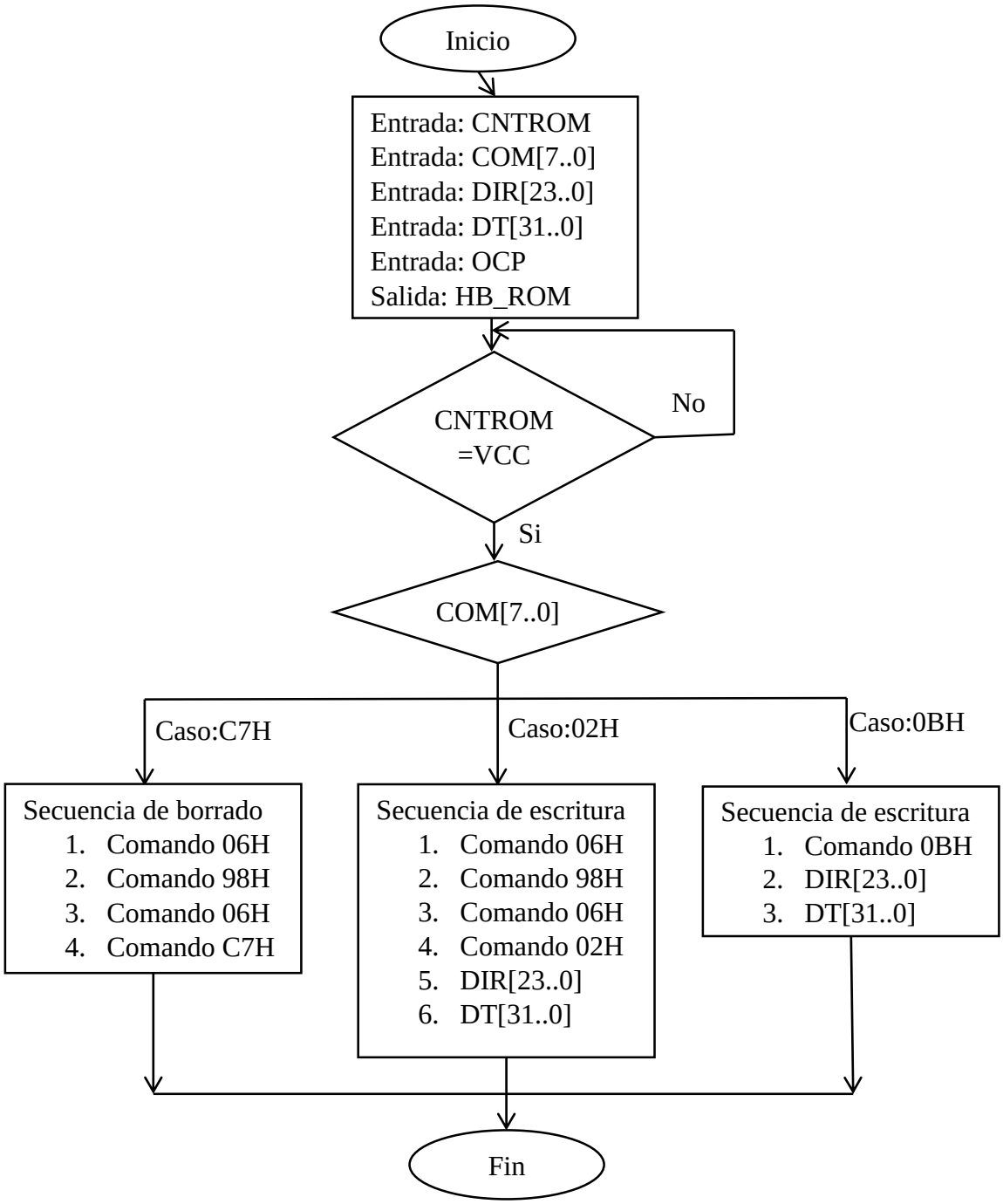


Figura 60: Firmware del módulo Control SPI ROM

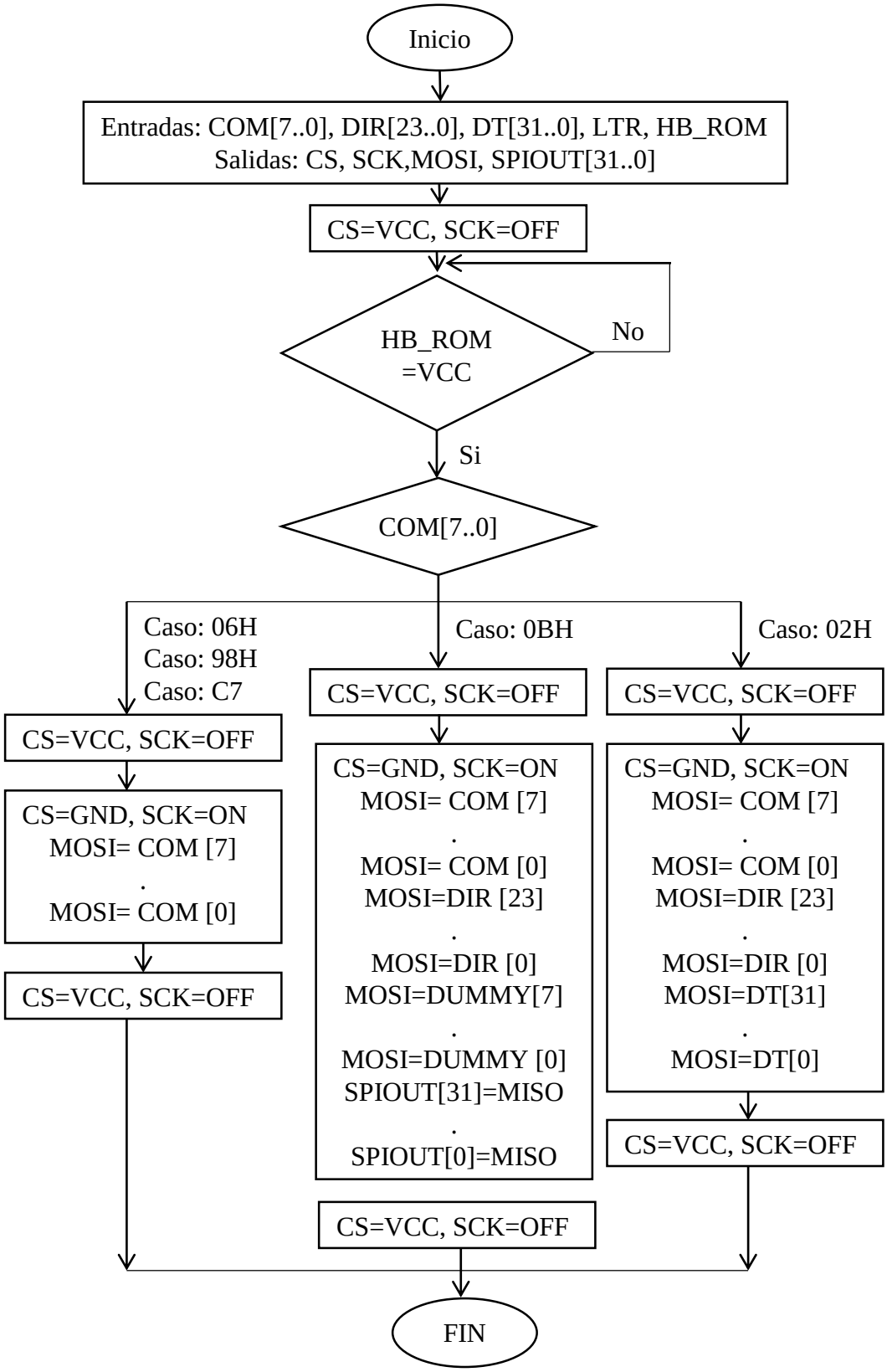


Figura 61: Firmware del módulo SPI ROM.

El módulo SPI ROM evalúa la entrada COM[7..0] para determinar las señales serializar y la secuencia en la que se tienen que enviar los datos de dirección y datos a la memoria Flash. Como se observa él envió de datos comienza con las señales de CS en bajo y las señal de reloj activa para sincronizar el envío y recepción de datos, y la transmisión de datos termina con el apagado del reloj y CS en alto.

Ya que se cuenta con todos los datos, sobre los tiempos de operación de las señales que conforman el protocolo SPI, se procede a implementar el diagrama de la figura 62, al FPGA con un reloj de 100MHz, por lo que cada estado tomará un ciclo completo de reloj, y considerando que cada estado representa un bajo o alto de la señal de reloj, entonces la frecuencia a la que se trabajara la memoria será de 50MHz, que está dentro del rango operación de la memoria flash. El bloque de "COMANDO" proporciona una constante hexadecimal igual a 9F, que representa el comando JEDEC-ID, el bloque SQI genera las señales de SCK, SIO y SO que se muestran en la figura 62, además de poner en alto a los pines WP y HOLD mediante máquinas de estados. Para este caso las señales WP, HOLD se deben mantener en alto para la correcta ejecución de la instrucción JEDEC-ID. Empleando el analizador lógico que proporciona Quartus II se lograron los resultados, mostrados en la figura 63.

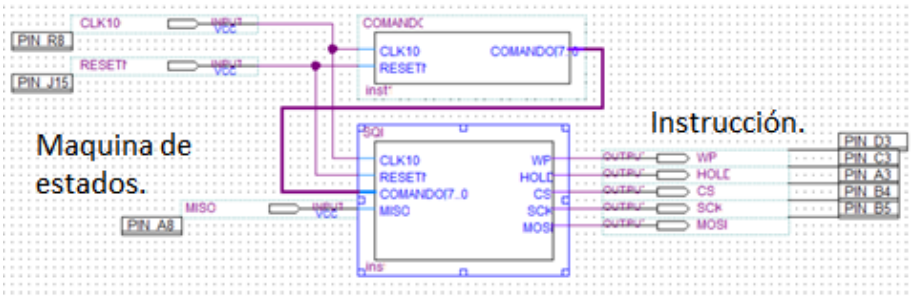


Figura 62: Firmware que ejecuta la instrucción JDEC-ID

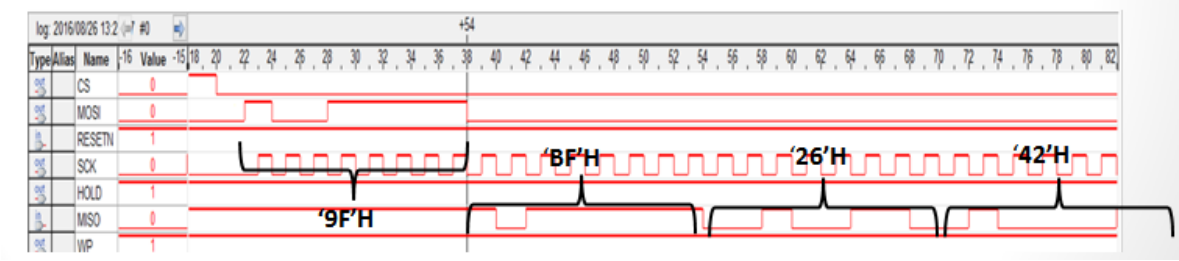


Figura 63: Resultados de la ejecución de la función JEDEC-ID

Como se observa en la figura 63, la señal CS se mantiene en bajo por el tiempo que dura la comunicación, por el canal MOSI entre los marcadores 22 a 38 se manda el comando '9F'H y en la línea MISO se lee el primero byte entre los marcadores 38-54 un valor de 'BF'H y entre los marcadores 54-70, se lee el segundo byte con '26'H y entre los marcadores 54-82 el tercer byte con '42'H. Se mencionó anterior mente esta instrucción es la más sencilla de implementar ya que no requiere de 3 bytes de dirección.

3.5 Administrador de memoria RAM

El sistema embebido cuenta con una memoria externa del tipo SRAM, con una capacidad de almacenamiento de 1 Mb, organizada en 128Kbytes. El circuito es el **23LC1024-I/SN** también de la marca Microchip que se caracteriza por operar bajo los protocolos SPI y SQI, una frecuencia de trabajo de 20MHz. Esta memoria ofrece un tiempo de acceso a datos cero y la posibilidad de grabar un número infinito de veces en sus registros por lo que para este trabajo se ocupa como memoria de datos del microprocesador embebido en el FPGA. A continuación se explica el funcionamiento de la memoria volátil.

3.5.1 Funcionamiento de la memoria SRAM

La memoria RAM, tiene como característica principal, un tiempo de acceso cero, una capacidad de almacenamiento de 1Mbit, para la operación de este dispositivo se consideran los diagramas de la figura 64 y 65.

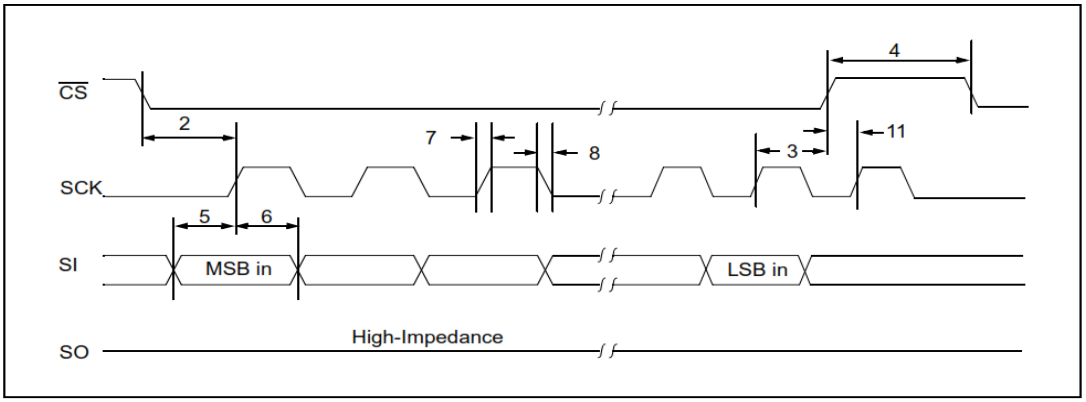


Figura 64: Diagrama de tiempo de datos de entrada (SRAM)

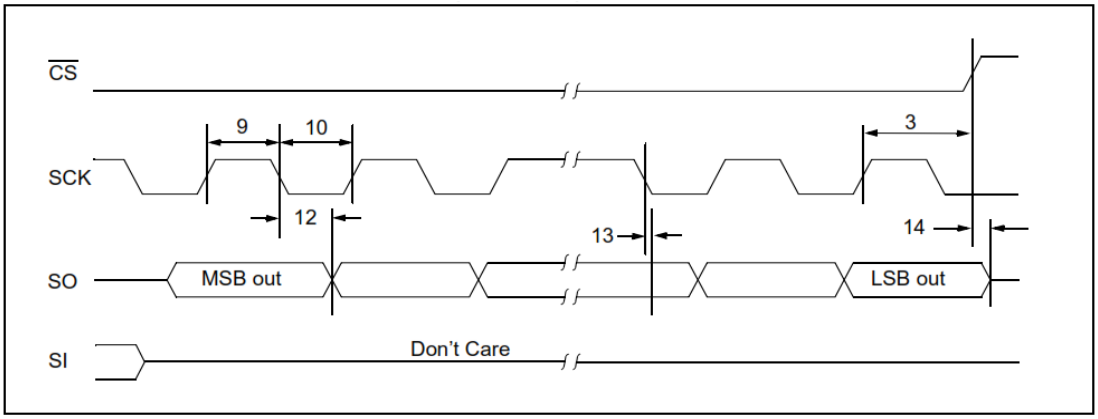


Figura 65: Diagrama de tiempos de salida (SRAM)

Como se observa en las figuras 64 y 65, las secuencias son similares a las de la memoria FLASH, lo único en lo que varían son los tiempos ya que esta memoria opera a 20MHz

A la memoria se accede a través de protocolo SPI, las señales que requiere son una entrada de reloj (SCK), un pin de datos de entrada (SI), y una línea de salida de datos (SO). El

acceso es controlado por medio de una entrada chip select (CS). Adicional mente la memoria opera con los protocolos SDI, SQI por si se requiere más velocidad en las aplicaciones. De las figuras 64 y 65 se toman en cuenta los tiempos que corresponden a los números 2, 3, 5 y 9, que son los necesarios para la correcta ejecución de las instrucciones y el correcto muestreo de las señales de salida. En la figura 66 se muestran los tiempos de operación para la memoria SRAM.

AC CHARACTERISTICS			Industrial (I): TA = -40°C to +85°C Automotive (E): TA = -40°C to +125°C			
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Max.**	**Units**	**Test Conditions**
1	FCLK	Clock Frequency	—	20	MHz	I-Temp
				16	MHz	E-Temp
2	TCSS	$\overline{CS}$ Setup Time	25	—	ns	I-Temp
			32	—	ns	E-Temp
3	TCSH	$\overline{CS}$ Hold Time	50	—	ns	
4	TCSD	$\overline{CS}$ Disable Time	25	—	ns	I-Temp
			32	—	ns	E-Temp
5	TSU	Data Setup Time	10	—	ns	
6	THD	Data Hold Time	10	—	ns	
7	TR	CLK Rise Time	—	20	ns	**(Note 1)**
8	TF	CLK Fall Time	—	20	ns	**(Note 1)**
9	THI	Clock High Time	25	—	ns	I-Temp
			32	—	ns	E-Temp

Figura 66: Tiempos de operación de memoria SRAM.

3.5.2 Instrucciones de operación del dispositivo SRAM 23LC1024-I/SN

La memoria SRAM cuenta con un conjunto de instrucciones reducido, en este trabajo solo se maneja el protocolo SPI por lo que mencionaremos únicamente las instrucciones utilizadas. Para este dispositivo las instrucciones de lectura y escritura ejecutan la operación sin la necesidad de manejar otras instrucciones. A continuación se muestra las instrucciones involucradas para realizar la lectura y escritura de datos.

Tabla 16: Instrucciones de operación del dispositivo SRAM.

INSTRUCTION NAME	INSTRUCTION FORMAT	HEX CODE	DESCRIPTION
READ	0000 0011	03H	READ DATA FROM MEMORY ARRAY BEGINNING AT SELECTED ADDRESS
WRITE	0000 0010	02H	WRITE DATA FROM MEMORY ARRAY BEGINNING AT SELECTED ADDRESS
RDMR	0000 0101	05H	READ MODE REGISTER

3.5.2.1 Secuencia para la lectura de un dato de 32 bits

1. Comando 03H // instrucción de lectura
2. Dirección XXXXXXH // se envía la dirección en la que se realizara la lectura
3. Dato 00000000H // se deja en ceros

3.5.2.2 Secuencia para la escritura de un dato de 32 bits

1. Comando 02H // se habilita la escritura
2. Dirección XXXXXXH // se envía la dirección en la que se realizara la escritura
3. Dato XXXXXXXXH // se envía el dato a escribir en la dirección indicada 32 bits.

3.5.2.3 Secuencia para la lectura del registro MODE

1. Comando 05H //instrucción de lectura del registro MODE

3.5.3 Implementación en FPGA del módulo administrador de memoria RAM

Para leer y escribir datos en la memoria SRAM en una determinada localidad de memoria se implementan en el FPGA el modulo administrador de memoria RAM, el cual tiene la función de determinar por medio de multiplexores, si las direcciones y datos, para la lectura o escritura de la memoria se toman de las señales enviadas del microprocesador o del software de la PC. Este módulo se compone de un inversor, dos multiplexores y el módulo SPI RAM. El inversor tienen la función de indicar a los multiplexores tomen la dirección y el dato del administrador de comandos siempre que el microprocesador este desactivado, al estar activo el microprocesador, este proporcione el dato y la dirección. El módulo SPI RAM, es el mismo que se ocupa para establecer comunicación con la memoria ROM, solo que con algunos ajustes, para ajustar el modulo a la memoria RAM. En la figura 67 se muestra en bloques marcados los componentes del módulo administrador de memoria RAM.

En el módulo administrador de memoria RAM, no tenemos un generador de secuencias como en el administrador de ROM, ya que las instrucciones del dispositivo SRAM, no requieren de instrucciones previas para realizar un proceso de escritura de datos. Debido a esto se conecta la señal de salida del administrador de comandos COM[7..0] directamente al módulo SPI RAM, la señales DIR[23..0] y DT[31..0] pasan por los multiplexores. Para permitir que la memoria sea operada desde el software o desde el microprocesador, dependiendo esto de la señal INMICRO que inicia o para la ejecución de un algoritmo en microprocesador.

Los datos almacenados en esta memoria son de 32 bits por lo que se manejaran las mismas restricciones de alineamiento que se manejaron para la memoria FLASH.

Para la comunicación entre el decodificador de instrucciones y la memoria SRAM, el modulo cuenta con señales de entrada LTR y ESC (lectura y escritura) que agilizan la solicitud de dichas operaciones. El módulo SPI RAM tiene una señal de salida OSC (ocupado) que cuando está en bajo, le indica al micro está listo para realizar una nueva lectura o escritura de datos.

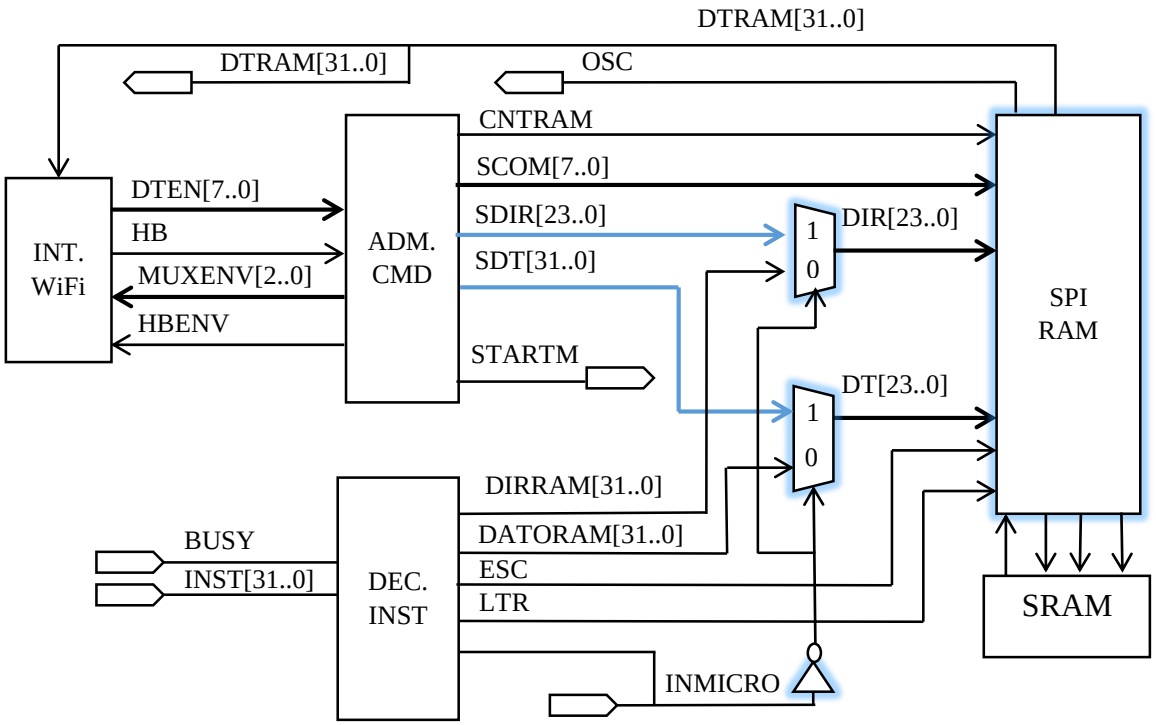


Figura 67: Diagrama a bloques del módulo administrador de memoria RAM.

El dispositivo de memoria SRAM, tiene tres modos de operación que son seleccionadas por los bits 7-6 del registro MODE, los modos de operación son BYTE, PAGE, Y SECUENCIAL. A continuación, se muestran los resultados de la ejecución de lectura del registro MODE, utilizando el diagrama a bloques de la figura 68.

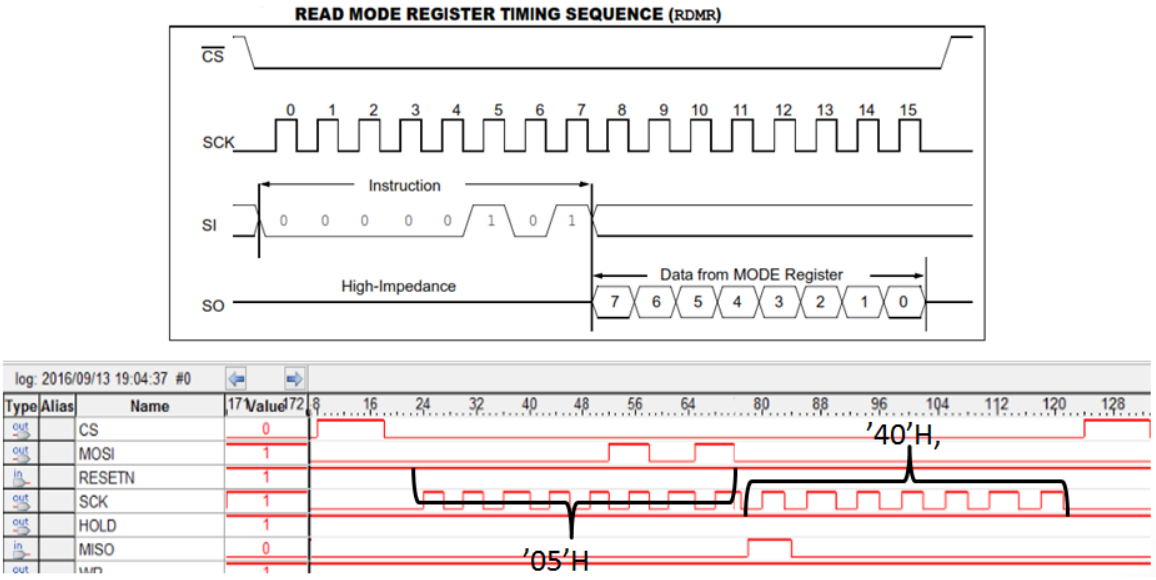


Figura 68: Resultados de la ejecución de instrucción RDMR.

Como se puede observar se envía el comando ’05’H y el valor que devuelve la memoria es “40”H, que corresponde al valor por default del registro MODE.

3.6 Puertos de entrada y salida

Los puertos de entrada y salida son el medio de comunicación del sistema embebido con el exterior. El firmware del sistema embebido se implementan 4 puertos paralelos de entrada y 4 puertos paralelos de salida cada uno de 32 bits. Los puestos de entrada se conectan a un multiplexor con el cual el decodificador de instrucciones permite el paso y procesamiento de un bus a la vez. Los puertos de salida se obtienen a partir de un demultiplexor que permite el direccionamiento de un único puerto de salida del decodificador de instrucciones hacia cuatro salidas distintas. El multiplexor utilizado corresponde a un IP de Altera [12], la finalidad de ocupar estos modulo es la utilizar los recursos que ofrece Quartus 13.0 y la de agilizar el desarrollo del sistema embebido. El multiplexor es controlador por el decodificar de instrucciones en el momento en que se procesa una instrucción de lectura de puerto. La salida MUX[1..0], lleva la dirección del puerto a leer. El demultiplexor igual que la mayoría de módulos del firmware del sistema fueron programados en AHDL, ya que es un lenguaje sencillo de aprender y que aprovecha completamente la arquitectura interna de los dispositivo lógicos programables de ALTERA. Este bloque cuenta con una señal de habilitado que activa al DEMUX para cargar el dato de entrada PSAL[31..0] al puerto de salida que indique el bus DEMUX[1..0]. El multiplexor tiene conectado un reloj de 100MHz, por lo que direccionar un dato solo le toma 10ns, lo cual no afecta para la lectura de datos de alta frecuencia. El demultiplexor está conectado también a un reloj de 100MHz y a la señal de reset del decodificador lo que permitirá no quede activa ninguna señal al momento de detener la ejecución de un algoritmo. Como se observa en la figura 69 tenemos 3 puertos de entrada conectados al administrador de comandos, lo cual nos permite enviar parámetro o datos directamente el microprocesador aun cuando esté trabajando. Cabe mencionar el dato más importante de los puertos de entrada y salida es que únicamente trabajan con números en punto flotante de simple precisión IEEE754. Para trabajar con otro tipo de datos, se tendrá que implementar en el FPGA circuitos adicionales para procesos de conversión u ocupar algún IP de ALETRA disponible para conversión de datos.

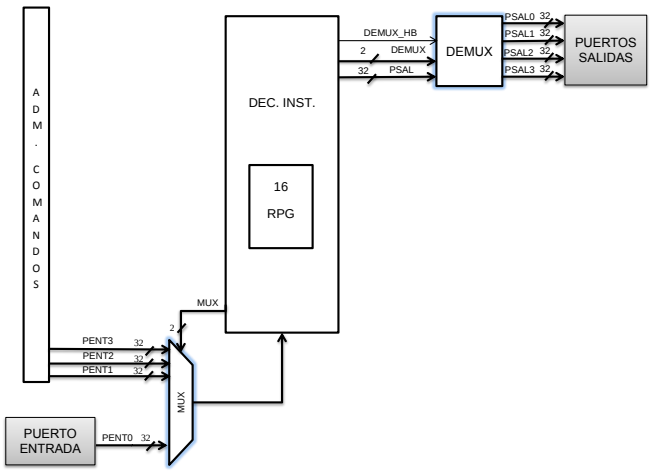


Figura 69: Firmware implementado para puertos de entrada y salida.

3.7 Microprocesador de 32 bits

En esta sección se presenta el diseño e implementación en FPGA de un microprocesador RISC de 32 bits destinado al control y automatización de sistemas. Entre las principales características del microprocesador esta la arquitectura Harvard, con memoria externa de programa de 1M x 32 bits y una memoria externa de datos de 32K x32 bits, un banco de 16 registros de propósito general de 32 bits cada uno, una unidad lógico aritmética para la ejecución de instrucciones aritméticas y de comparación con números en punto flotante de simple precisión y para comunicar el microprocesador con el exterior tenemos cuatro puertos de entrada y cuatro de salida. La mayoría de los módulos fueron desarrollados utilizando el Lenguaje de Descripción de Altera AHDL, ya que para la implementación de multiplexores y módulos de cálculo de suma, resta multiplicación, inversión y comparación de datos en punto flotante de simple precisión se ocuparon módulos IP de Altera, esto con el fin de agilizar el desarrollo del firmware del sistema. En la figura 70 se muestra en bloques marcados los elementos que conforman al microprocesador.

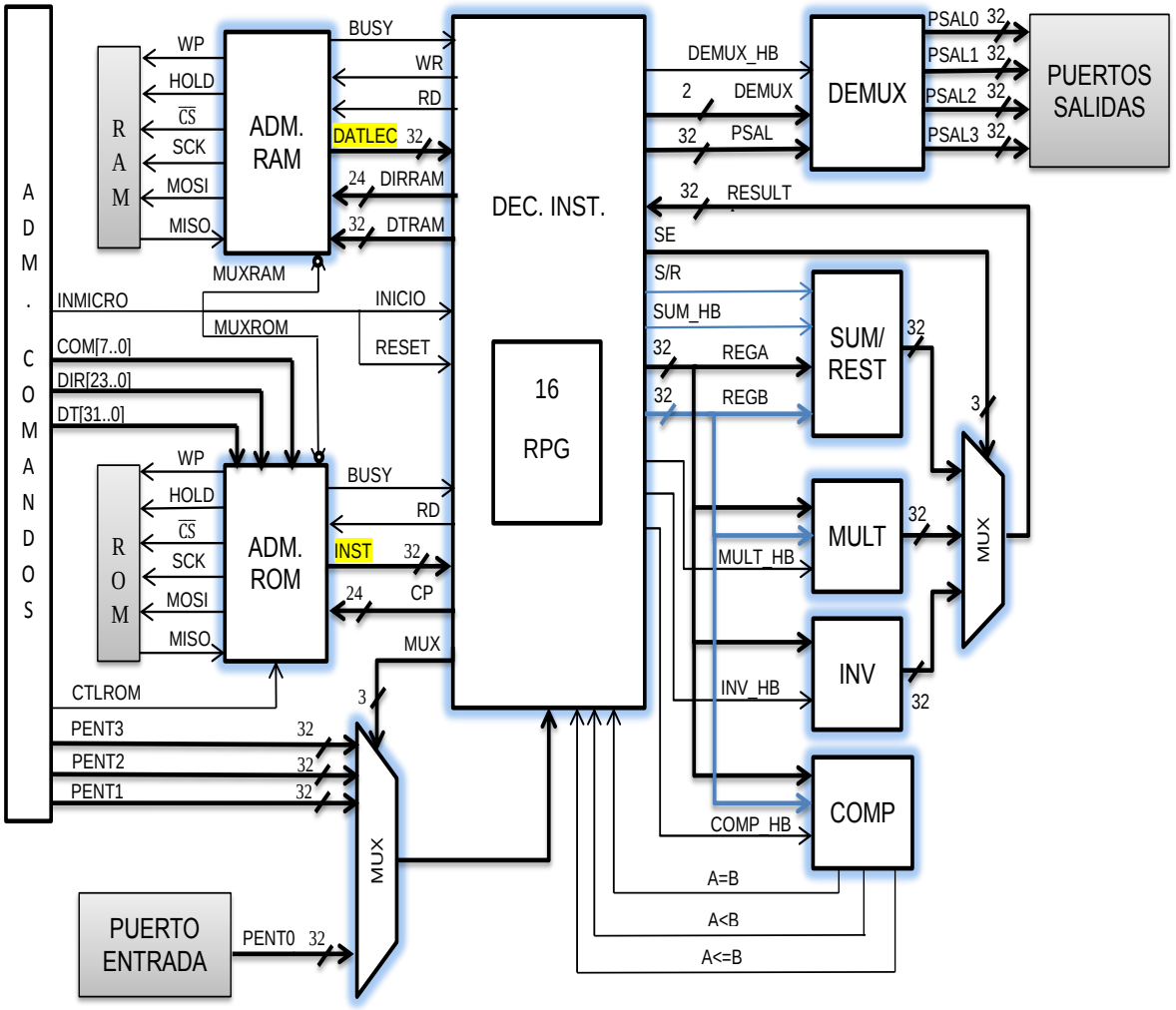


Figura 70: Diagrama a bloques del microprocesador embebido.

3.8 Señales de reloj

El firmware del sistema embebido opera con tres señales de reloj distintas, 80MHz, 100MHz y 20MHz. Para generar estas señales de reloj se utilizó un PLL (Phase-Locked Loops), que se configuro con una frecuencia de 50MHz que es la frecuencia del oscilador con la que cuenta la tarjeta DEO-NANO. El decodificador de instrucciones, la unidad lógico aritmética, administrador de comandos y puertos de entrada y salida operan a una frecuencia de 100MHz, el administrador de memoria ROM opera a 80MHz y el administrador de memoria RAM opera a 20MHz. Este circuito requiere el uso de un inversor para le señal de reset.

3.9 Unidad lógico aritmética

La unidad lógico aritmética se implementó utilizando módulos IP de altera que realizan operaciones de suma/resta, multiplicación, inversión y comparación de números en punto flotante de simple precisión, su función es la de realizar los cálculos que las instrucciones aritméticas soliciten así como también la de realizar la comparación de dos numero para ejecución de instrucciones condicionales. En la figura 71 se muestra el diagrama a bloques de la unidad lógico aritmética implementada.

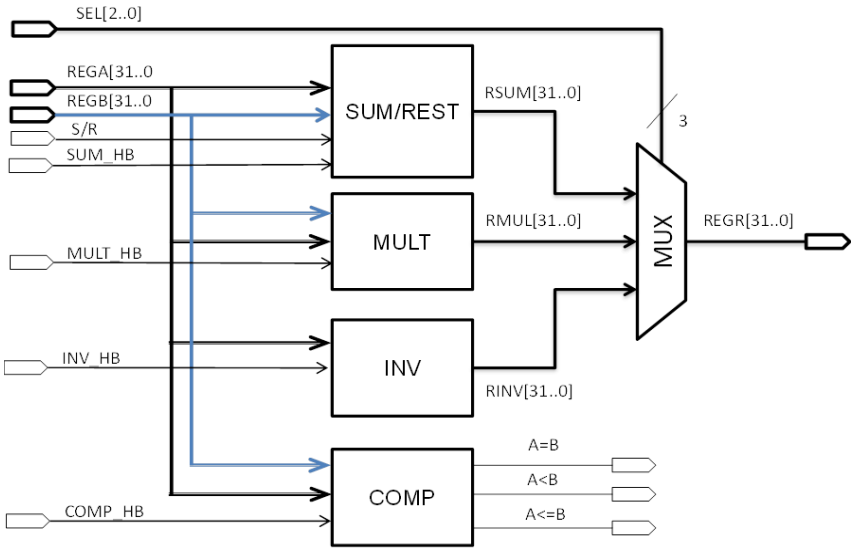


Figura 71: Diagrama a bloques de unidad lógico aritmética.

Como se muestra en la figura 71, tenemos una señal de habilitado para cada módulo, esto con el fin activar únicamente la operación que solicite cada instrucción aritmética, y de misma forma para las instrucciones de condicionales, tenemos una señal de habilitado en el comparación. Las salidas de las operaciones aritméticas se conectan a un multiplexor que es controlado por el decodificador de instrucciones, para obtener el resultado de la operación solicitada. El modulo comparador toma lectura de las señales REGA y REGB, para determinar si A=B, A<B o A<=B, según sea el resultado se activa el bit de salida correspondiente y se envía al decodificador de instrucciones para su procesamiento. Cada módulo se configuro para optimizar elementos lógicos y realizar las operaciones en un número de ciclos mínimo.

3.10 Decodificador de instrucciones

El microprocesador embebido tiene como núcleo al decodificador de instrucciones, ya que se encarga de leer y decodificar las instrucciones de la memoria de programa, generar la dirección de la siguiente instrucción y generar las señales de control del resto de elementos que conforman al sistema para la ejecución de instrucciones, este se explica en conjunto con las instrucciones.

3.10.1 El conjunto de instrucciones

Para el desarrollo del microprocesador del sistema embebido se diseñó un conjunto de instrucciones basadas en MIPS, esto con el objetivo de manejar instrucciones de tamaño fijo y un reducido número de formatos. Las instrucciones diseñadas para este trabajo son de 32 bits y se clasifican en instrucciones aritméticas (tipo R), de transferencia (tipo I) y de salto (tipo J).

3.10.2 Instrucciones aritméticas

Esta instrucciones se componen de seis campos, el primer campo de 6 bits que corresponden al código de operación cuya utilidad es la de indicar el tipo de instrucción. A continuación 10 bits destinados a los registros fuente, 5 de esos bits son para la primera fuente (RS) y los 5 para la segunda (RT) 5 bits de campo destino o RD, que nos indica el registro en el que se va a almacenar el resultado de la operación, 5 bits de shamt o desplazamiento que para estas instrucciones no es utilizado y para finalizar tenemos 5 bits para el campo function (función), que indicara que hacer con esos valores. En la tabla 17 se muestran las instrucciones aritméticas implementadas para desarrollo del microprocesador.

Tabla 17: Instrucciones Aritméticas (TIPO R)

INST.	COD. OP	REGISTRO FUENTE 1	REGISTRO FUENTE 2	REGISTRO DESTINO	SHAMT	FUNCT
		RS	RT	RD	SHAMT	FUNCT
ADD.S	17d	XXXXX	XXXXX	XXXXX	00000	000000
SUB.S	17d	XXXXX	XXXXX	XXXXX	00000	000001
MUL.S	17d	XXXXX	XXXXX	XXXXX	00000	000010
INV.S	17d	XXXXX	00000	XXXXX	00000	000011
ADD.I	17d	XXXXX	XXXXX	XXXXX	00000	000100
SUB.I	17d	XXXXX	XXXXX	XXXXX	00000	000101
	6	5	5	5	5	6
BITS	31-26	25-21	20-16	15-11	10-6	5-0

El código de instrucción para las instrucciones tipo R, corresponde al valor b”010001”, que en decimal es d”17”, para diferenciar entre las operaciones de suma, resta, multiplicación, inversión de números en punto flotante de simple precisión y las operaciones de suma y resta con números enteros utilizamos distintos valores para el campo función, el cual le indica la operación a realizar, el campo función tiene la combinación binaria b”000000” para la suma, b”000001” para la resta, b”000010” para la multiplicación, b”000011” para el

cálculo del inverso de un número, b"000100" para el cálculo de la suma de números y b"000100" para el cálculo de la resta de números enteros. Como se observa en la tabla 16, tenemos una instrucciones que trabajan con solo un registro fuente, esta es INV.S para el resto utilizamos ambos registro fuente, los campos de los registros fuente y destino son de 5 bits lo que nos permite direccionar hasta 32 registros, en nuestro microprocesador solo se cuentan con 16 registros de propósito general por lo que para el buen funcionamiento del microprocesador estos campos solo deben indicarse en rangos de b"00000" a b"001111". El campo shamt no se ocupa en ninguna instrucción del microprocesador diseñado. En la figura 72 se muestra un diagrama a bloque de lectura, decodificación y ejecución de instrucciones aritméticas.

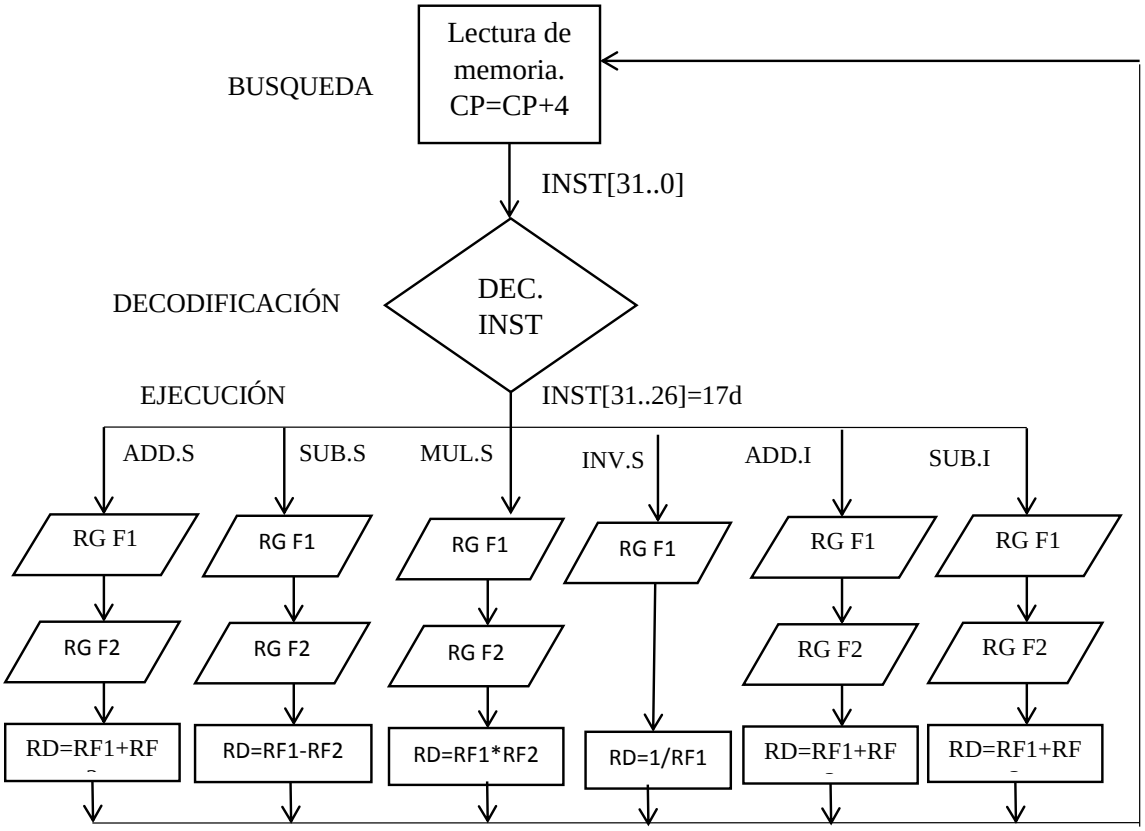


Figura 72: Diagrama a bloques de lectura, decodificación y ejecución de instrucciones aritméticas.

Las instrucciones aritméticas operan únicamente con registros de propósito general, para realizar cálculos con algún valor almacenado en memoria o que esté presente en algún puerto de entrada, estos deben copiarse a alguno de los 16 registros disponibles, el microprocesador diseñado a comparación con otros trabajos, este trabaja con registros de propósito general, y no un acumulador como la mayoría de microprocesadores comerciales, esto permite almacenar datos o resultados de una operación en cualquier RPG. A continuación se presenta la implementación a bloques de la memoria ROM, decodificador

de instrucciones y la unidad lógico aritmética que son los elementos que interactúan para la ejecución de instrucciones aritméticas.

3.10.3 Instrucciones de trasferencia

Esta instrucciones se componen de seis campos, el primer campo de 6 bits que corresponden al código de operación, luego tenemos 5 bits destinados al registros fuente (RS), seguido tenemos 5 bits que no usamos, después 5 bits para definir el campo destino o RD, que nos indica el registro al que se va copiar el contenido del registro fuente (RS), luego tenemos 5 bits de shamt o desplazamiento que para estas instrucciones no es utilizado y para finalizar tenemos 5 bits para el campo función (función). En la tabla 18 se muestran las instrucciones aritméticas implementadas para desarrollo del microprocesador.

Tabla 18: Instrucciones de transferencia y de lectura/escritura de puertos (TIPO I).

INST.	COD. OP	REGISTRO FUENTE 1	REGISTRO FUENTE 2	REGISTRO DESTINO	SHAMT	FUNCT
		RS	RT	RD	SHAMT	FUNCT
MOV	50d	XXXXX	00000	XXXXX	00000	000000
MOVC	50d	00000	00000	XXXXX	00000	000001
RIO	50d	XXXXX	00000	XXXXX	00000	000010
WIO	50d	XXXXX	00000	XXXXX	00000	000011
	6	5	5	5	5	6
BITS	31-26	25-21	20-16	15-11	10-6	5-0

El código de instrucción para las instrucciones mostradas en la tabla 17, corresponde al valor b”110010”, que en decimal es d”50”, para diferenciar entre las operaciones MOV, MOVC, RIO y WIO utilizamos distintos valores para el campo función, el cual le indica la operación a realizar, el campo función tiene la combinación binaria b”000000” para la instrucción MOV que mueve el contenido de un registro a otro, b”000001” para MOVC que carga una constante de 32 bits desde ROM a un registro, b”000010” para RIO que copia el contenido de un puerto de entrada a un registro y b”000011” que copia el contenido de un registro a un puerto de salida. Como se observa en la tabla 17, tenemos una instrucciones que no trabaja con registros fuente, esta es MOVC para el resto utilizamos un registro fuente. En la figura 1 se muestra un diagrama a bloques de lectura, decodificación y ejecución de instrucciones de transferencia. En la figura 73 se muestra el diagrama de flujo de la lectura, decodificacion y ejecucion de instrucciones de trasferncia. Comenzando con la lectura de datos se obtiene la instrucción y se genera la direccion de la siguiente instrucción, se decodica el codigo de operación, al tratarse de una instrucción se procede a decodificar la funcion, y en base a este ultimo campo se ejecutan las instrcciones de la siguiente manera: La instrucción MOV, lee el contenido del registro fuente 1 (RS) y lo copia al registro distino (RD). La instrucción MOVC solicita la lectura de la siguiente localidad de memoria y genera la direccion de la siguiente instrucción, el dato leido de la ROM es almacenado en el registro destino. La instrucción RIO lee el puerto de entrada que se especifica con los 5 bits del registro fuente 1 y copia este valor al registro destino y por

ultimo para la instrucción WIO se toma el valor del registro fuente 1 y se imprimen en el puerto de salida que se especifica con los 5 bits del registro destino

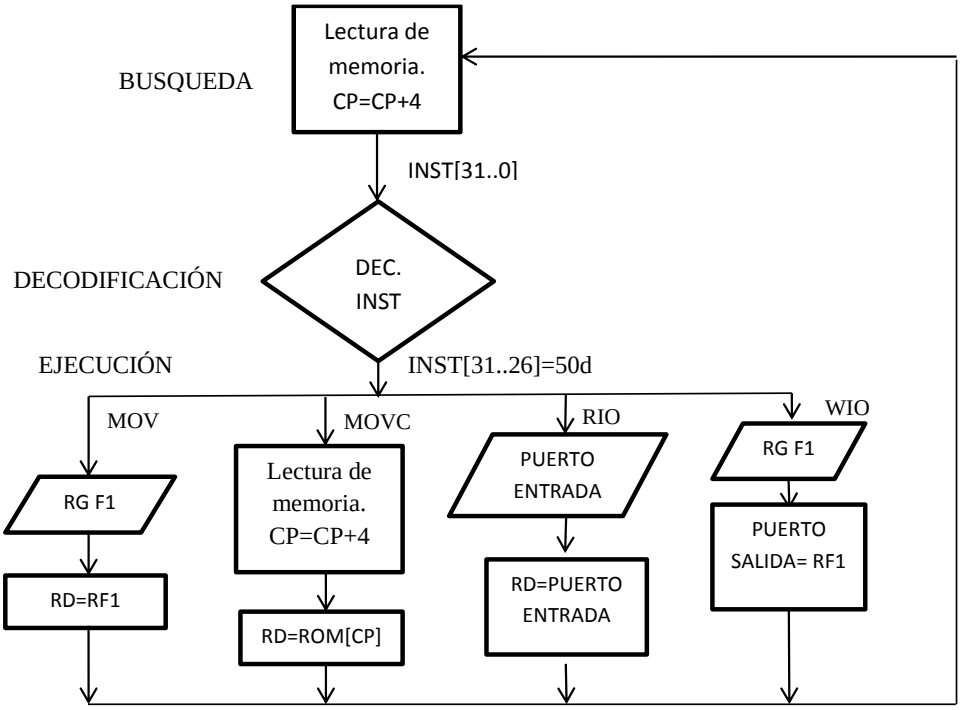


Figura 73: Diagrama a bloques de lectura, decodificación y ejecución de instrucciones de transferencia.

3.10.4 Instrucciones condicionales

En la tabla 19 se muestran estas instrucciones que cuentan con cuatro campos, el primero de 6 bits que corresponde al código de instrucción, el segundo de 5 bits que corresponde al primer registro fuente, el tercero con 5 bits que corresponden al registro fuente 2 y el cuarto campo con 16 bits para declarar el desplazamiento (en número de instrucciones) respecto al contador de programa.

Tabla 19: Instrucciones condicionales.

INST.	COD. OP	REGISTRO FUENTE 1	REGISTRO FUENTE 2	DESPLAZAMIENTO
		RS	RT	INMEDIATO
C.EQ.S	6	XXXXX	XXXXX	XXXXXXXXXXXXXXXX
C.LE.S	5	XXXXX	XXXXX	XXXXXXXXXXXXXXXX
C.LT.S	4	XXXXX	XXXXX	XXXXXXXXXXXXXXXX
	6	5	5	16
BITS	31-26	25-21	20-16	15-0

Como instrucciones condicionales tenemos a C.EQ.S que verifica si el registro fuente 1 es igual al registro fuente 2, de cumplirse la condición la siguiente instrucción correspondería a la dirección CP=CP+ (desplazamiento x 4), de no cumplirse, la siguiente instrucción a

ejecutarse correspondería a la dirección CP=CP+4. La instrucción C.LE.S verifica si el registro fuente 1 es menor al registro fuente dos, la instrucción C.LT.S verifica que el RF1sea menor igual a RF2, de cumplirse cualquier instrucción condicional la operación a realizar es la misma que ya se explicó para la instrucción C.EQ.S

Las instrucciones condicionales pueden saltar $2^{15} - 1$ instrucciones hacia adelante y 2^{15} hacia atrás.

Veamos un ejemplo de uso de estas instrucciones:

C.EQ.S R0, R1, 6205

Si R0: es igual 3f800000H y R1=3f800000H y CP=2000

Entonces la dirección d la instrucción destino saltara PC=PC+(6205*4)= 26820d

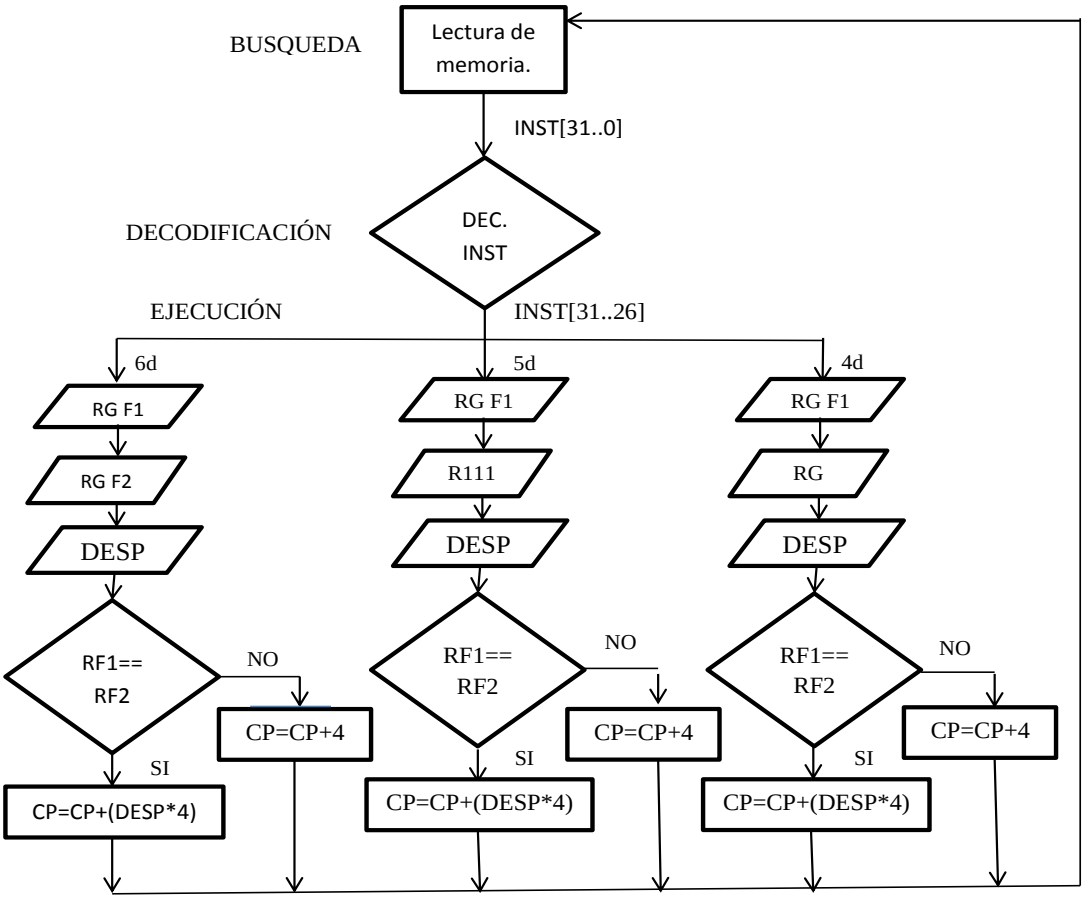


Figura 74: Diagrama a bloques de lectura, decodificación y ejecución de instrucciones condicionales.

Como se observa en la figura 74, el código de operación para estas instrucciones no es la misma, esto se debe a que no contamos con un campo de función. Como ya se había mostrado en el ejemplo anterior estas instrucciones condicionales comparan dos números en punto flotante y si la condición se cumple la siguiente instrucción corresponderá al de la

dirección CP=CP+(DESP*4) en caso de que la señal no se cumpla se ejecutara las siguiente instrucción con CP=CP+4.

3.10.5 Instrucción de salto

Esta instrucción cuenta con dos campos, el código de instrucción y el campo de dirección destino. Como se observa en la tabla 20 se muestra el formato de la instrucción de salto JUMP. Donde la X denota bits que podemos manejar y los bits marcados con un 0 corresponden a valores no utilizados.

Tabla 20: Instrucciones condicionales.

INST.	COD. OP	Dirección destino
		Dirección
JUMP	2d	00XXXXXXXXXXXXXXXXXX
	6	26
	31-26	25-0

El proceso de ejecución de esta instrucción toma la dirección especificada en la instrucción y lo carga al valor del contador de programa CP, quedando CP=Dirección, con esto la siguiente instrucción a ejecutar será la que se encuentre en la dirección especificada en la instrucción como se muestra en la figura 75.

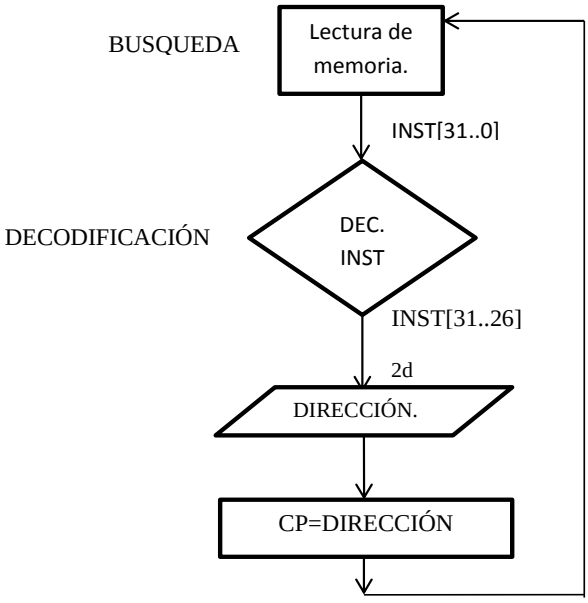


Figura 75: Diagrama a bloques de lectura, decodificación y ejecución de instrucciones de salto.

En las tablas 17, 18, 19 y 20 se observan los campos que componen cada una de las instrucciones, y se indica en números decimales constantes el código de instrucción y en números binarios la función y los demás campos. Para los campos o bits cuyo valor no es considerado por el microprocesador se usa el cero. Con X se marcan los campos y los bits

que el decodificador de instrucciones lee e interpreta. A continuación se describe a detalle cada una de las instrucciones, comenzando con las instrucciones aritméticas.

3.11 El compilador

En la figura 76 se muestra el compilador que se encarga de traducir del lenguaje ensamblador a hexadecimal [25].

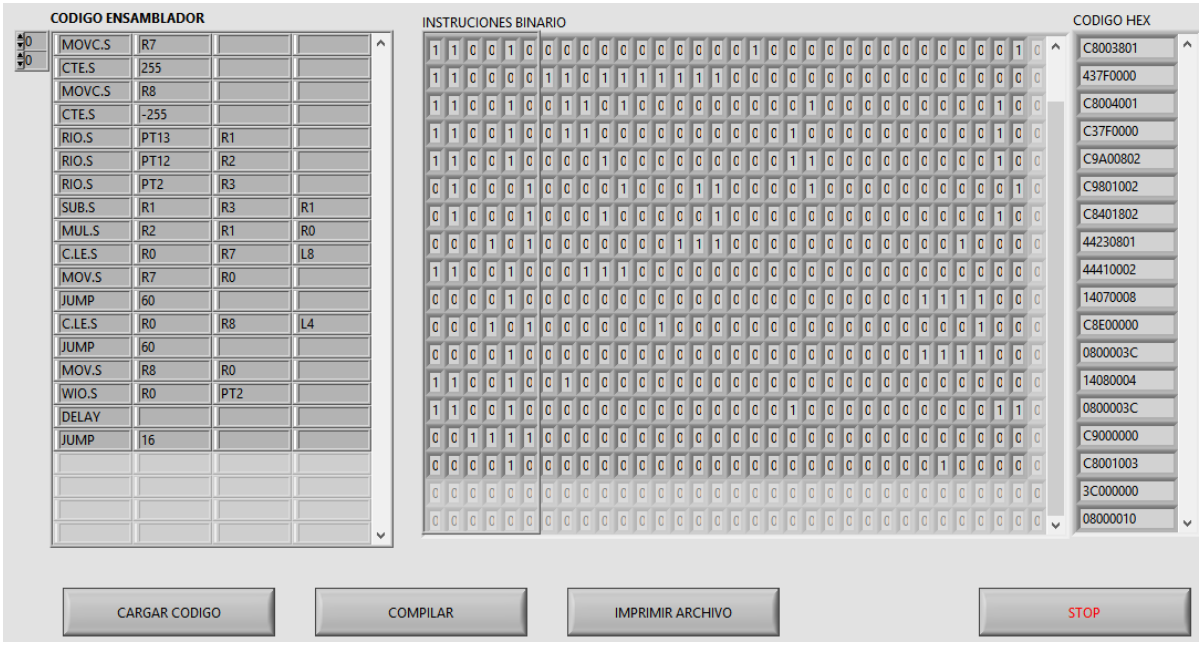


Figura 76: Compilador.

3.12 Software

Para manipular el sistema embebido desde un equipo de cómputo vía WiFi se realizó una interfaz de usuario mediante la plataforma LabView. En la figura 77 se observa la interfaz gráfica que permite la programación, envió de parámetros y lectura de datos de memoria RAM, para la visualización y manipulación de resultados generados por la ejecución de un programa. A continuación se explican cada una de las partes de la interfaz de usuario.

1. Botón de Inicio de comunicación wifi y paro de emergencia.
2. Borrado de memoria ROM.
3. Escritura de memoria ROM con algoritmo compilado en hexadecimal.
4. Parámetro 1.
5. Parámetro 2.
6. Parámetro 3.
7. Botones de Inicio y paro de ejecución de algoritmo de control e indicador.

Para ejecutar el algoritmo de control es necesario realizar los siguientes pasos:

1. Iniciar comunicación con modulo wifi botón “inicio/paro”
2. Tener el archivo precompilado del algoritmo de control haciendo uso del compilador.
3. Presionar botón 3 “borrar memoria”.
4. Presionar indicador “archivo completo” seguido de “escritura secuencial”.
5. Cargar parámetros.
6. Ejecutar el algoritmo mediante el botón “start”

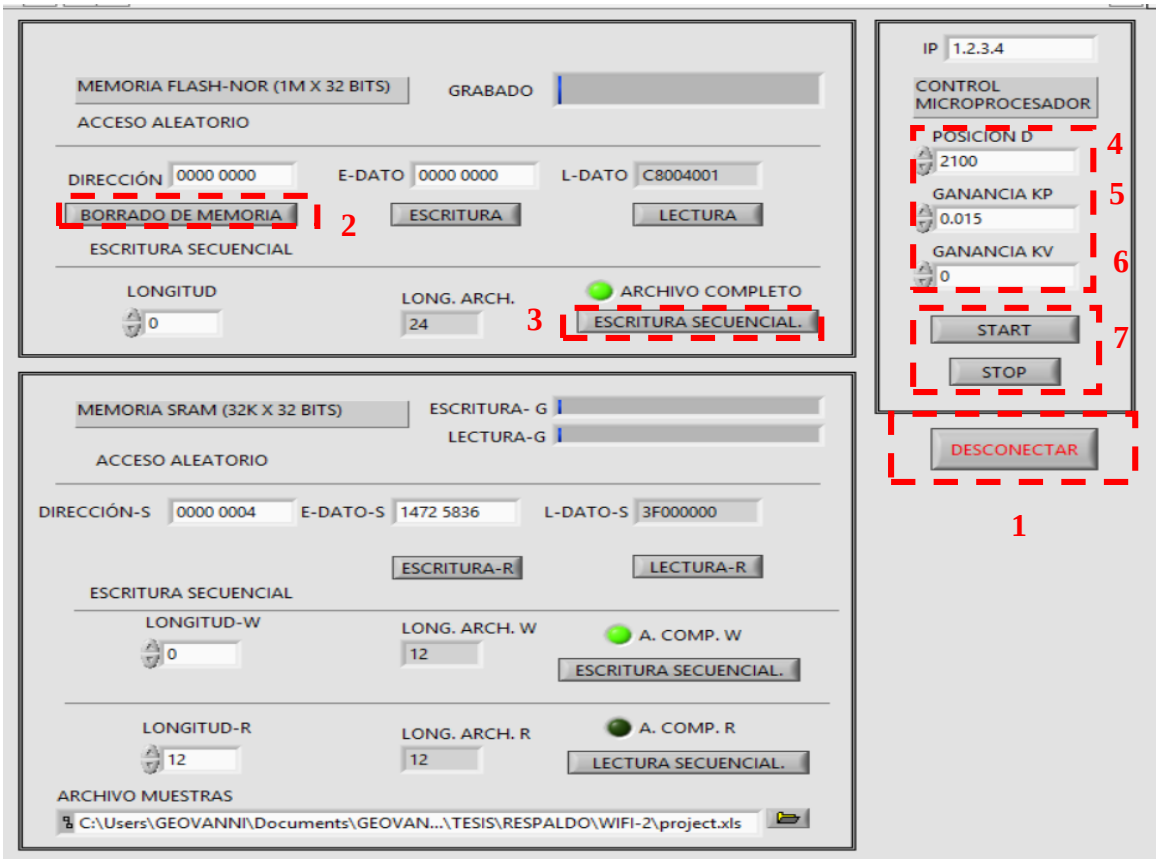


Figura 77: interfaz de usuario para la programación, control y monitoreo del sistema embebido.

3.13 Conclusiones

El uso de un microprocesador embebido en FPGA permite resolver diversidad de problemas en forma secuencial suministrando versatilidad en la integración de sistemas digitales complejos, economizando recursos de hardware, esfuerzos de ingeniería y tiempo de desarrollo.

Capítulo 4. Pruebas y resultados

Para verificar el funcionamiento del microprocesador, a continuación se expone tres aplicaciones de control en los que se utilizó el sistema desarrollado en este trabajo de tesis (tarjeta electrónica, microprocesador embebido y software de interfaz), la aplicaciones son el control de una mano robótica de 5 grados de libertad, el control de un sistema mecatrónico con 3 grados de libertad que emula los movimientos del cuello humano y este se implementó en el manejo de una mano robótica y el control de un robot CNC de 3 grados de libertad.

4.1 Aplicación de sistema embebido en el control de una mano robótica

La comunicación y adquisición de datos de la mano de cinco grados de libertad se realiza mediante una interfaz gráfica en LabView por medio de wifi hacia un módulo Wifly rn-xv y una tarjeta con FPGA la cual tiene implementada el microprocesador desarrollado en este trabajo. El microprocesador ejecuta su propio conjunto de instrucciones desde la ROM con su previa compilación y grabado, se procesa el algoritmo de control, una vez realizado el control se envían las señales correspondientes a las etapas de potencia que realizan el movimiento de los actuadores. Los actuadores tienen un encoder que sirven de retroalimentación mediante una etapa de acondicionamiento hacia los decodificadores de posición donde se obtiene la posición actual y el resultado entra en los puertos de entrada recalculando el algoritmo de control proporcional en el cual se obtiene el error de posición y se manda hacia los puertos de salida donde entran a una etapa de PWM y salen las señales hacia las etapas de potencia para cerrar los cinco lazos de control para el control de la mano como se muestra en la figura 78 [26].

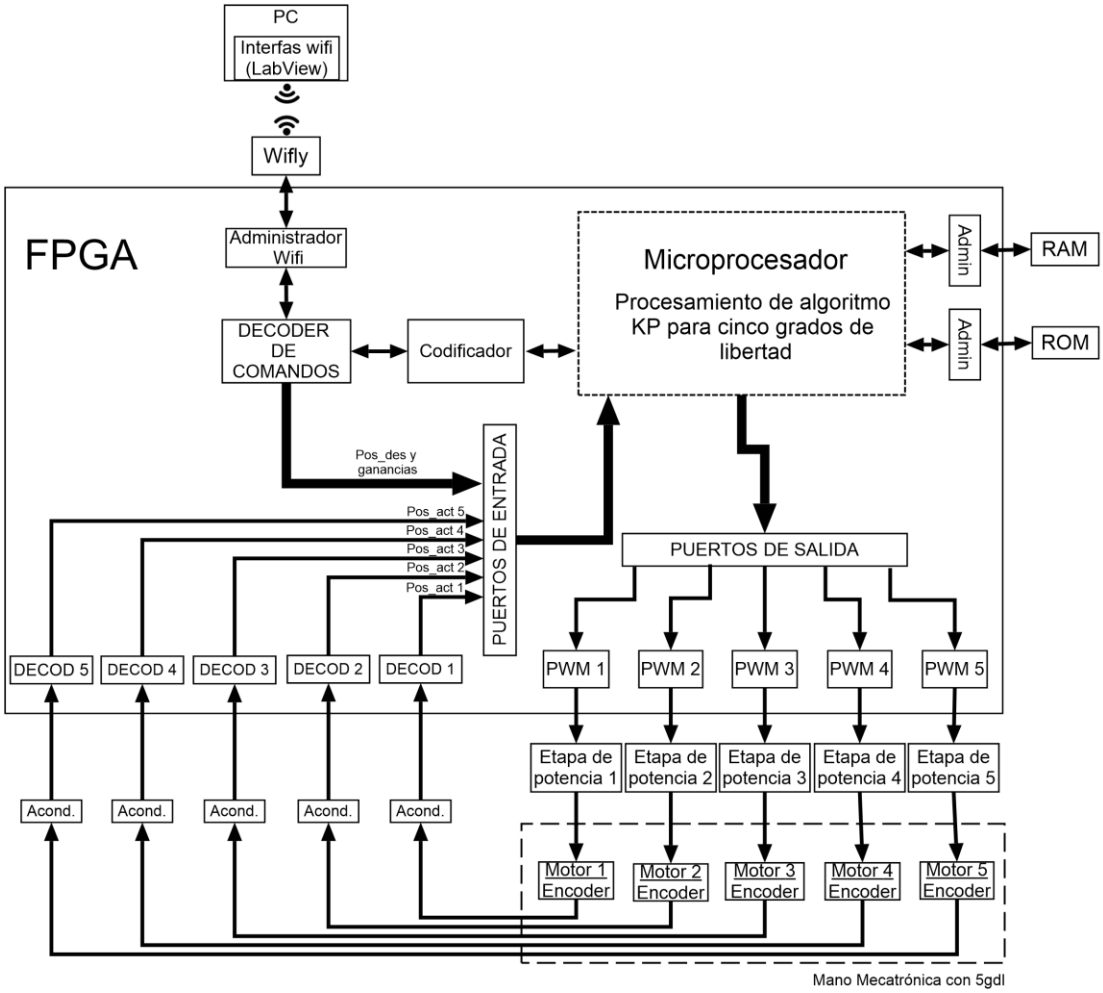


Figura 78: Firmware de microprocesador implementado y modificado para 5gdl.

En la figura 79 se muestra la conexión de cinco placas o módulos los cuales cuentan cada uno con su respectiva etapa de potencia protección y acondicionamiento de señales para el control del sistema de la mano macarrónica, las cinco placas se conectan a una tarjeta de expansión y la tarjeta se conecta mediante un cable plano hacia el FPGA que contiene las señales de Encoders, señales PWM así como también consta de un conector para las alimentaciones a las etapas de potencia para los motores.

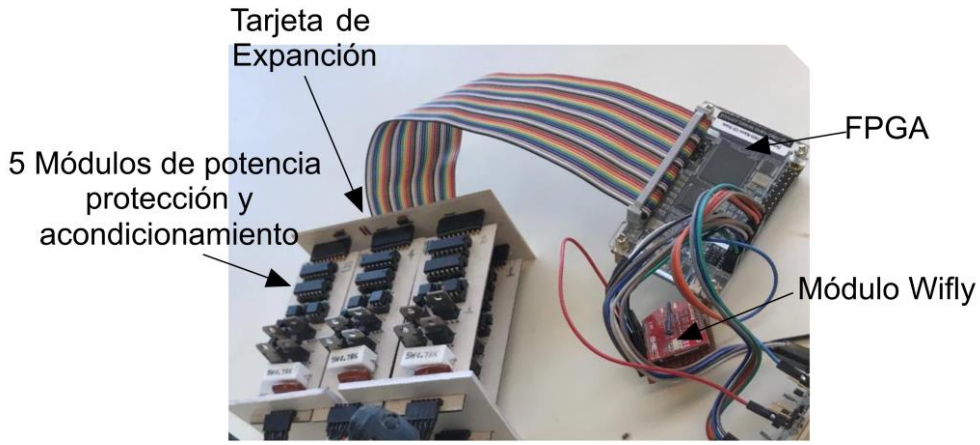


Figura 79: Sistema electrónico implementado

En la figura 80 se muestra el sistema completo de la mano macarrónica la cual tiene 5 actuadores para el movimiento en flexión correspondientes a los cinco dedos de la mano y los actuadores se conectan a su respectiva etapa de potencia, protección y acondicionamiento hacia el FPGA por medio de un cable plano permitiendo realizar la conexión entre las señales que nos interesan para el control mediante comunicación wifi por un módulo wifly rn-xv.

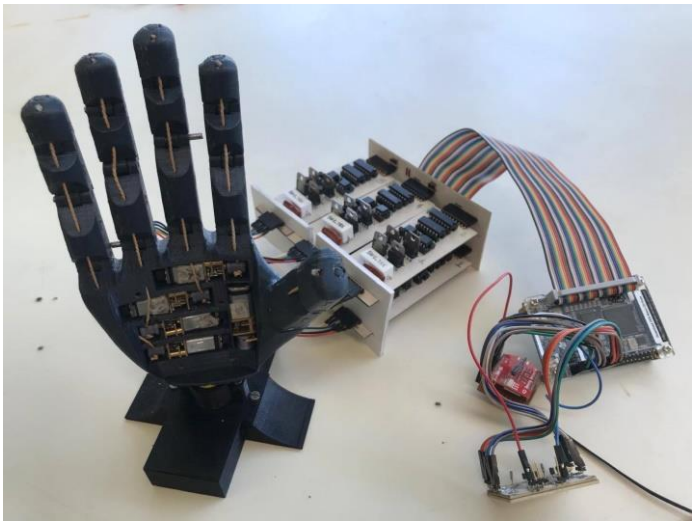


Figura 80: Sistema completo de la mano mecatrónica.

Mediante una interfaz en LabView fue posible enviar las posiciones deseadas a cada uno de los actuadores para la flexión de los dedos, (ver Fig 81).

Figura 81: Interfaz gráfica realizada en LabView

En la figura 82 se muestra el resultado del analizador lógico herramienta de Quartus.

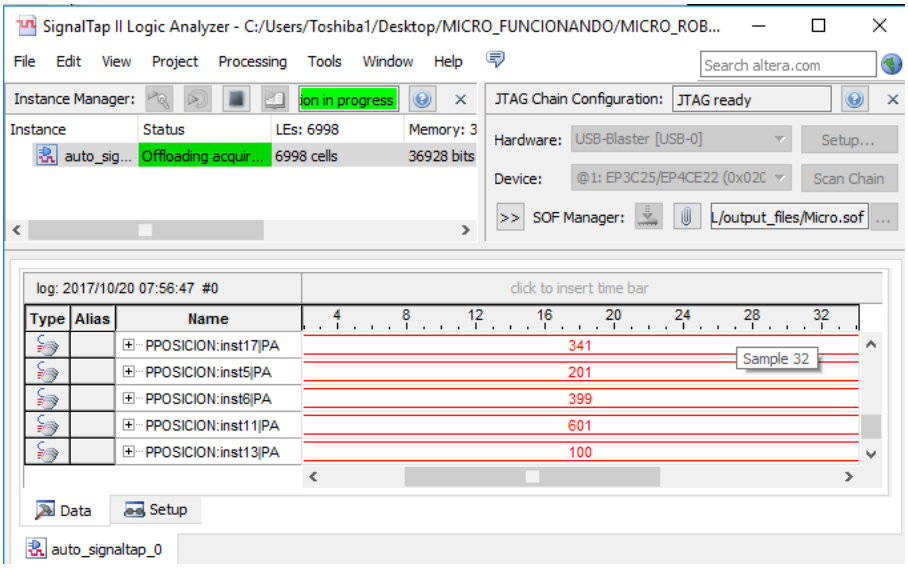


Figura 82: Posiciones deseadas alcanzadas +/- 1cuenta.de error

Las gráficas correspondientes al error de posición para una posición deseada a 180 grados para el motor 1 (pulgar), el torque calculado y la velocidad estimada se muestran en la figura 83.

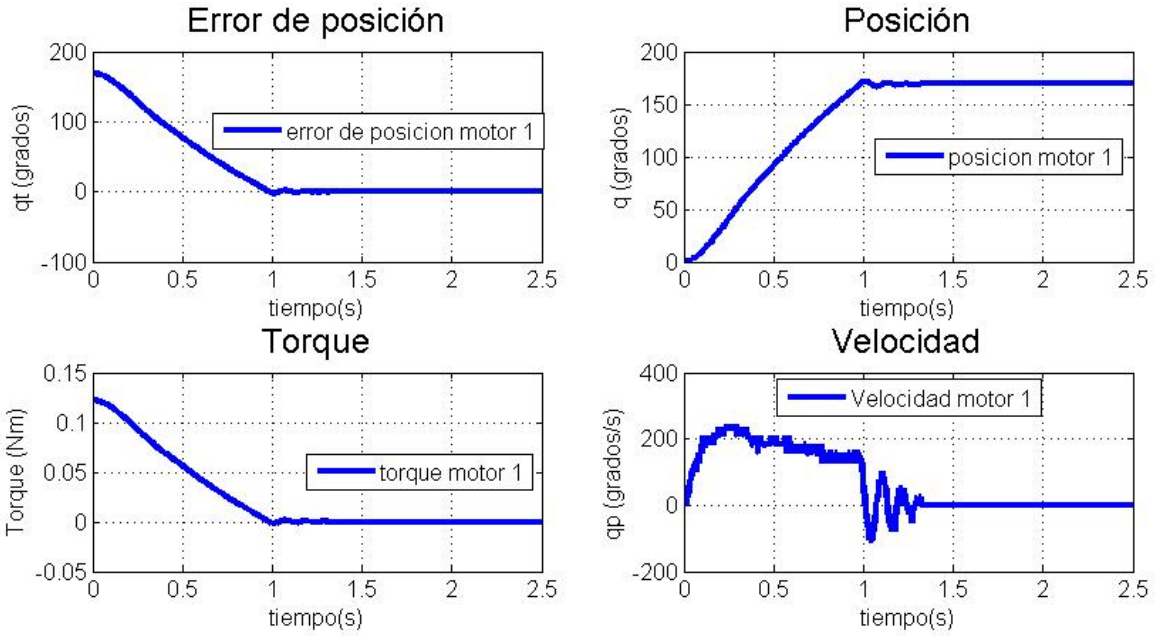


Figura 83: Gráficas para el motor 1 (pulgar).

Las gráficas correspondientes al error de posición para una posición deseada a -210 grados para el motor 2 (Anular), el torque calculado y la velocidad estimada se muestran en la figura 84.

.

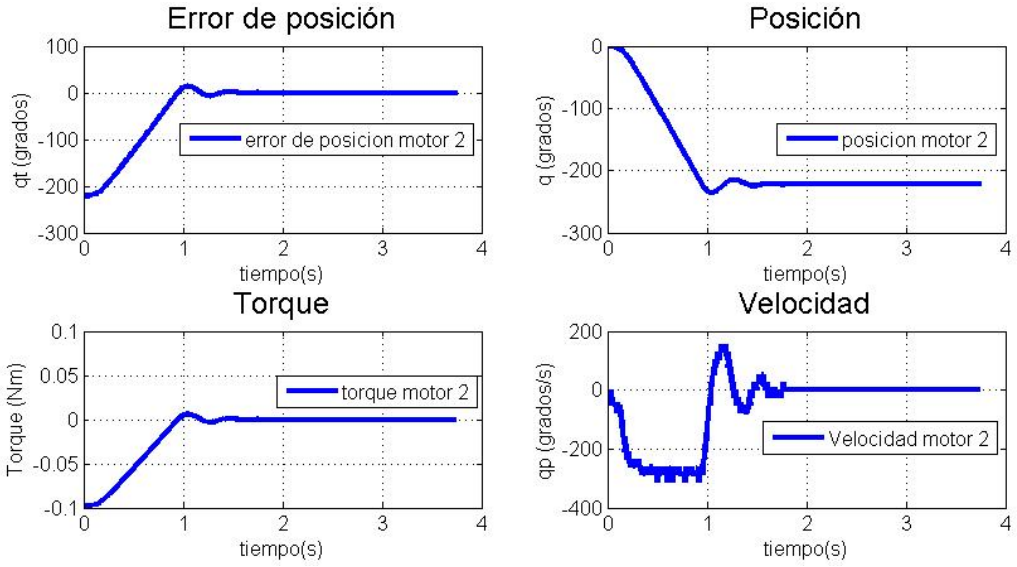


Figura 84: Gráficas para el motor 2 (Anular).

En la figura 85 se muestran las gráficas de error de posición, posición, torque, y velocidad para una posición deseada de 280° para el motor 3 (índice), el Torque aplicado mediante el algoritmo de control proporcional realizado de esta gráfica se puede deducir que a medida el error es muy grande el torque o par es máximo y a medida que error tiende a cero el torque se mantiene al mínimo sin llegar a ser cero para mantener la posición deseada.

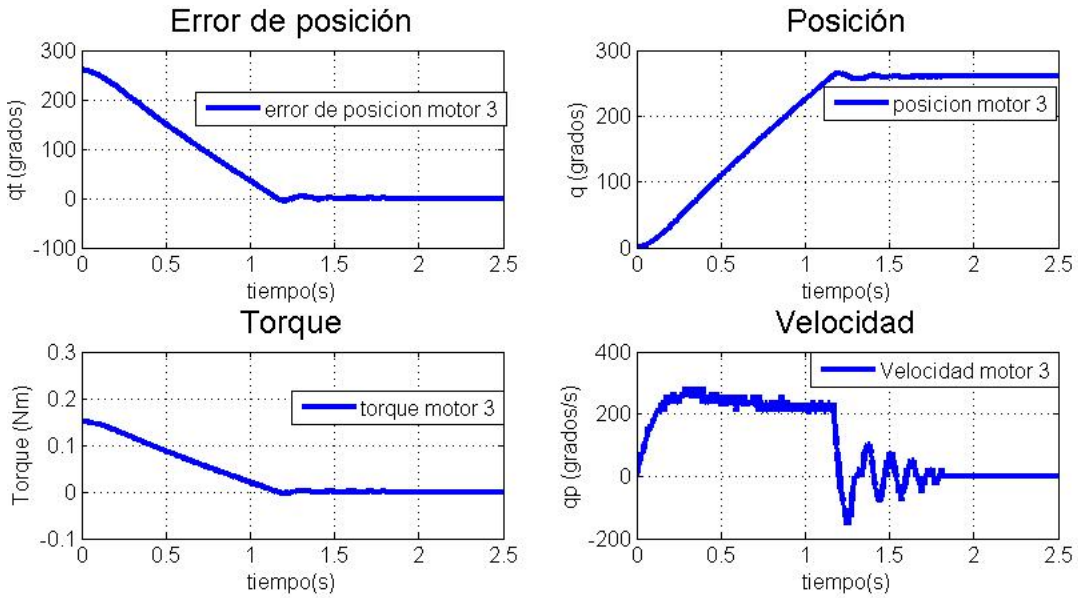


Figura 85: Gráficas de motor 3 (Índice).

En la Figura 86 se muestran las gráficas de error de posición, posición, torque y velocidad estimada para un posición deseada de 50°.

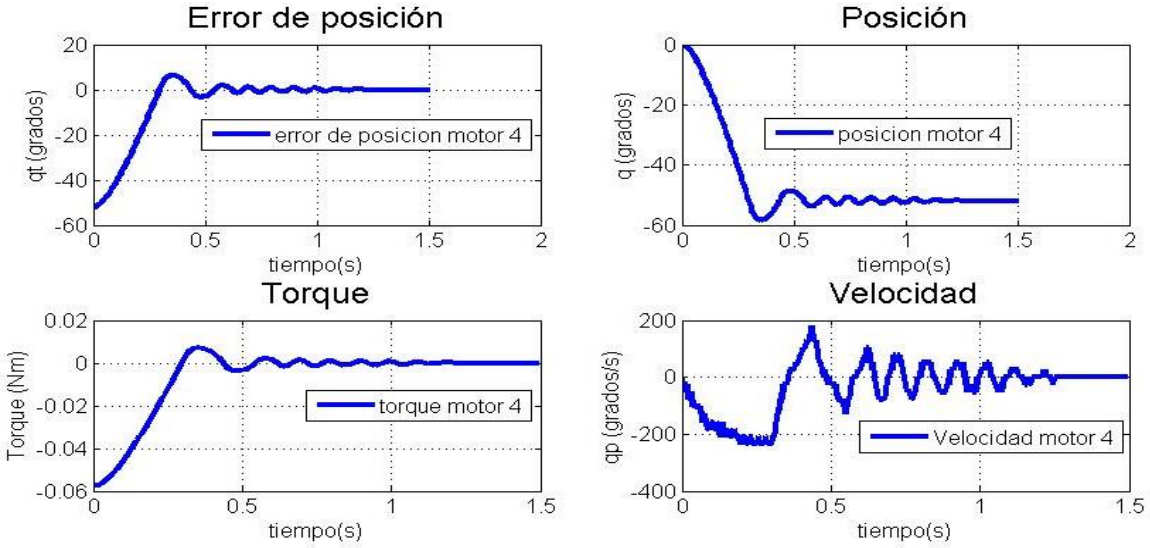


Figura 86: Gráficas de motor 4 (Meñique).

En la Figura 87 se muestran las gráficas de error de posición, posición, torque y velocidad estimada para una posición deseada de 270°.

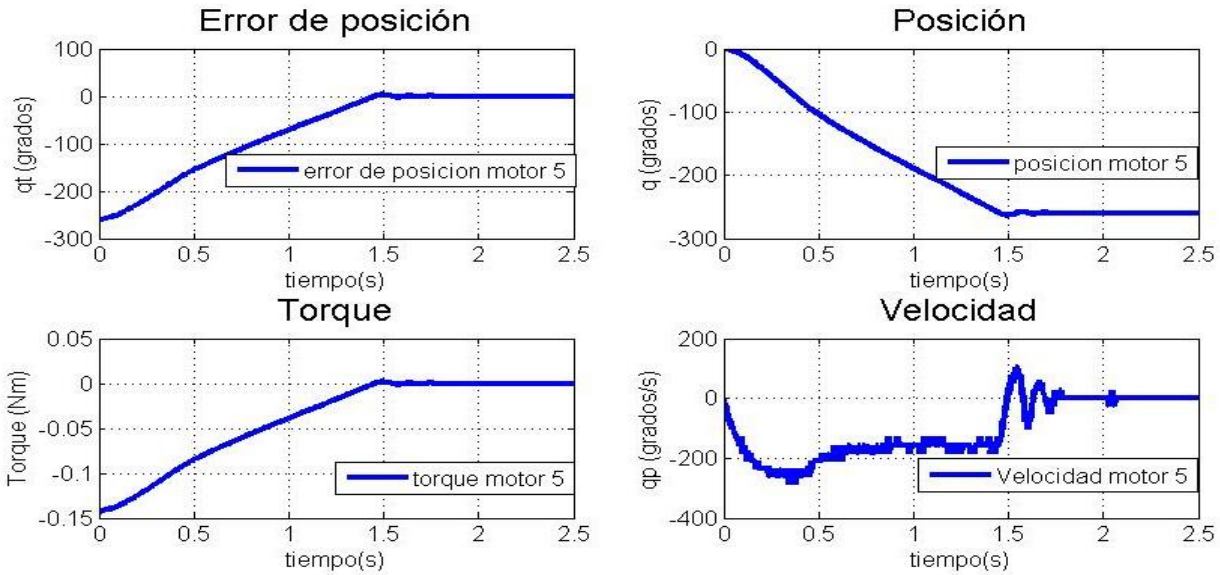


Figura 87: Gráficas de motor (Medio).

4.2 Aplicación del sistema embebido en el control de un robot que emula los movimientos del cuello humano

La etapa de control consta del microprocesador desarrollado en este trabajo microprocesador, en el cual se realiza la programación en lenguaje ensamblador de la acción de control, el código de programa se almacena en una memoria ROM y también se cuenta con una memoria RAM para almacenar variables y datos calculados por el mismo, además cuenta con periféricos de entrada, que son decodificadores para la adquisición de datos de los codificadores de cuadratura, y registros donde se almacenan los datos enviados desde la interfaz gráfica de la PC, tanto la programación como la comunicación de datos es realizada mediante un módulo externo Wi-Fi, los periféricos de salida son generadores de señal PWM, lo anterior fue implementado en un FPGA Cyclone 4 y se realizó el firmware en lenguaje AHDL. Se diseñó un puente H para la etapa de potencia de cada motor, ya que las señales de control solo pueden entregar hasta 1mA a 3.3v, por este motivo se requiere un circuito con transistores BJT en configuración Darlington. En la figura 88 se muestran los módulos que conforman el firmware del sistema [25].

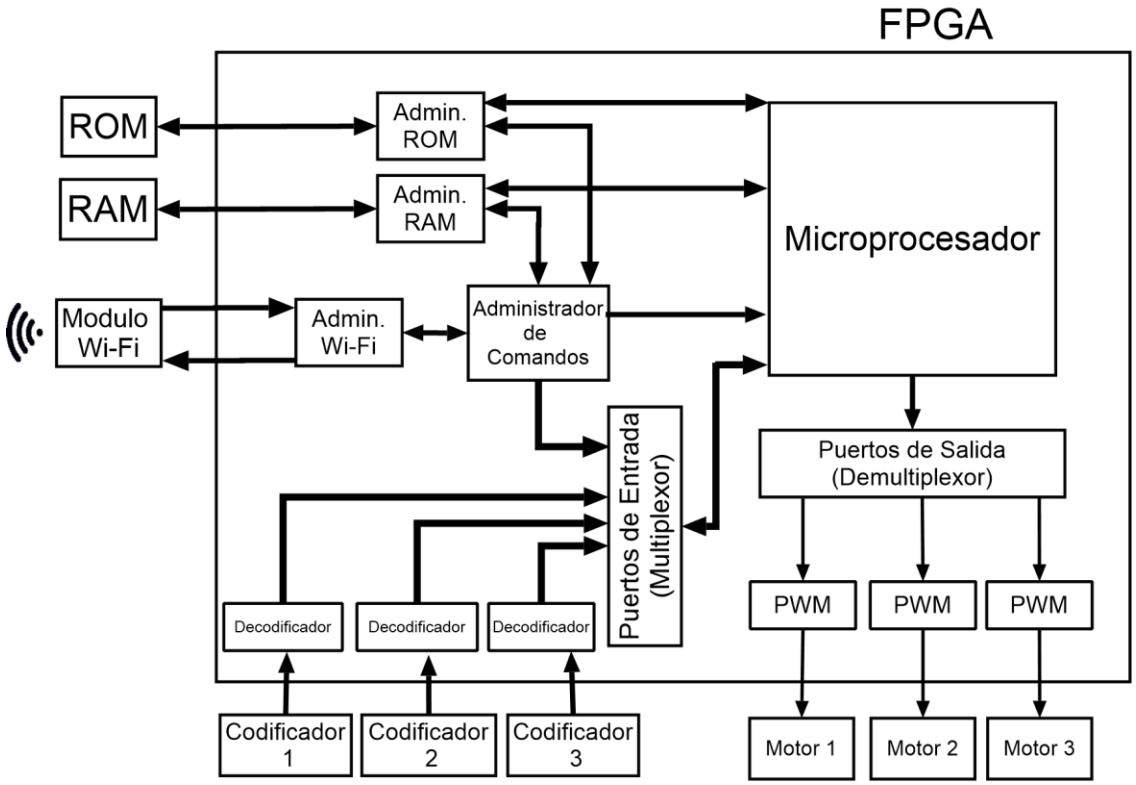


Figura 88: Diagrama general del firmware

En la figura 89 se observa la integración del sistema mecatrónico.

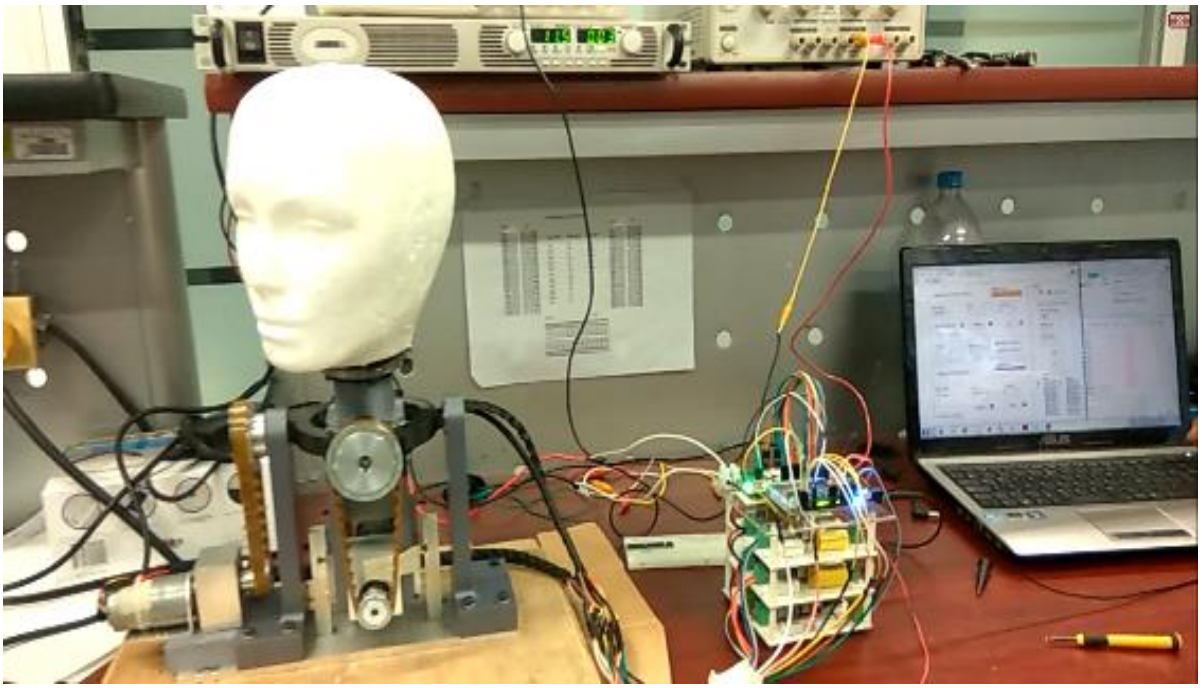


Figura 89: Sistema mecatrónico que emula el movimiento del cuello de los seres humanos.

El software utilizado para el control del sistema mecatrónico se implementó en LabVIEW, lo cual permite que el usuario tenga una interfaz visual amigable y logre controlar el sistema mecatrónico sin ningún problema, se muestra en la figura 90.

La interfaz principal de control consta de una ventana donde se encuentra una sección para programar el sistema embebido y otra para iniciar la acción de control. La pestaña que contiene las herramientas para programar el sistema embebido puede cargar un archivo de texto con las instrucciones para controlar el sistema en formato hexadecimal, las cuales se programan directamente en la memoria ROM. Así mismo, la interfaz contiene una pestaña de control, la cual contiene nueve casillas para ingresar datos, esto es de ayuda para el usuario ya que puede controlar el sistema en tiempo real y no es necesario programar el microprocesador, ya que estas casillas fueron diseñadas como puertos de entrada.

Para programar el microprocesador se necesita el archivo que contiene las instrucciones en formato hexadecimal, una vez obtenido se procede a iniciar el software con el botón “inicio” con lo que se encenderá el indicador “conectado” cuando la comunicación Wi-Fi sea establecida con éxito, el siguiente paso es borrar la memoria ROM con el botón “borrar memoria” para que no se generen errores como el tener una instrucción de algún firmware que se haya programado con anterioridad, se comprueba que la memoria este vacía con el botón “leer” y en la casilla lectura deberá mostrar el dato en hexadecimal “FFFFFFFF”, por lo tanto presionamos el indicador “archivo completo” y con el botón “escritura secuencial” se despliega una ventana para que seleccionemos el archivo hexadecimal, la barra indicadora “grabado ROM” representa el porcentaje de avance de grabado. Por último, se

proporcionan los parámetros de posiciones, ganancias para iniciar el proceso de control con el botón “Start” y para terminar el proceso se presiona el botón “Stop”.

Figura 90: Interfaz visual que controla el sistema mecatrónico.

Se realizaron pruebas con el sistema mecatrónico diseñado con el firmware presentado en la figura 88, donde se implementó un controlador proporcional con par gravitacional, cabe mencionar que el modelo dinámico obtenido en este trabajo nos proporciona el elemento de par gravitacional. Una vez programado el microprocesador se realizaron pruebas y posteriormente se adquirieron los datos guardados en la memoria RAM, cabe mencionar que los datos de velocidad se calcularon posteriormente a las pruebas, ya que no se cuenta con un sensor de velocidad, por tanto, se conoce la posición y se deriva aplicando el método sencillo de Euler.

En la figura 91 se presenta la respuesta del sistema al realizar control de posición en el eslabón de flexo-extensión, la posición deseada es de 10° y se utilizó una ganancia de 0.3, se logra observar que la respuesta del torque tiene la misma trayectoria que el error de posición, esto indica que la energía entregada por los actuadores se comporta de manera. Además se observa un error en estado estacionario de 0.8°, esto es característico de un controlador proporcional ya que se debe sintonizar las ganancias y es un método que se realiza de manera empírica.

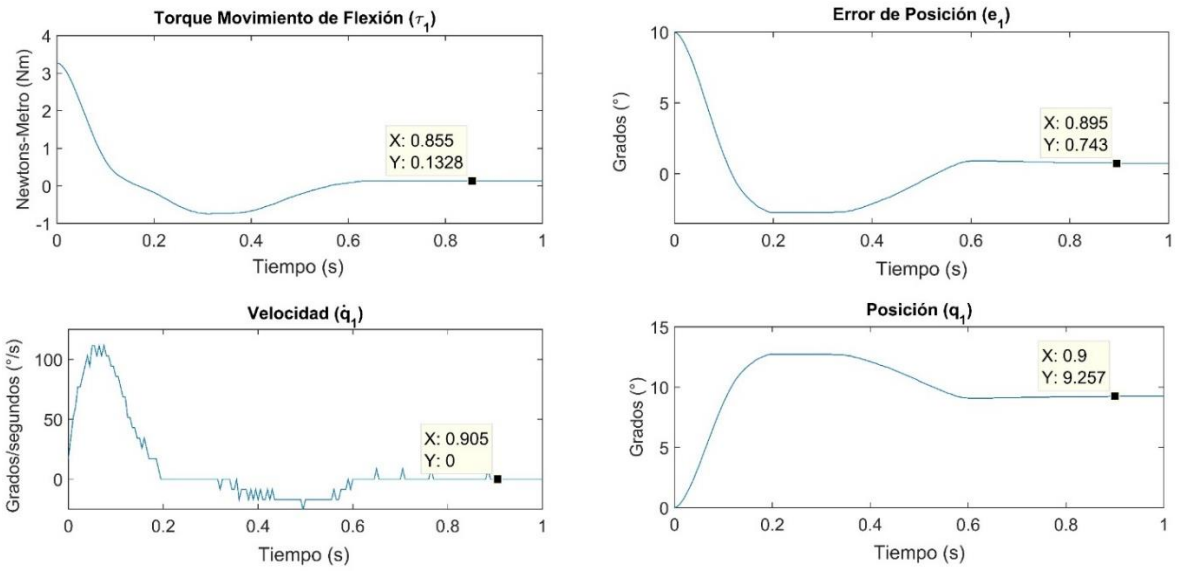


Figura 91: Datos de la acción de control del sistema para el movimiento de flexión.

En la figura 92 se presenta la respuesta del sistema al realizar control de posición en el movimiento de inclinación bilateral, la posición deseada es de 10° y se utilizó una ganancia de 0.3, se puede apreciar que el comportamiento de este movimiento del sistema es oscilante, debido a las bandas de transmisión de potencia que actúan de esta manera, se observa un error en estado estacionario de 1.1°, aunque para esta aplicación la magnitud del error puede ser despreciable ya que no es un sistema de precisión

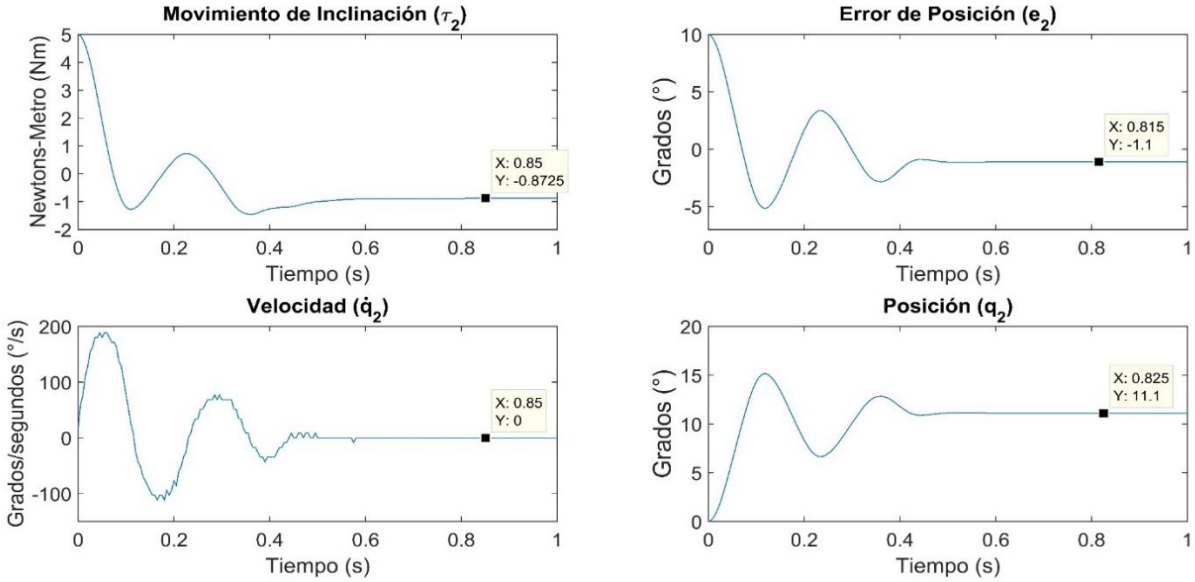


Figura 92: Datos de la acción de control del sistema para el movimiento de flexión.

En la figura 93 se presenta la respuesta del sistema al realizar control de posición en el movimiento de rotación, la posición deseada es de 35° y se utilizó una ganancia de 0.07, se puede observar que en este movimiento el sistema responde efectivamente sin ningún tipo de perturbación ya que es el movimiento que menos dinámica representa, cabe mencionar que el torque entregado por el actuador se comporta de forma lineal, el error en estado estacionario es de 1.4° lo que representa que la ganancia ocupada en este movimiento es poca y se deben sintonizar para mejores efectos de control.

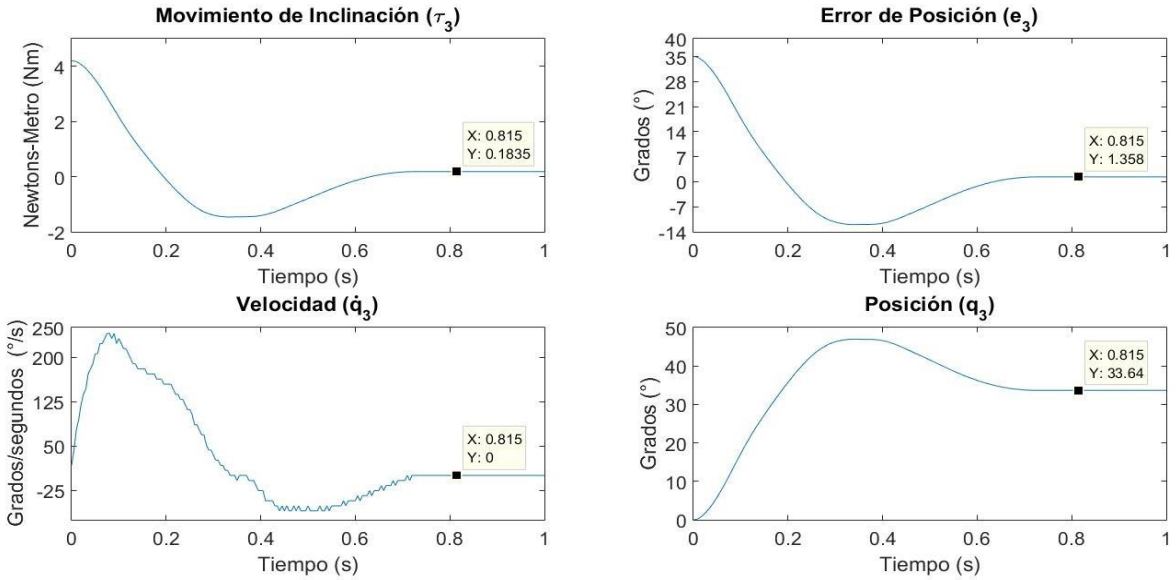


Figura 93: Datos de la acción de control del sistema para el movimiento de rotación.

4.3 Aplicación del sistema embebido para el control de un robot cartesiano de 3gds de libertad

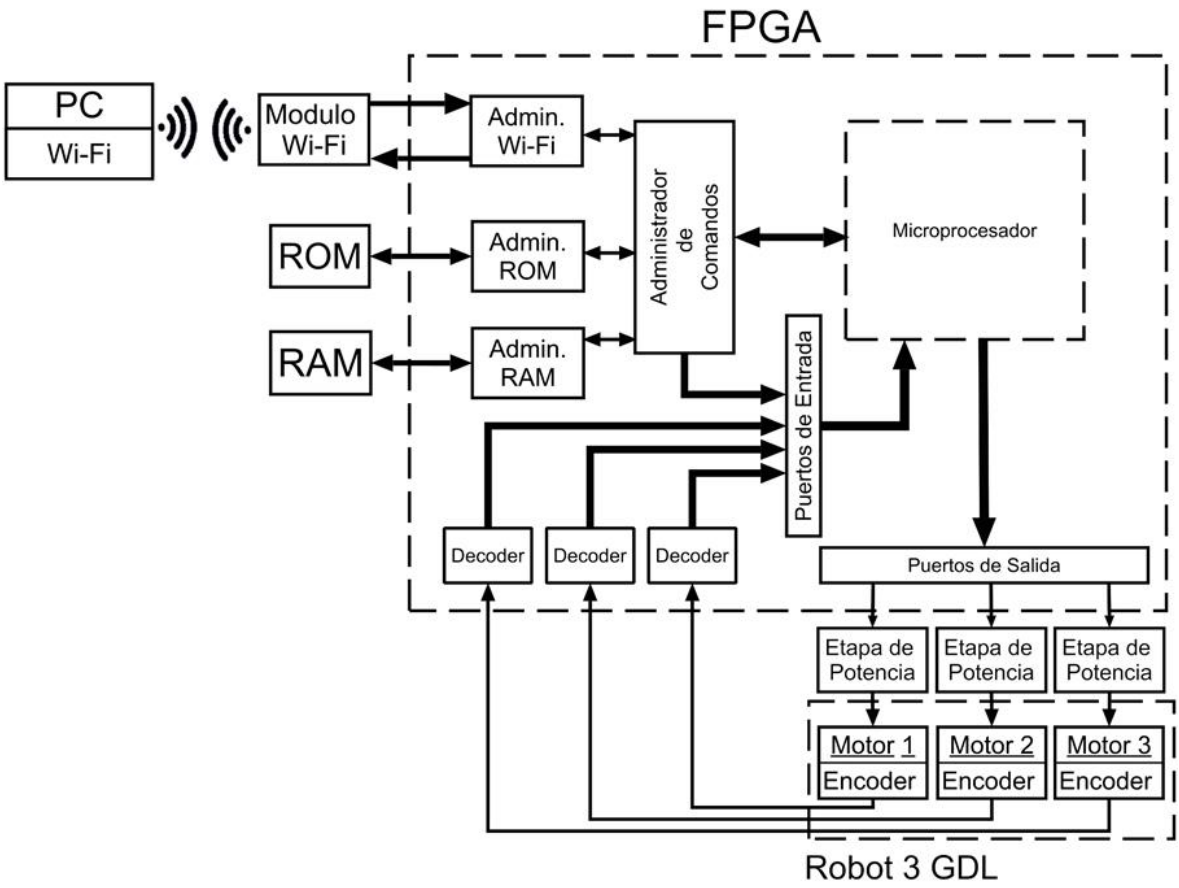


Figura 94: Diagrama a bloques general del sistema de control del robot

Del diagrama anterior, el elemento que se monitorea es la articulación del robot, se obtienen los valores correspondientes a la posición actual y par aplicado, en base al tiempo y a la posición deseada, se calcula el error de posición a lo largo del funcionamiento del sistema. Estos datos son leídos a través de la FPGA y son enviados a un módulo WIFI. El módulo WIFI es el RN-XV, en donde este dispositivo realizara la tarea de conversión de WIFI a serial y de serial a WIFI. Esta comunicación es bastante atractiva para el usuario ya que se hace compatible con computadoras de escritorio o computadoras portátiles de manera inalámbrica.

En el dispositivo se implementaron módulos de lógica programable que conforman las etapas de administración de comunicación Wi-Fi, administración de almacenamiento de información, administración de comandos, decoders, módulo de control y finalmente puertos de entrada y salida. Se pretende que la unidad de procesamiento digital reciba información para el control del sistema mecatrónico a través del módulo de comunicación inalámbrico, para lo cual el administrador de comunicación organizara la información para

que el administrador de comandos defina qué acción realizar ya sea el caso de guardar variables en la memoria ROM a través de la etapa de administración de ROM, enviar los datos procesados por medio del acceso a la memoria RAM a través de su módulo de administración, o iniciar el proceso de control.

Para iniciar el proceso de control del mecanismo se debe contar con los parámetros de posiciones deseadas para los actuadores y se deberá monitorear la posición de estos por medio de los módulos decoders encargados de transformar las señales recibidas por los sensores del sistema para lograr implementar la ecuación de control, todo esto se realizará en un módulo de control. Para transmitir la respuesta hacia los actuadores se diseñará la etapa de potencia acorde a los requerimientos de los actuadores.

La ventaja de este sistema, es que, el algoritmo de control se encuentra dentro de la FPGA mediante firmware, reduciendo el tiempo de ejecución del algoritmo de control tomando así datos en tiempo real, puesto que es un hardware dedicado a realizar esa tarea específica.

Haciendo mención [27], en la cual se hace uso de su robot cartesiano, para lo cual la etapa de potencia y lecturas de señales del encoder, no se contaba con ello y se realizó la fabricación de estas tarjetas para poder usar el robot cartesiano ya hecho en la MCEA.

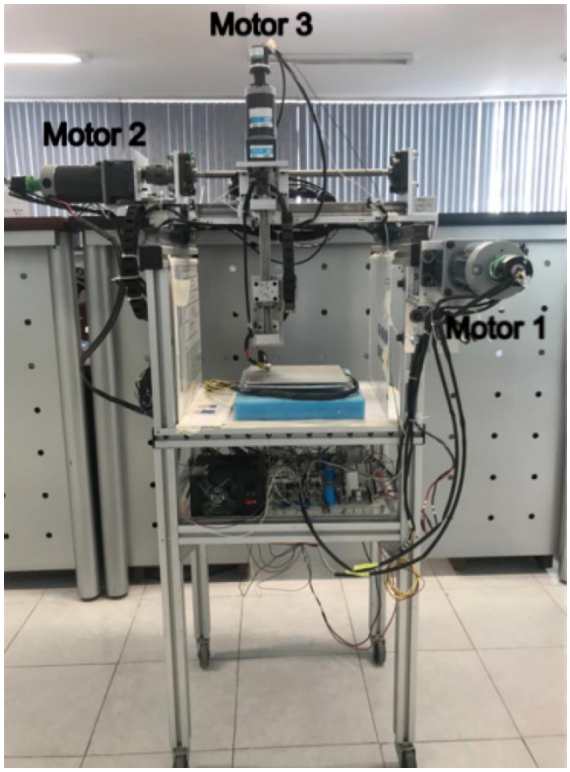


Figura 95: Robot CNC

Hoy en día, las aplicaciones son cada vez más complejas, por lo que el software especializado cada vez se presenta en forma más amigable al usuario, quien puede tener un cierto nivel de abstracción hacia esta herramienta, de aquí que, en la actualidad, existan diferente software de control, utilizados en la industria o en la educación y cada software presenta diferentes ventajas respecto a los demás. Basados en la ventaja del software especializado en la adquisición y control, para este trabajo el software utilizado es LabView de National Instruments, Labview es mucho más amigable, usa iconos para representar subprogramas, y se unen estos iconos para definir el flujo de los datos, permitiendo la adquisición de señales, análisis, escritura, además de ejecutar algoritmos matemáticos. De esta manera al utilizar la interface gráfica de Labview, e ir trazando el flujo de los datos utilizando iconos, el usuario realiza su código con lo cual se pueden escribir programas en menos tiempo que si se realizaran en forma de texto.

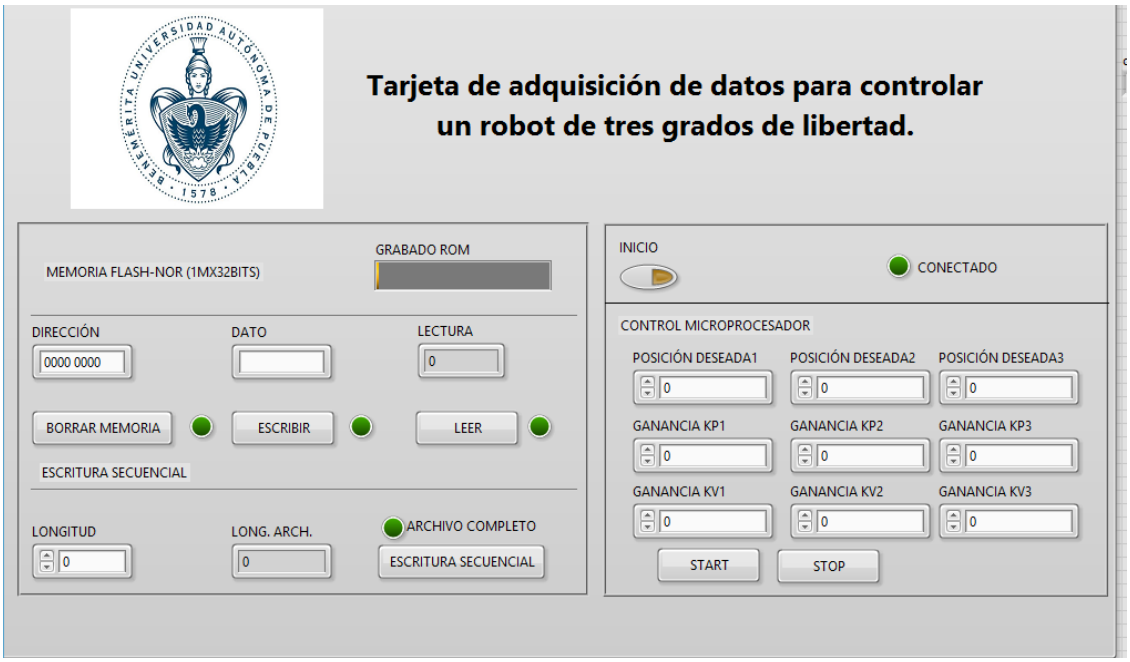


Figura 96: Pantalla principal del software.

En esta pantalla nosotros podemos visualizar las posiciones deseadas, para cada eje, además de poder ejecutar y detener del software, se puede borrar y leer la memoria ROM, así como grabarle a esta el firmware desarrollado para el control del robot de tres grados de libertad, el cual se mencionará más adelante, así también se puede modificar los parámetros de las ganancias que desee usar el usuario.

Para verificar el correcto funcionamiento del funcionamiento del robot de tres grados de libertad tipo cartesiano, se monitoreo la posición, error de posición, así como el torque y se compararon con los resultados obtenidos con el sistema comercial (sección I). En esta prueba al sistema se controlan los tres grados de libertad del robot, llevando cada grado de libertad a una posición deseada fija; las posiciones deseadas a la cual se realiza la prueba se puede ver en la tabla 21, para los tres grados de libertad.

Tabla 21: Posiciones deseadas para cada motor.

Grado de libertad.	Posición Deseada. (en grados)
Motor 1	90
Motor 2	90
Motor 3	90

En la figura 97, se muestran las gráficas de evolución de la posición para ambos sistemas de control durante el experimento. Observando una posición final de 89. 98958 º para el sistema con la tarjeta de adquisición de datos y control realizada en la tesis [7] y una posición final de 89. 2312 º para el sistema con la tarjeta PCI [6], la diferencia entre ambas posiciones finales es de 0.72838 º.

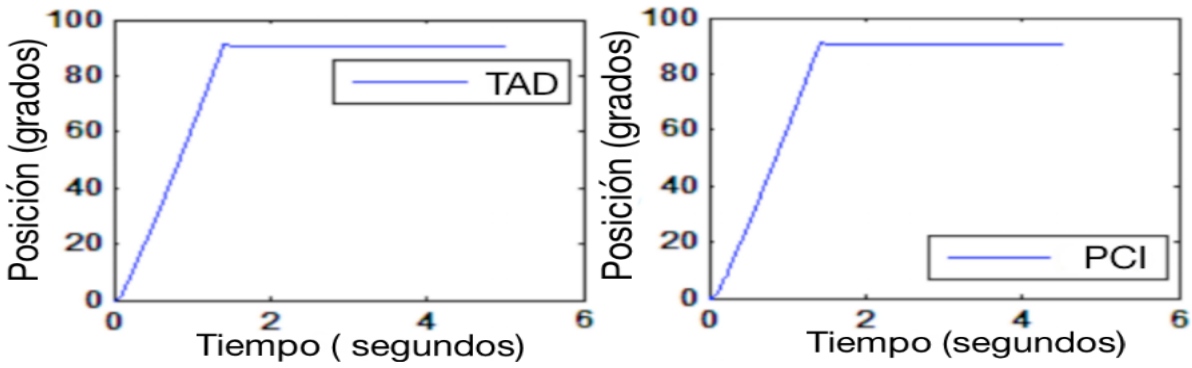


Figura 97: Gráfica de posición a 90º del motor 1.

El error de posición durante el experimento en ambos sistemas puede apreciarse en las gráficas de la figura 98. Observando un error de posición final de 0. 412519º para el sistema con la tarjeta de adquisición de datos y control realizada en la tesis [7] y un error de posición final de 0. 41324 º para el sistema con la tarjeta PCI [6].

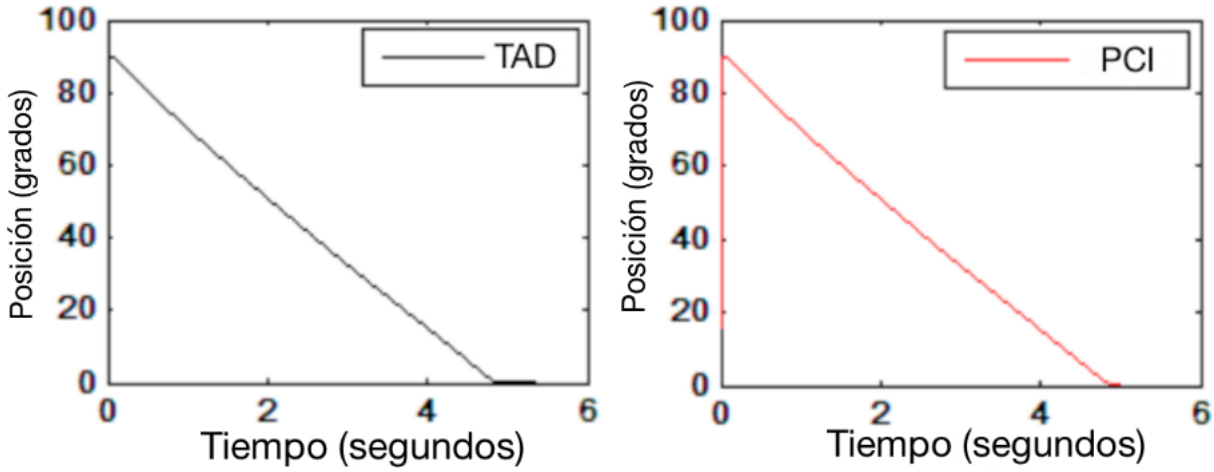


Figura 98: Gráfica de error de posición.

En la figura 99, se muestra la respuesta de par en ambos sistemas. La respuesta de par del sistema con la tarjeta PCI presenta una respuesta con una oscilación mayor respecto de la respuesta de par del sistema con la tarjeta de adquisición de datos y control diseñada en la tesis [7].

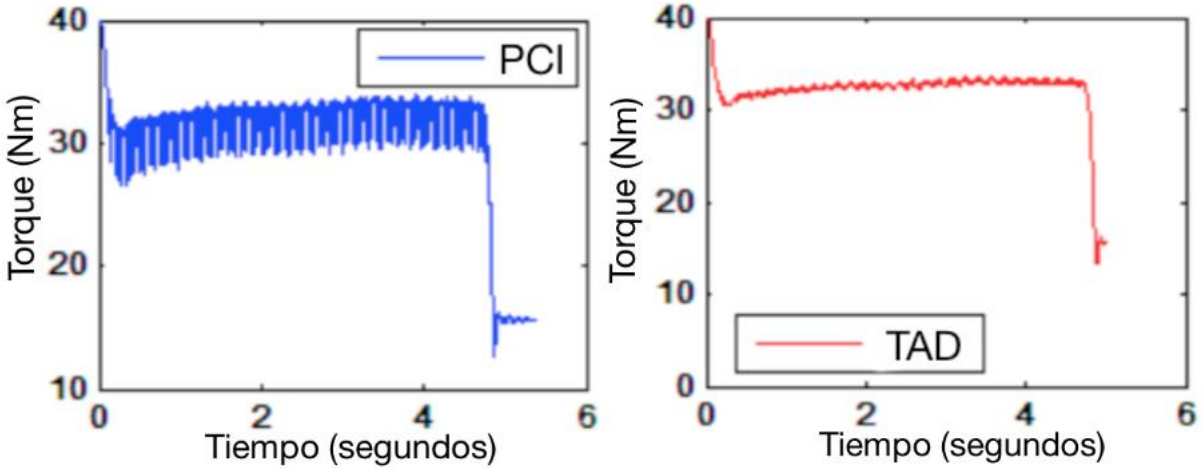


Figura 99: Grafica de par del motor.

En la figura 100, se muestran las gráficas de evolución de la posición para ambos sistemas de control durante el experimento. Observando una posición final de 89. 98958 º para el sistema con la tarjeta de adquisición de datos y control realizada en la tesis [7] y una posición final de 89. 3312 º para el sistema con la tarjeta PCI [6], la diferencia entre ambas posiciones finales es de 0.72838 º.

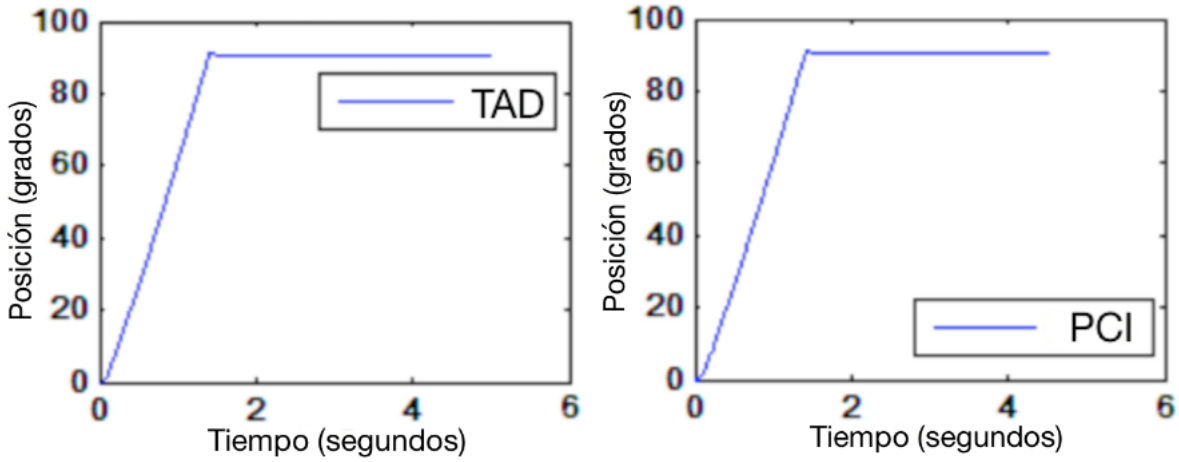


Figura 100: Grafica posición del motor 2.

Conclusiones generales

De acuerdo a los resultados presentados en este trabajo de tesis podemos concluir que el objetivo general y los objetivos particulares se han cumplido satisfactoriamente.

La contribución más importante de este trabajo es haber desarrollado una tarjeta de arquitectura abierta que une las ventajas de una computadora y un FPGA, lo anterior permite realizar proyectos de control, automatización, instrumentación, etc., en muy poco tiempo. Además, brinda la ventaja de tener una conexión inalámbrica para la descarga de programas y datos para su configuración. Otra ventaja es que después de la configuración si es necesario la tarjeta puede llevar a cabo los procesos de manera independiente.

El contar con esta tarjeta ha permitido que tres estudiantes de la maestría en Ciencias de la Electrónica que habían terminado los sistemas mecánicos y de potencia de sus robots lograran controlarlos en tan solo 3 semanas, lo cual redujo sustancialmente el tiempo de integración de cada robot, que normalmente tomaba de 4 a 6 meses, ya que anteriormente los robots se tenían que acoplar a una tarjeta de interfaz basada en un FPGA y conectada a una PC. Las aplicaciones se mostraron en el Capítulo 4. Otra ventaja es la reducción del cableado que antes se tenía hacer entre la PC, el FPGA y el robot, y ahora casi todas esas conexiones se realizan dentro del FPGA.

Aunque el número de instrucciones es básico ha mostrado ser suficiente para controlar robots de hasta 5 grados de libertad. Otra ventaja es que al ser de arquitectura abierta permite que el número de instrucciones se pueda seguir incrementando.

Como trabajo futuro queda el implementar un coprocesador matemático que calcule funciones exponenciales, trigonométricas e hiperbólicas, que son usadas en control y automatización.

La tarjeta queda como parte de la infraestructura de la Maestría de Ciencias de la Electrónica opción Automatización, por lo que espera que sea usada como base de los sistemas de control de los robots y naves no tripuladas desarrolladas por varios estudiantes de la maestría.

Referencias

[1] Wolf M., (2016). Computers as Components: Principles of Embedded Computer Systems Desing. USA: Morgan Kaufman.

[2] Dubey R. (2009), Introduction to Embedded System Design Using Field Programmable Gate Arrays. India:Springer

[3] Xilinx, Inc. (2016). MicroBlaze Soft Processor Core. Consultado en octubre del 2016, http://www.xilinx.com/products/design-tools/microblaze.html

[4] Altera, Inc. (2016). Nios II Processor. Consutada en noviembre del 2016, https://www.altera.com/products/processors/overview.html

[5] Caysslais R. (2014), Sistemas embebidos en FPGA. México: ALFAOMEGA.

[6] Bezares G. H. (2014), Robot CNC multiherramienta. México: Tesis de Maestría, Benemérita Universidad Autónoma de Puebla.

[7] Moya J. De J. (2017). Tarjeta para controlar 3 grados de libertad de robots vía WiFi. Tesis de Maestría, Benemérita Universidad Autónoma de Puebla.

[8] MIPS Computer Systems Inc. MIPS Proccessors. Consultado en noviembre del 2016, https://www.mips.com/

[9] Patterson D., Hennessy J (2000). Estructura y diseño de computadores. USA:Reverté S.A.

[10] Altera Inc (2016). Floating-Point IP Cores User Guide. Consultada en septiembre 2016, https://www.altera.com/documentation/eis1410764818924.html

[11] "IEEE Standard for Floating-Point Arithmetic" in IEEE std 754-2008, Agosto 29 del 2008, pp. 1-70.

[12] Ritpurkar S, Thakare M. Korde D. (2015). Desing and Simulation of 32-bit RISC Architecture Based on MIPS using VHDL. ICACCS.

[13] Altera Inc. (2016) The Automotive-Grade Device Handbook. Consultado en agosto del 2016. https://www.altera.com/documentation/mcn1407908847901.html

[14] Altera Inc. 14. Serial Configuration Devices Data Sheet. Consultado en agosto del 2016.https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/hb/cyc/cyc_c51014.pdf

[15] http://www.datasheetcatalog.com/datasheets_pdf/T/P/S/7/TPS75501KTT.shtml

[16] http://www.datasheetcatalog.com/datasheets_pdf/T/L/C/7/TLC7701.shtml

[17] http://search.datasheetcatalog.net/key/23LC1024-I/SN

[18] http://www.datasheetcatalog.com/datasheets_pdf/S/S/T/2/SST26VF032B-104I_SM.shtml

[19] López J. R. (2014). Control de un robot articular de 1 grado de libertad utilizando un sistema embebido configurable por medio de una interfaz inalámbrica. México: Tesis de Maestría. Benemérita Universidad Autónoma de Puebla.

[20] Torres M. (2005). Diseño e ingeniería electrónica asistida con protel dxp. México: ALFAOMEGA.

[21] De Gante R. A. (2016). Diseño y construcción de un sistema de adquisición de datos. México: Tesis Maestría. Benemérita Universidad Autónoma de Puebla.

[22] Terasic Inc (2016). DEO-Nano Development and Education Board. Consultado en septiembre 2016, http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=139&No=593&PartNo=4

[23] Gallego O. A., Vázquez T. R., Lasa A. I (2004). Análisis Básico de Circuitos Eléctricos y Electrónicos. España: Pearson Educación.

[24] National Instrument Inc. ¿Qué significa el termino Endian?. Consultado en septiembre 2016, http://digital.ni.com/public.nsf/allkb/0D9FE6005669BAED8625708C007940BD

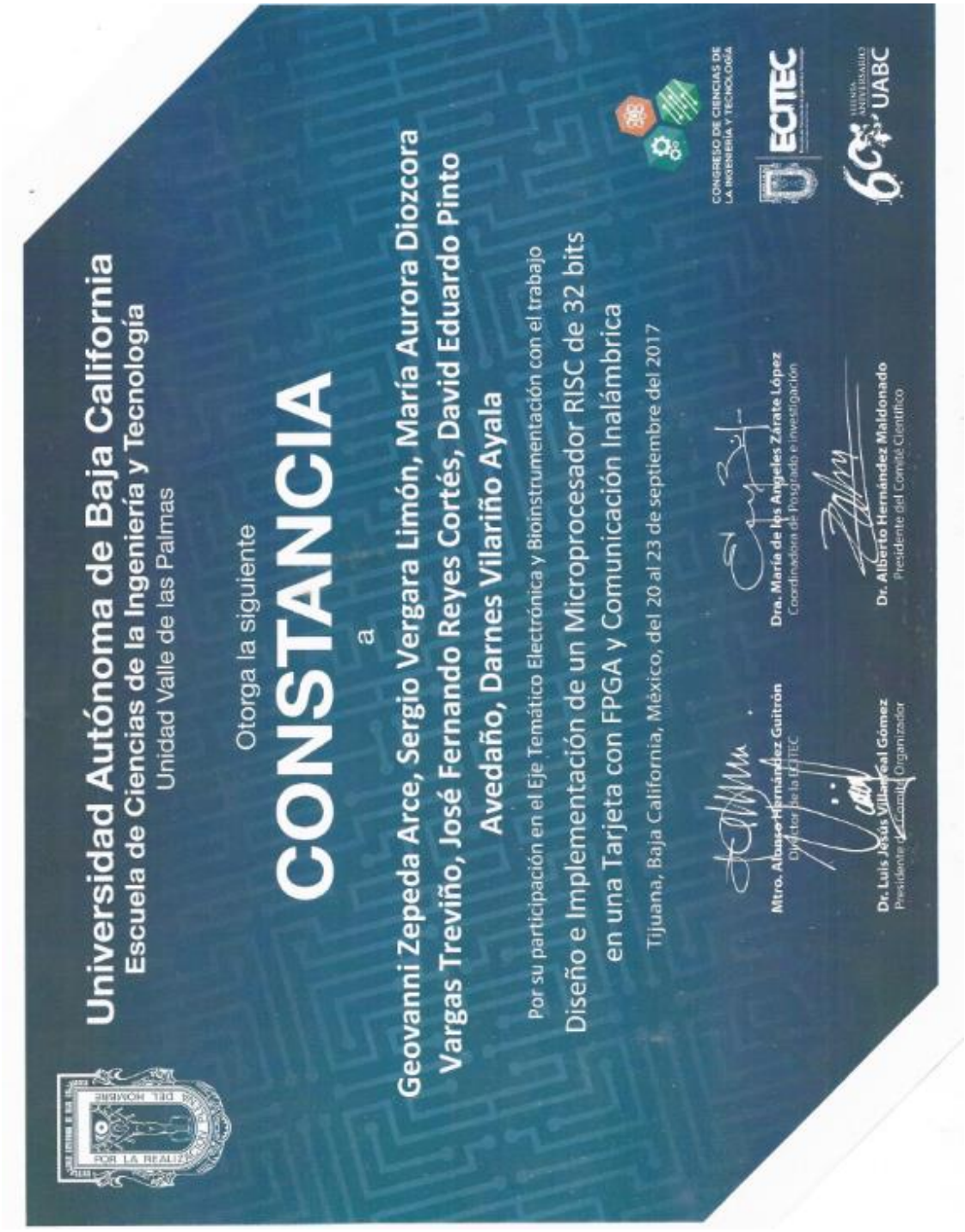
[25] López F. J., (2017). Desarrollo de un sistema mecatrónico para robot humanoide que permita emular el movimiento del cuello de los seres humanos. México: Tesis de Maestría, Benemérita Universidad Autónoma de Puebla.

[26] Herrera F., (2017). Desarrollo de una mano mecatrónica de bajo costo para un robot humanoide. México: Tesis de Maestría, Benemérita Universidad Autónoma de Puebla.

[27] López A. (2016) Robot CNC para impresoras 3D. Tesis de Maestría, Benemérita Universidad Autónoma de Puebla.

Apéndice A

Participación en congreso: Constancia de ponencia.



Apéndice B

Artículo.

Revista de Ciencias de la Ingeniería y Tecnología (RECIT). Vol. 1. No. 1. 2017

Diseño e implementación de un Microprocesador RISC de 32 bits en una Tarjeta con FPGA y Comunicación Inalámbrica.

Zepeda Arce G.[1], Vergara Limon S.[1], Vargas Treviño M. A.[1], Reyes Cortés J. F.[1]
Pinto Avedaño D. E.[2], Vilariño Ayala D.[2]

[1]Facultad de Ciencias de la Electrónica, Benemérita Universidad Autónoma de Puebla. México.
gio140987@gmail.com, svergara2@hotmail.com, auroravargast@hotmail.com, fernando.reyes@correo.buap.mx
[2]Facultad de Ciencias de la Computación, Benemérita Universidad Autónoma de Puebla. México.
dpinto@cs.buap.mx, darnes@cs.buap.mx

Resumen. – *En este trabajo se presenta el diseño e implementación de un microprocesador RISC de 32 bits con arquitectura Harvard. El microprocesador cuenta con 15 instrucciones las cuales incluyen operaciones aritméticas de punto flotante (suma, resta, multiplicación y división), transferencia de datos y control. Para implementar estas instrucciones el diseño incorpora los módulos de Unidad de Control (CU), Unidad de Punto Flotante (FPU), comparador y controladores de memoria. El diseño de cada elemento del microprocesador se desarrolló con el Lenguaje de Descripción de Hardware de Altera (AHDL). Para la instrumentación del sistema se diseña una tarjeta electrónica que tiene como base un FPGA (Field Programmable Gate Array) de la familia Cyclone IV, una memoria RAM, una ROM y un módulo de interfaz inalámbrica Wi-Fi. Mediante la interfaz inalámbrica se validar el sistema completo, cargando un programa de prueba para ejecutar un algoritmo y visualizar el resultado desde un equipo de cómputo.*

Palabras clave: FPGA; Microprocesador RISC de 32 bits; memoria (RAM y ROM); Wi-Fi.

Abstract. - *This paper presents the design and implementation of a 32-bit RISC microprocessor with Harvard architecture. The microprocessor has 15 instructions which include floating-point arithmetic operations (addition, subtraction, multiplication and division), data transfer and control. To implement these instructions the design incorporates the Control Unit (CU), Floating Point Unit (FPU), comparator and memory controller modules. Each microprocessor component was designed using the Altera Hardware Description Language (AHDL). For the instrumentation of the system, an electronic card based on a Field Programmable Gate Array (FPGA) of the Cyclone IV family, a RAM memory, a ROM and a Wi-Fi wireless interface module is designed. Using the wireless interface, the entire system is validated, loading a test program to execute an algorithm and displaying the result from a computer.*

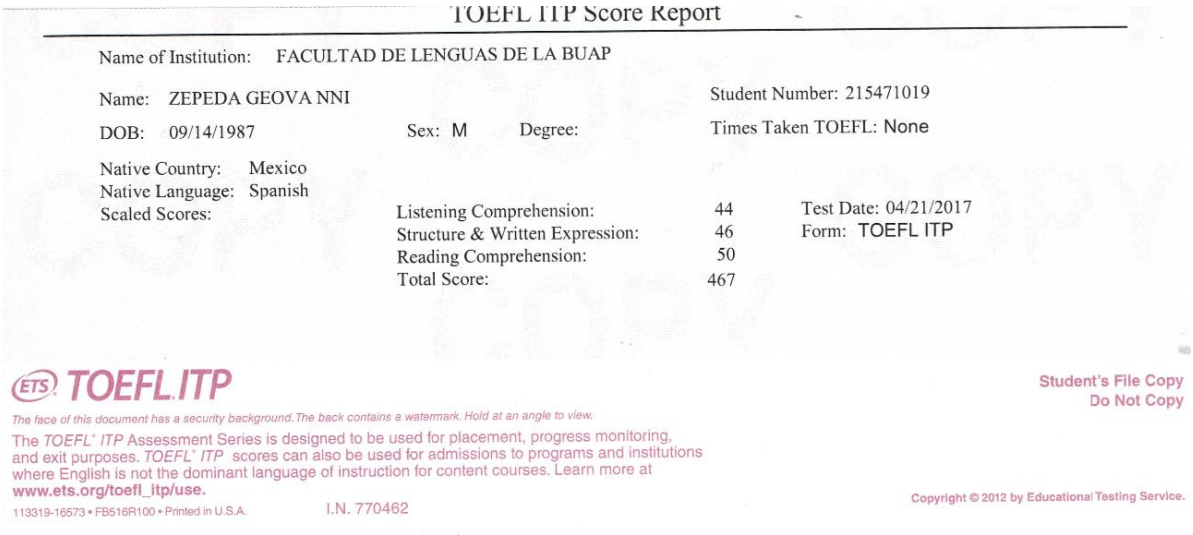
Key words: FPGA; 32-bit RISC microprocessor; memory (RAM and ROM); Wi-Fi.

1. Introducción

El empleo de FPGAs tiene ciertas ventajas, entre ellas la flexibilidad que se obtiene al desarrollar un prototipo. Un microprocesador o microcontrolador comercial tiene ciertas funciones que no pueden ser modificadas después de ser adquiridas. En cambio, un FPGA puede ser reprogramado en poco tiempo con las funciones específicas que requiere un proyecto. Esto da lugar al cómputo reconfigurable, que tiene ciertas ventajas sobre dispositivos con una función fija, por ejemplo reducción de costos, mayor desempeño y confiabilidad, además de la implementación de interfaces especializadas de alta velocidad [1]. Por otra parte, en la actualidad la microelectrónica ha hecho posible desarrollar sistemas completos dentro de un solo circuito integrado SOC (System On Chip), con lo cual se han mejorado de manera notoria características

Apéndice C

Certificado TOEFL



TOEFL ITP Score Report

Name of Institution: FACULTAD DE LENGUAS DE LA BUAP

Name: ZEPEDA GEOVA NNI

Student Number: 215471019

DOB: 09/14/1987 Sex: M Degree:

Times Taken TOEFL: None

Native Country: Mexico
Native Language: Spanish
Scaled Scores:

Listening Comprehension:	44	Test Date: 04/21/2017
Structure & Written Expression:	46	Form: TOEFL ITP
Reading Comprehension:	50	
Total Score:	467	

ETS TOEFL ITP

Student's File Copy
Do Not Copy

The face of this document has a security background. The back contains a watermark. Hold at an angle to view.

The *TOEFL® ITP* Assessment Series is designed to be used for placement, progress monitoring, and exit purposes. *TOEFL® ITP* scores can also be used for admissions to programs and institutions where English is not the dominant language of instruction for content courses. Learn more at **www.ets.org/toefl_itp/use.**

113319-16573 • FB516R100 • Printed in U.S.A.

I.N. 770462

Copyright © 2012 by Educational Testing Service.