



**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO**

---

**FACULTAD DE INGENIERÍA**

**Diseño de un  
microcontrolador de 32 bits  
con base en un FPGA**

**TESIS**

Que para obtener el título de  
**Ingeniero Eléctrico - Electrónico**

**P R E S E N T A**

Manuel Pérez Aguilar

**DIRECTOR DE TESIS**

M.I. Juan Carlos Roa Beiza



Ciudad Universitaria, Cd. Mx., 2017

“La cosa más hermosa que podemos experimentar es el misterio.

Es la fuente de toda arte y toda ciencia.”

- EINSTEIN, Albert

**Agradecimientos:**

Agradezco a la U.N.A.M que me recibió como estudiante y el haberme formado en sus aulas desde preparatoria, a la Facultad de Ingeniería por la preparación que me ha permitido enfrentarme a los retos de la vida profesional, a cada uno de los profesores que me dedicó tiempo, y compartió conmigo no solo información sino formación para la vida, a mi Director de Tesis el M.I. Juan Carlos Roa Beiza por ser tan generoso en compartir su conocimiento y guiarme durante el presente trabajo de Tesis, al Programa de Apoyo a la Titulación (PAT) y al Programa de Vinculación con Egresados (PVE) por brindarme esta oportunidad para culminar una de mis metas, a mis amigos que han hecho de mi etapa universitaria un trayecto de vivencias que nunca olvidaré.

Son muchas las personas que han formado parte de mi vida profesional a las que me gustaría agradecerles su amistad, apoyo, ánimo, consejos y compañía durante esta etapa de mi vida, que sin importar en donde se encuentren, quiero darles las gracias por formar parte de mí y por todo lo que me han brindado.

A todo ellos, muchas gracias.

**Dedicatoria:**

Dedico este Trabajo de Tesis a mis padres, Manuel Perez Grande y María Alejandra Mónica Aguilar Tlapale por el inmenso cariño que me han dado toda la vida, por su apoyo y generosidad sin límites.

# Índice

Objetivo.....	1
RESULTADOS ESPERADOS.....	1
Capítulo 1 ANTECEDENTES.....	2
1.1 Introducción.....	2
1.2. Estado del arte, Sistemas embebidos.....	6
1.3.Estado del arte de microcontroladores comerciales y de uso industrial.....	10
1.4. Acotamiento del problema.....	14
1.5. Diseño a bloques de la computadora de 32 bits para instrumentación.....	15
Capítulo 2. TEORÍA BÁSICA.....	19
2.1 Marco teórico para el diseño de un microcontrolador con base en un FPGA...	19
2.2. Dispositivos Lógicos Programables, estructura interna y clasificación.....	21
2.2.1. Clasificación de los PLDs.....	23
2.3. FPGA de alto nivel de integración y lenguaje HDL.....	25
2.3.1 FPGA.....	26
2.3.2 HDL.....	29
2.4 Bloques de Propiedad Intelectual y SOFT CORE.....	29
2.4.1 SOFT CORE.....	30
2.5 BUSES DE COMUNICACIÓN: BUS I2C, Hardware I2C, Protocolo de comunicación I2C, Comunicación SPI y UART.....	33
2.5.1 BUS I2C.....	33
2.5.2 Hardware del I2C.....	33
2.5.3 Protocolo de comunicación I2C.....	35
2.5.4 COMUNICACIÓN SPI.....	36
2.5.5 UART.....	39
2.6 Transceptores para módulos de telemetría.....	42
2.6.1 Transceptores.....	42
Capítulo 3. ANÁLISIS Y DISEÑO DE LA PROPUESTA A IMPLEMENTAR.....	44
3.1 Requerimientos Generales del microcontrolador.....	45
3.2 Diseño conceptual del prototipo a realizar.....	46

3.3 Elección de FPGA.....	48
3.4. Estrategia de desarrollo por modulo .....	54
3.5.Diseño del hardware en el FPGA.....	54
Capítulo 4. DESARROLLO Y CONSTRUCCIÓN.....	74
4.1 Implementación del núcleo de la computadora, pruebas y resultados .....	74
4.2. Construcción y prueba del módulo UART, pruebas y resultados .....	78
4.3 Desarrollo y prueba de módulo GPIO e interrupciones .....	82
4.4. Evaluación y prueba del módulo I2C .....	91
4.5 Prueba de módulo SPI .....	96
4.6 Prueba del microcontrolador mediante simulación por software.....	99
4.7 Pruebas de Telemetría.....	104
CONCLUSIONES.....	107
BIBLIOGRAFÍA .....	108

# **Diseño de un microcontrolador de 32 bits con base en un FPGA**

## **Objetivo**

Diseñar e implementar un microcontrolador de 32 bits, con base en un FPGA, para uso en instrumentación electrónica y manejo de información remota.

La compañía desea crear su propia infraestructura de procesamiento para no depender de terceros. Y así poder aplicarla a resolver todas las necesidades que vayan surgiendo de recolección de información en sus plantas.

La cantidad de información es muy robusta por lo cual la compañía decidió que se hiciera un sistema a la medida de tecnología propia.

## **RESULTADOS ESPERADOS.**

- Se presenta el desarrollo de un microcontrolador de propósito general con base en un FPGA para una aplicación de alta tecnología en la industria petrolera, el cual va a tener la capacidad de poder implementar los protocolos de comunicación más utilizados para el manejo de sensores y actuadores útiles en aplicaciones industriales.
- Se dará la pauta para el desarrollo de este tipo de aplicaciones, proporcionando la información y herramientas necesarias para su implementación.
- Se utilizará un FPGA comercial que posea las características más avanzadas para este desarrollo.
- Se ejecutará la evaluación del prototipo a través de utilerías del fabricante.
- Se presentará el prototipo físicamente y la evaluación del mismo
- Se desarrollará una metodología para el análisis y diseño de sistemas de ultra escala de integración

# Capítulo 1 ANTECEDENTES

## 1.1 Introducción

El microcontrolador es uno de los inventos más sobresalientes del siglo pasado, en la actualidad aporta soluciones al ser humano en comunicación, entretenimiento, salud, seguridad, en general existen millones de soluciones, implementadas a base de microcontroladores.

El núcleo de un microcontrolador (MCU) es un microprocesador (CPU). Siendo éste un dispositivo lógico secuencial utilizado en sistemas electrónicos digitales el cual realiza operaciones aritméticas, lógicas y de control.

En la actualidad, los microcontroladores han alcanzado capacidades de manipulación de datos de 32 bits hasta 64 bits, que los hace muy poderosos en cálculos matemáticos y lógicos.

Haciendo una definición de un microcontrolador; es un dispositivo lógico secuencial, el cual realiza operaciones aritméticas, lógicas y utiliza sus periféricos para tener comunicación con el exterior.

A continuación, se resaltan algunos aspectos importantes en la tecnológica de los MCU.

- Capacidad en bits: Cuando escuchamos que un procesador es de 4, 8, 16, 32 ó 64 bits, nos estamos refiriendo a procesadores que realizan sus operaciones con registros de datos de ese tamaño, y por supuesto, esto determina muchas de las potencialidades de un microcontrolador.

•Tecnología de memoria; Para guardar el programa a ejecutar éste era almacenado en una memoria ROM de máscara y era el fabricante quien lo generaba, la desventaja de esta tecnología eran los altos volúmenes de producción para que fuera más económico, más tarde aparece la memoria EPROM, un gran inconveniente de esta tecnología era la demora en el borrado de los datos, debido a la larga exposición del chip ante lámparas de luz ultra violeta, posteriormente esta tecnología es mejorada por la EEPROM, que permite grabar y borrar eléctricamente los datos, finalmente aparece la tecnología FLASH la que se impone en los mercados por su bajo costo, velocidad y facilidad de manipulación en la programación.

•Velocidad de reloj: Para ejecutar tareas que demanden una alta velocidad de procesamiento de datos se debe escoger con cuidado la frecuencia en la que opera su procesador, así mismo éstos además pueden ofrecer bajo consumo y un costo razonable.

•Capacidad de memoria; Debido al código que se escribe para ejecutar algún proceso, éste ocupa un cierto espacio en memoria y esto se convierte en el factor determinante para un desarrollo con microcontrolador.

La memoria generalmente es interna, pero existen máquinas con manejo de memoria externa y pueden acceder a varios megabytes, tanto en memoria de código como datos. En general un microcontrolador no requiere de grandes cantidades de memoria en código, en este sentido los microcontroladores vienen equipados con memoria de código de hasta 1MB y de datos de hasta 128KB.

•Los módulos o periféricos Gran variedad de periféricos orientados a los diferentes mercados tecnológicos como: las telecomunicaciones, el transporte, aparatos eléctricos de consumo masivo, entre otros; son implementados en los microcontroladores. Los usuarios encontrarán una gran variedad de periféricos como:

- Temporizadores

- Convertidores análogo a digital
- Entradas/salidas de propósito general
- Reloj de tiempo real
- Puertos de comunicación serial asíncrona (UART, CAN)
- Puertos de comunicación serial sincrónica (IIC, SPI)
- Bus universal de comunicación serial (USB)
- Puerta trasera de depuración (BDM, JTAG)

Algunas máquinas de mayor desempeño involucran módulos más especializados como:

- Controladores para ETHERNET
- Unidades de generación y aceleración criptográfica
- Unidades de generación de números aleatorios
- Unidades de generación y verificación de código de redundancia cíclica
- Unidades de tratamiento de aritmética flotante (FPU)
- Unidades de multiplicación, acumulación y corrimiento (MAC)
- Unidades para manejo directo de memoria (DMA)

En el presente trabajo se muestra el diseño de un microcontrolador de 32 bits, con base en un FPGA para uso en instrumentación electrónica. Por ello esta tesis se divide en 5 capítulos:

El capítulo 1 comprenderá una investigación en el estado del arte para mostrar los requerimientos que poseen los microcontroladores de 32 bits actuales, tecnologías usadas, protocolos de comunicación, entre los requerimientos más importantes, para posteriormente acotar el diseño a implementar un microcontrolador de 32 bits de propósito general, que se encuentre al margen de los microcontroladores usados actualmente en la industria.

El capítulo 2 proporcionará los antecedentes necesarios para la implementación de los módulos de un microcontrolador diseñado con base en un FPGA, para esto se centra el marco conceptual de un FPGA para después abarcar la descripción y funcionamiento de las comunicaciones seriales más empleadas.

El capítulo 3 explicará los requerimientos propuestos de nuestro microcontrolador, así como su diseño de forma conceptual, por lo que posteriormente se dan los resultados de una investigación y el fundamento técnico de la elección del FPGA a usar, y por último su implementación en el FPGA, ya que se trata de un diseño complejo, se explica el uso de la técnica de High Level Synthesis (HLS) y de las herramientas usadas.

El capítulo 4 explicará el desarrollo del microcontrolador en HLS por módulos de forma que se va describiendo su implementación de cada uno, para después dar resultados de las pruebas realizadas del microcontrolador basándonos en utilerías proporcionadas por el fabricante, las cuales nos permiten evaluar el software y hardware diseñado.

La motivación en la elaboración de este tema de tesis, es la importancia del desarrollo de aplicaciones con base en un FPGA debido a que con modificaciones en su hardware descriptivo e implementación de funcionalidades extras, da posibilidad de escalar a proyectos más robustos en un futuro cercano, buscando aprovechar los recursos tangibles e intangibles que existen en nuestro país, generando autonomía en tecnología en México.

Como principal objetivo de este trabajo es el diseño e implementación de un microcontrolador de 32 bits con base en un FPGA, para uso de instrumentación electrónica y manejo de información remota, esto con el fin de realizar mediciones en tuberías de una compañía petrolera, el cual va a tener la capacidad de poder implementar los protocolos de comunicación más utilizados para el manejo de sensores y actuadores útiles en aplicaciones industriales, que dará la pauta para el desarrollo de este tipo de aplicaciones, proporcionando la información y

herramientas necesarias para su implementación, se utilizará un FPGA comercial que posea las características más avanzadas para este desarrollo, se usará una metodología para el análisis y diseño de sistemas de ultra escala de integración, se ejecutará la evaluación del prototipo a través de utilerías del fabricante.

## **1.2. Estado del arte, Sistemas embebidos**

Un sistema embebido según la publicación de J. G. Tong de la universidad de Windsor del departamento de ingeniería eléctrica y en computación del centro de investigación para microistemas integrados explica que; “Los sistemas embebidos son componentes en hardware y software funcionando conjuntamente para realizar una función en específico”.

Por lo que se puede entender que un sistema embebido es un sistema basado en un procesador acompañado de hardware adicional, los cuales funcionan de manera conjunta para ejecutar un rango de funciones de una tarea en específico.

Un sistema embebido está constituido de módulos y/o partes electrónicas las cuales realizan tareas tales como adquisición de datos y su procesamiento, ejecución de comandos, manejo de puertos de entrada/salida.

Las principales características que debe tener un sistema embebido son tres:

1. Debe contener hardware el cual cubra el propósito de la aplicación
2. Debe contener una aplicación en software que cumpla con las tareas a realizar
3. Debe tener un sistema operativo en tiempo real (RTOS, por sus siglas en Inglés) que supervise la aplicación en software y proporcione un mecanismo que permita al procesador realizar varias tareas, es importante mencionar que un Sistema embebido de pequeña escala puede o no contener un RTOS

Entre el hardware principal que debe cubrir un sistema embebido se compone de un procesador o microcontrolador; este se elige para procesar la información ya que aporta capacidad de cómputo al sistema, una memoria; como elemento de almacenamiento de software y datos, periféricos; que se utilizan para tener comunicación de comandos externos y/o control de subsistemas que pudieran ser requeridos, además debe contener su propio software; en donde éste define la función del sistema y puede consistir de manejo de errores, depurador y soporte de mantenimiento.

### **Clasificación de los sistemas embebidos.**

Sistemas embebidos de pequeña escala: Estos sistemas están diseñados con un microcontrolador o procesador de 8 ó 16 bits, contienen poca complejidad en hardware y software, son operados con baterías. Unos ejemplos de sistemas embebidos de pequeña escala son [33]:

- Máquinas dispensadoras
- Sistemas de lavado
- Sistemas de control de motores de corriente directa
- Sistemas de adquisición de datos

Sistemas embebidos de mediana escala: Estos sistemas son usualmente diseñados con procesadores o microcontroladores de 16 ó 32 bits, con arquitectura RISC, estos sistemas tienen un grado de complejidad elevado por lo que deben ser diseñados con herramientas de software que permitan el manejo de RTOS, simulación, depurador y un entorno de desarrollo integrado (IDE, por sus siglas en Inglés), pueden usar en su desarrollo e implementación IPs (explicado más adelante).

Unos ejemplos de sistemas embebidos de mediana escala son:

- Dispositivos para sistemas de redes como: Router, Switch, Hub y Gateway

- Sistemas de entretenimiento
- Sistemas de bancos, por ejemplo: ATM y transacciones de tarjetas de crédito
- Sistemas de rastreo
- Sistemas de procesamiento de imagen

Sistemas sofisticados embebidos: Estos sistemas se caracterizan por tener complejidades en hardware y software, los cuales necesitan procesadores escalables o arreglos de compuertas programables, se utilizan para aplicaciones de vanguardia que necesitan co-diseño de hardware y software, esto para enfocarse a una optimización en el manejo de fuertes restricciones de tiempo asociadas con el RTOS. Las herramientas de desarrollo para estos sistemas pueden no estar fácilmente disponibles.

Los ejemplos más comunes de estos tipos de sistemas embebidos sofisticados son:

- Sistemas embebidos para redes “wireless LAN”
- Sistemas embebidos para video en tiempo real o sistemas de procesamiento multimedia
- Interfaces embebidas y sistemas de redes de trabajo usando “high speed” (400 MHz plus), “Ultra high speed” y anchos de bandas largos para Routers, “Switches” y “Gateways”

### **Sistemas embebidos en FPGAs**

Los FPGAs poseen ventajas de una alta velocidad de operación, capacidad de reconfiguración, un número muy grande de componentes y protocolos soportados.

En los sistemas embebidos, los FPGA se utilizan de dos maneras: para implementar las funcionalidades deseadas directamente en la lógica digital, o bien por la implementación de la arquitectura de un microprocesador; esto mediante un

procesador y sus periféricos deseados. Por lo que al implementar un sistema embebido en un FPGA conduce a la idea de una computación reconfigurable.

Al diseñar sistemas embebidos en un FPGA existe una solución fija en el diseño de PCB ya que las matrices basadas en RAM del FPGA ofrecen una reprogramación fácil sin cambiar el diseño del PCB.

El diseño de sistemas embebidos se está volviendo cada vez más difícil debido a las limitaciones estrictas del área del dispositivo lógico, el consumo de energía y el rendimiento que se necesita lograr. Además de estas limitaciones, muchos desarrolladores de sistemas embebidos se enfrentan a plazos ajustados para el mercado.

Por lo tanto, la metodología de programas de diseño de hardware / software se utiliza a menudo para diseñar sistemas embebidos con el fin de ayudar a reducir la cantidad de tiempo dedicado al desarrollo y la depuración.

A medida que aumenta la complejidad de los diseños de sistemas embebidos, el diseño de éstos a partir desde cero de cada componente de hardware, llegó a ser poco práctico y costoso para la mayoría de los diseñadores. Por lo tanto, la idea de usar núcleos de propiedad intelectual (IP) pre-diseñados y pre-probados se convirtió en una alternativa atractiva

### 1.3.Estado del arte de microcontroladores comerciales y de uso industrial.

A continuación, se presentan una revisión de los últimos módulos de microcontroladores de carácter comercial e industrial.

#### MICROCHIP (ATMEL)

Enfocándonos a un fácil uso de implementación, consumo de baja potencia y alto nivel de integración tenemos que la compañía microchip ofrece microcontroladores de 8 a 32 bits, estos dispositivos ofrecen una combinación de bajo consumo de potencia y flexibilidad para el diseño de aplicaciones, estos están basados en la arquitectura de programación más eficiente en la industria que es C y ensamblador [1] [3].

Entre los microcontroladores de 32 bits que destacan de Atmel se muestran en la tabla 1.3.1.

Familia de los dispositivos	Aplicaciones	Tecnología	Características
32-bits AVR UC3	Propósito general	picoPower,SleepWalking,FlashVault ,DMA,Event System	16-512 KB Flash,48-144 pins,66 MHz
megaAVR MCU	Propósito General	PTC,picoPower,SleepWalking,EEP ROM	4-256KB Flash,28-100 pins, 20 MHz
Smart SAM E ARM Cortex-M7 MCUs	Industrial	Wireless,low power	2MB Flash,384kB SRAM,300MHZ, Multiple serial communications,100-144 pins

Smart SAM3s Cortex-M3 based	Industrial	ARM processor,low power	64KB a 256KB Flash,64MHz,Multiple high-speed communication peripherals
Smart SAM4E Cortex-M4 based	Industrial	ARM processor,low power	512KB a 1MB Flash,128KB SRAM, 2KB cache,120MHz

Tabla 1.3.1 Microcontroladores de ATMEL [2]

## TEXAS INSTRUMENT

Por parte de Texas Instrument ofrecen una amplia variedad de microcontroladores para la industria que abarcan la solución a los sectores de automatización, soluciones a la medida, control de motores y el control lógico de sensores [4], vea tabla 1.3.2.

Familia de los dispositivos	Aplicación	Características
MSP432P4x MCUs	Propósito general	48MHz,256KB Flash,64kB SRAM,32-bits ARM Cortex
C2000™ Piccolo™ MCUs	Aplicaciones de control	120 MHz,12KB Flash,100KB SRAM,32 bits, incluye DSP,QEP,CAN,LIN,USB
C2000™ Delfino™ MCUs	Aplicaciones Industriales	300 MHz,1 MB Flash,516KB SRAM,32 bits, PWM, ADC, incluye QEP,CAN,LIN,USB, interfaces seriales
CC26x Wireless MCUs	Aplicaciones Industriales e inalámbricas	48MHz,128 KB Flash,20Kb SRAM 32 bits ARM M3,AES,GPIO,Timers,UART,SPI,I2C,DMA

Tabla 1.3.2 Microcontroladores de TI [5]

## MICROSEMI

Microsemi para el sector industrial ofrece FPGA con características de bajo consumo y alta confiabilidad con procesadores ARM Cortex-M3 embebidos, lo cual asegura un funcionamiento óptimo en automatización de procesos, Smart energy, sistemas de control de vuelo entre las que destacan [6], véa tabla 1.3.3.

<b>FPGA</b>	<b>Características</b>
IGLOO series	330-35K Les,04 kb SRAM, PLLs,32 bits ARM Cortex M1
ProASIC3 FPGA	ARM Cortex M1,100 a 3 K Les
SMARTFUSION	ARM Cortex M3,512 KB Flash, 64 KB SRAM,700 - 6K Les
SAMRTFUSION2	ARM Cortex M3,10K Les,lowest Power

Tabla 1.3.3 Microcontroladores de Microsemi

## XILINX

Para el sector industrial tenemos por parte de Xilinx una solución para implementar procesos que requieran altos grados de velocidad de procesamiento, versatilidad, interfaces múltiples sin perder de vista una reducción en costo, lo cual es usar e implementar funciones en FPGAs, por lo cual xilinx muestra los dispositivos para automatización industrial [7] [8], vea tabla 1.3.4.

<b>Dispositivo</b>	<b>Características</b>
Zynq 7000	ARM cortex-A9,NEON,controlador de memoria endurecido,CAN,USB,Tri Gigabit Ethernet,SD,UARTs, Linux

Tabla 1.3.4 FPGAs de Xilinx

Comparando los microcontroladores comerciales más usados en la industria, se observa que aparte del uso de procesadores de 32 bits y tipo ARM, hay familias diseñadas específicamente para tener procesadores embebidos en dispositivos FPGA, esto hace que el diseñador pueda implementar sistemas más robustos para tener facilidad de procesar información en paralelo siendo una de las principales ventajas de un FPGA, dando un ejemplo, es el uso de sensores que demanden velocidades de procesamiento altos como los sensores ópticos, los cuales requieren tener un dispositivo con la capacidad de configurarlos, la obtención y procesamiento de sus datos obtenidos.

Por lo que se analizó que es conveniente migrar el diseño de un microcontrolador e implementarlo con base en un FPGA, en los cuales se puede hacer una descripción de hardware de todo un microcontrolador.

## 1.4. Acotamiento del problema.

Como se observó en el estado del arte la, tecnología está migrando al uso de microcontroladores basados en FPGA por lo que en esta tesis se utiliza un FPGA para el diseño e implementación de un microcontrolador, como principal ventaja al emplear un FPGA es que este dispositivo es flexible lo que significa que se le pueden agregar o quitar funcionalidades como sean necesarias y en este sentido presenta una gran ventaja debido a su capacidad de reconfiguración sobre los ASICs, DSP y  $\mu$ C, característica que lo hace muy atractivo para el desarrollo de aplicaciones industriales que requieran un constante mantenimiento y aplicaciones que demandan alta capacidad de reconfiguración, además que en el diseño de aplicaciones en FPGAs se tienen la ventaja de trabajar en lenguajes estandarizados por lo que si se llegara a cambiar el FPGA, el diseño realizado es implementado fácilmente en el nuevo hardware.

Se pretende desarrollar un microcontrolador siguiendo la tendencia en el estado del arte al utilizar un FPGA, con lo siguiente:

- Utilizar un FPGA de tecnología SRAM de última generación; para que éste presente las características más avanzadas del mercado y se pueda implementar funcionalidades complejas a futuro.
- Tener un microcontrolador de 32 bits de bajo consumo.
- Desarrollar el proyecto en un entorno de desarrollo que este en constante mantenimiento y tenga suficiente documentación para proyectos a futuro.
- Embeber un procesador de 32 bits en el FPGA que contenga:
  - Registros de 32 bits
  - Stack Pointer

- Program Counter
- Set de instrucciones RISC
- Habilitar en Hardware manejo de interrupciones
  
- Contar con un módulo de reseteo en el microcontrolador; el cual reinicie el programa si fuera necesario.
  
- Contar con un módulo depurador para verificar la programación del microcontrolador y así validar el trabajo.
  
- Tener una arquitectura Harvard
  
- Contar con una memoria cache
  
- Implementar comunicaciones seriales: UART, SPI, I2C
  
- Implementar manejo de puertos GPIO

## **1.5. Diseño a bloques de la computadora de 32 bits para instrumentación.**

En el diseño de un microcontrolador, para lograr un funcionamiento óptimo es necesario analizar tanto los distintos factores del procesador (registros, memorias, unidades de cálculo, unidades de direccionamiento, módulos de decodificación) como el flujo de información (de instrucciones, datos y control) requerido durante la ejecución de las posibles instrucciones.

Hecho este análisis, deben considerarse las ventajas arquitectónicas que ofrece la tecnología a emplear, teniendo en cuenta que ciertas soluciones se acomodan mejor que otras a cierto tipo de arquitecturas.

Para el diseño del microcontrolador se busca un control eficiente por lo que se propone en su diseño de computadora una arquitectura Harvard la cual maneje un procesador de 32 bits implementado en un FPGA, esta arquitectura se caracteriza por tener dos tipos de buses de memorias una para datos y otra para el programa ofreciendo una ventaja de poder manejar en paralelo la información, nuestro diseño a implementar se observa en el siguiente diagrama de bloques el cual se muestra en la figura 1.5.1. En este caso se pretende usar un bloque funcional IP de un procesador de 32 bits en otros términos se va utilizar un Soft-Core ya sea de código abierto o cerrado.

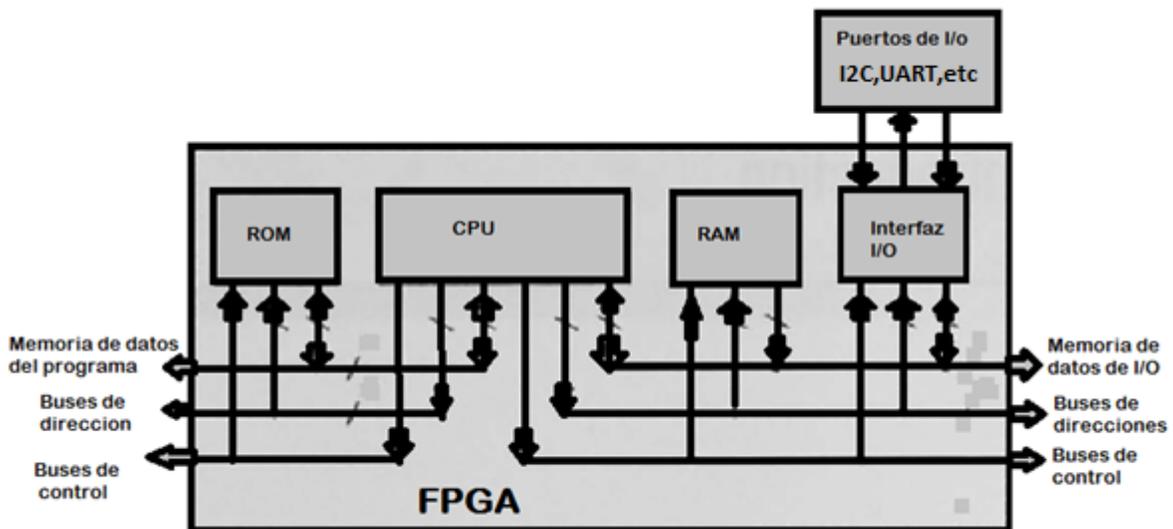


Figura 1.5.1 Diseño a bloques de la computadora de 32 bits

El diseño comprende cinco módulos principales, el módulo de memoria ROM, el módulo de memoria RAM, el módulo de CPU, el módulo de interfaz para manejo de puertos de entrada/salida y los puertos de entrada/salida

## BLOQUE CPU:

El procesador que se implementará será de tipo Soft-Core, es decir un procesador descrito en su totalidad en software que tenga como características las siguientes:

-Memoria de cache: Esta memoria se implementará debido a su rápido acceso de escritura, ya que ayuda a reducir el tiempo de acceso a datos ubicados en la memoria principal que se utilizan con mayor frecuencia.

-Interfaces para periféricos de I/O: Este interfaz será de ayuda para implementar la conexión entre el procesador y el bus de conexiones de I/O.

-Program counter: Este es un registro, con el que tendrá que contar el procesador para indicar la posición donde se encuentre en su secuencia de instrucciones.

-Registro de instrucciones: Es un registro de la unidad de control donde se almacenará la instrucción que se estará ejecutando.

-Stack Pointer: Es un registro de un CPU cuyo propósito es mantener la posición actual de una pila de información sobre subrutinas activas de un programa a realizar.

-Machine Status Register: Es un registro de control de proceso presentado en especial para procesadores de 32 y 64 bits, el cual auxilia en el control o reportes de estados de instrucciones, habilitación o inhabilitación de interrupciones.

## BUSES DE CONEXIÓN

Se implementarán buses de comunicación los cuales tendrán la característica de aprovechar los 32 bits del procesador, soportar los modos asíncronos y síncronos que se requieran, ser configurables con tamaño de información de 8, 16 y 32 bits según necesiten los puertos de comunicación de I/O.

## BLOQUES DE MEMORIA ROM Y RAM

Estas memorias de igual forma se efectuarán de manera descriptiva, las cuales tendrán una implementación eficiente buscando tener buses separados de lectura y escritura en ambas memorias para tener una mayor velocidad de acceso a los datos, y sean configurables a direcciones hasta de 32 bits

## BLOQUE DE INTERFAZ I/O

En este se implementarán las comunicaciones seriales más usadas buscando una eficiencia óptima tanto de recursos utilizados en el FPGA, así como en tiempos de adquisición y envío de datos por parte de éstas.

Una vez teniendo el diagrama a bloques de la computadora a realizar, se pretende proceder en la implementación del microcontrolador como se muestra en la figura 1.5.2, en el capítulo 3.5 y 4 se detallará el procedimiento.

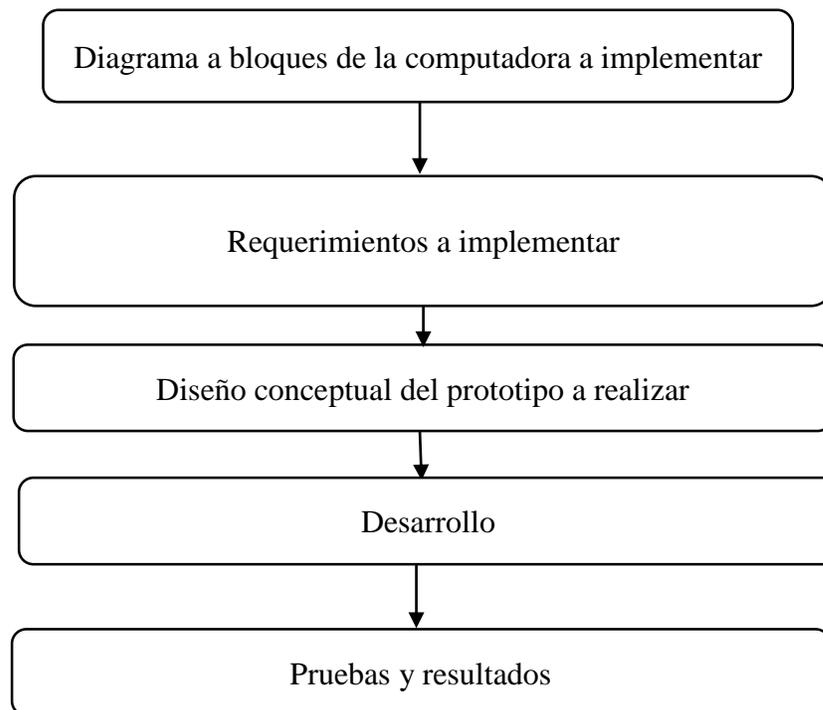


Figura 1.5.2 Pasos para la implementación del microcontrolador

## **Capítulo 2. TEORÍA BÁSICA**

### **2.1 Marco teórico para el diseño de un microcontrolador con base en un FPGA**

Una excelente opción para el diseño de un sistema embebido tipo SoC en el cual se implementará un microprocesador es el empleo de un FPGA, para esto es necesario tomar en cuenta en que consiste un microcontrolador; éste incluye en su diseño tres principales unidades funcionales; una unidad central de procesamiento, unidades de memoria y periféricos de entrada y salida, en la actualidad los microcontroladores cuentan con una mayor cantidad de recursos auxiliares como circuitos de reloj, temporizadores, watchdog, convertidores analógicos digital y digital analógico, comparadores analógicos, protección ante fallos de alimentación.

Los procesadores que pueden ser embebidos en un FPGA funcionando como unidad central de proceso, se pueden clasificar en tres familias: Hard cores, Firm cores y Soft cores.

Hard-cores son aquellos que son procesadores físicos, que no tienen gran posibilidad de configuración.

Los Firm-Cores son procesadores físicos los cuales tienen la capacidad de ser configurados en ciertos aspectos.

Los Soft-cores son procesadores que son implementados en su totalidad mediante lenguaje de descripción de hardware (HDL, por sus siglas en Inglés) de un procesador en específico.

En el diseño de sistemas embebidos se tiene la opción de usar bloques de propiedad intelectual (IP, por sus siglas en Inglés), los IPs tienen un profundo impacto ya que son descripciones funcionales de circuitos digitales, interfaces de buses seriales, funciones matemáticas, bloques de procesamiento de video, controladores de memoria, que conllevan a un rápido desarrollo en dispositivos de lógica programable.

Básicamente existen dos arquitecturas que, están presentes en el mundo de los microcontroladores: Von Neumann y Harvard. Ambas se diferencian en la forma de conexión de la memoria al procesador y en los buses que cada una necesita.

### Arquitectura Von Neumann

La arquitectura Von Neumann utiliza el mismo bloque de memoria tanto para las instrucciones como para los datos y periféricos, su ventaja de implementar solo una memoria es ahorrar una buena cantidad de líneas de E/S, que son bastante costosas, sobre todo para aquellos sistemas donde el procesador se monta en algún tipo de zócalo alojado en una placa madre. También esta organización les ahorra a los diseñadores de placas madre una buena cantidad de problemas y reduce el costo de este tipo de sistemas.

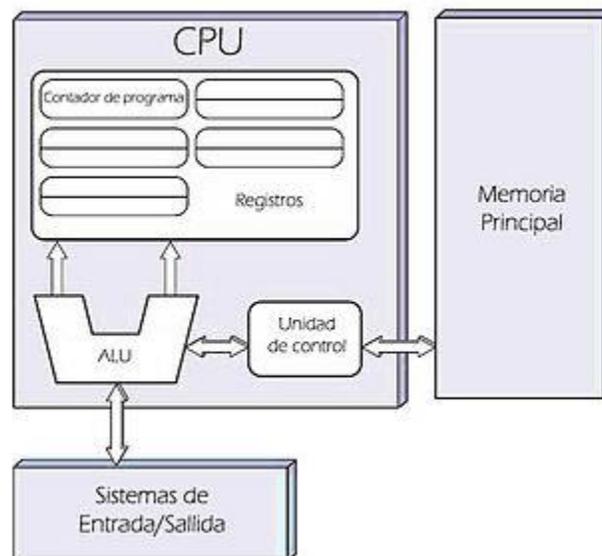


Figura 2.1.1 Arquitectura Von Neumann [9]

## Arquitectura Harvard

La otra variante es la arquitectura Harvard, y por excelencia la utilizada en supercomputadoras, en los microcontroladores, y sistemas integrados en general. En este caso, además de la memoria, el procesador tiene los buses independientes, de modo que cada tipo de memoria tiene un bus de datos, uno de direcciones y uno de control.

La ventaja fundamental de esta arquitectura es que permite adecuar el tamaño de los buses a las características de cada tipo de memoria; además, el procesador puede acceder a cada una de ellas de forma simultánea, lo que se traduce en un aumento significativo de la velocidad de procesamiento. Típicamente los sistemas con esta arquitectura pueden ser dos veces más rápidos que sistemas similares con arquitectura Von Neumann

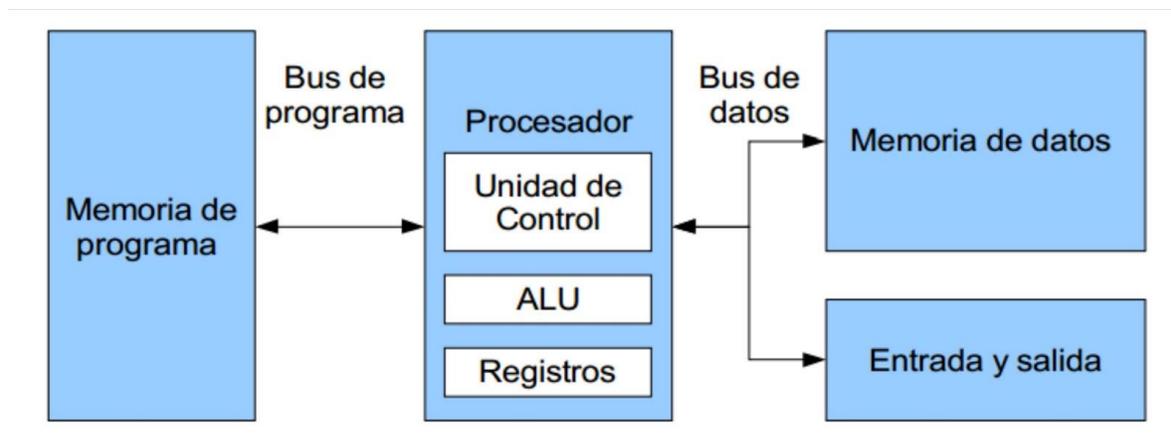


Figura 2.1.2 Arquitectura Harvard [9]

## 2.2. Dispositivos Lógicos Programables, estructura interna y clasificación

Los Dispositivos Lógicos Programables (PLD, por sus siglas en Inglés) son circuitos integrados los cuales contienen una gran cantidad de puertas lógicas dentro de un solo encapsulado y permiten al diseñador determinar cómo éstas

deben unirse. Esta tecnología se llama lógica programable debido a que las compuertas no están comprometidas con ninguna función específica hasta que el diseñador describa el circuito del que formarán parte, entendiéndose este proceso de descripción como “programación”.

Los PLD permiten una integración de aplicaciones y desarrollos lógicos mediante el empaquetamiento de soluciones en un circuito integrado. El resultado es la reducción de espacio físico dentro de la aplicación.

### **Ventajas y desventajas de los PLD**

A medida que aumenta la complejidad de un diseño, los inconvenientes de su realización con circuitos estándar se hacen muy importantes. En estos casos los circuitos programables pueden ayudar a disminuir esos inconvenientes.

Las principales ventajas que aporta el diseño en base a PLD son:

- Pueden reemplazar a varios componentes discretos, reduciendo con ello el número de circuitos integrados a utilizar. Esto a su vez permite reducción de espacio, reducción del número de conexiones, reducción de la potencia de consumo, disminución del costo.
- Aumento de la fiabilidad. Los PLDs nos brindan la posibilidad de ser reprogramados siendo una característica muy importante ya que permite utilizar el mismo dispositivo para aplicaciones distintas, sin más que modificar su programación.

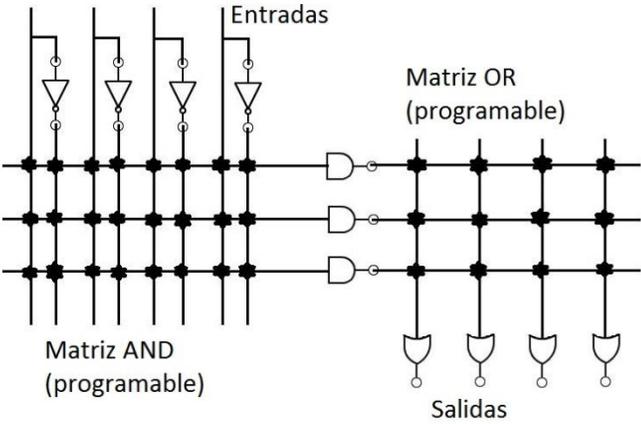
- Gran facilidad de diseño, las herramientas disponibles para este fin simplifican considerablemente el proceso de diseño, haciendo que la implementación al más bajo nivel sea transparente para el usuario.

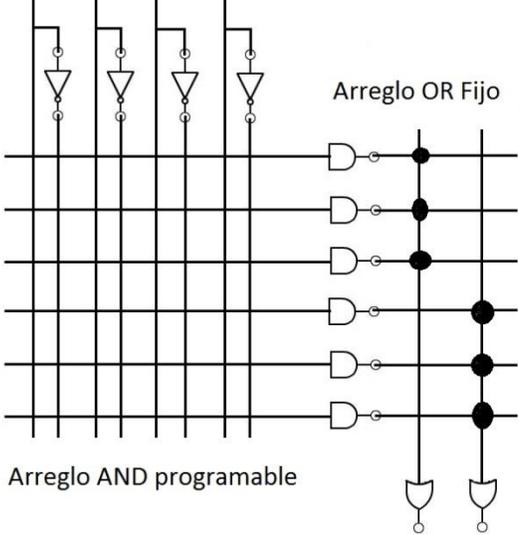
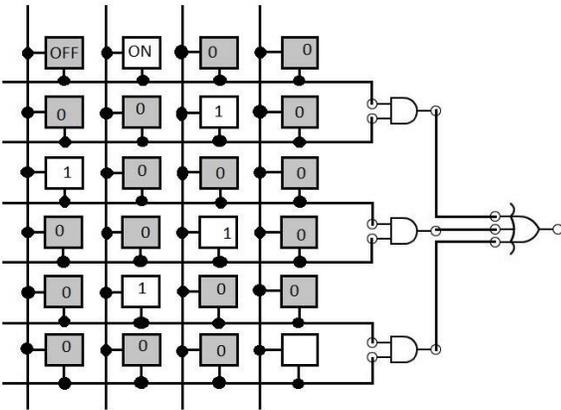
La principal desventaja que aparece al diseñar con PLD es el elevado costo de estos dispositivos frente al reducido costo de los circuitos estándar. Sin embargo, esta desventaja es compensada a medida que aumenta la complejidad del diseño, además de que cada vez se producen PLD más baratos.

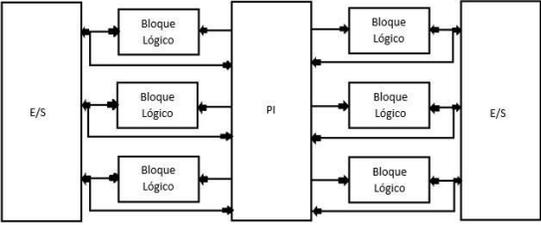
### 2.2.1. Clasificación de los PLDs

Se clasifican de acuerdo a su ordenación funcional de sus elementos internos, las cuales proporciona al dispositivo características específicas.

Los principales PLD son [10]:

Dispositivo	Características
PLA	<p>Arreglo Lógico Programable (PLA, por sus siglas en Inglés)  Implementan arreglos OR y AND programables para ejecutar una función compleja de lógica combinacional.</p>  <p>El diagrama ilustra la estructura interna de un PLA. A la izquierda, cuatro líneas verticales representan las 'Entradas'. Cada entrada tiene un punto de conexión en la 'Matriz AND (programable)', que está formada por una cuadrícula de puntos. Desde esta matriz, tres líneas horizontales pasan por puertas AND (representadas por triángulos con una 'D' en su interior) y se conectan a la 'Matriz OR (programable)'. Esta matriz OR también es una cuadrícula de puntos. Desde ella, cuatro líneas verticales descendentes pasan por puertas OR (representadas por triángulos con una 'O' en su interior) y se conectan a las 'Salidas'.</p>

<p>PAL</p>	<p>Lógica de Arreglos Programables (PAL, por sus siglas en Inglés)</p> <p>Implementa arreglos AND programables y arreglos OR fijos a la salida, las cuales fueron implementados para superar desventajas de las PAL principalmente los retardos provocados por la implementación de fusibles adicionales.</p>  <p>The diagram illustrates the internal structure of a PAL. It features a grid of horizontal lines representing AND gates and vertical lines representing OR gates. On the left, four vertical lines are connected to inverters. The grid shows connections between these lines and the AND gates. The output of the AND gates is connected to a fixed OR gate structure on the right. Labels include 'Arreglo OR Fijo' and 'Arreglo AND programable'.</p>
<p>GAL</p>	<p>Arreglo Lógico Genérico (GAL, por sus siglas en Inglés)</p> <p>Implementan arreglos AND programable y una matriz OR fija, con una salida lógica programable, la programación se realiza mediante el uso de celdas programables y contiene configuraciones de salida programables.</p>  <p>The diagram shows the internal structure of a GAL. It consists of a grid of horizontal lines representing AND gates and vertical lines representing OR gates. Each intersection in the grid contains a programmable cell, represented by a box with 'OFF' or 'ON' labels, or a box with '0' or '1' values. The output of the AND gates is connected to a fixed OR gate structure on the right. Labels include 'OFF', 'ON', '0', and '1'.</p>

CPLD	<p>Dispositivo Lógico Programable Complejo (CPLD, por sus siglas en Inglés)</p> <p>Son Implementadas con múltiples bloques lógicos, cada uno similar a un PLD básico, estos bloques lógicos se comunican entre sí utilizando una matriz programable de interconexiones (PI) la cual se encarga de interconectar los bloques lógicos y los bloques de entrada / salida.</p> 
FPGA	<p>Campos de Arreglos de Compuertas Programables (FPGA, por sus siglas en Inglés)</p> <p>Son campos de arreglos de compuertas programables, estos dispositivos lógicos programables constituyen el grupo actual con más alta densidad de integración, superando ampliamente el millón de compuertas lógicas</p>

### 2.3. FPGA de alto nivel de integración y lenguaje HDL

La principal característica de los PLD de alto nivel de integración es implementar la mayor cantidad de dispositivos en un circuito, se caracterizan por la reducción de espacio y costo, además de ofrecer una mejora sustancial en el diseño de sistemas complejos, con incremento de velocidad y mayores frecuencias de operación.

## 2.3.1 FPGA

La tecnología arreglos de campos de compuertas programables (FPGA, por sus siglas en Inglés) son circuitos integrados de silicio reprogramables, en los cuales se pueden implementar funciones personalizadas en hardware y forman parte del grupo de PLD de alto nivel de integración.

Los FPGAs están hechos de un número limitado de recursos ya previamente definidos con interconexiones programables para implementar un circuito digital reconfigurable y bloques de entrada y salida, vea figura 2.3.1.1.

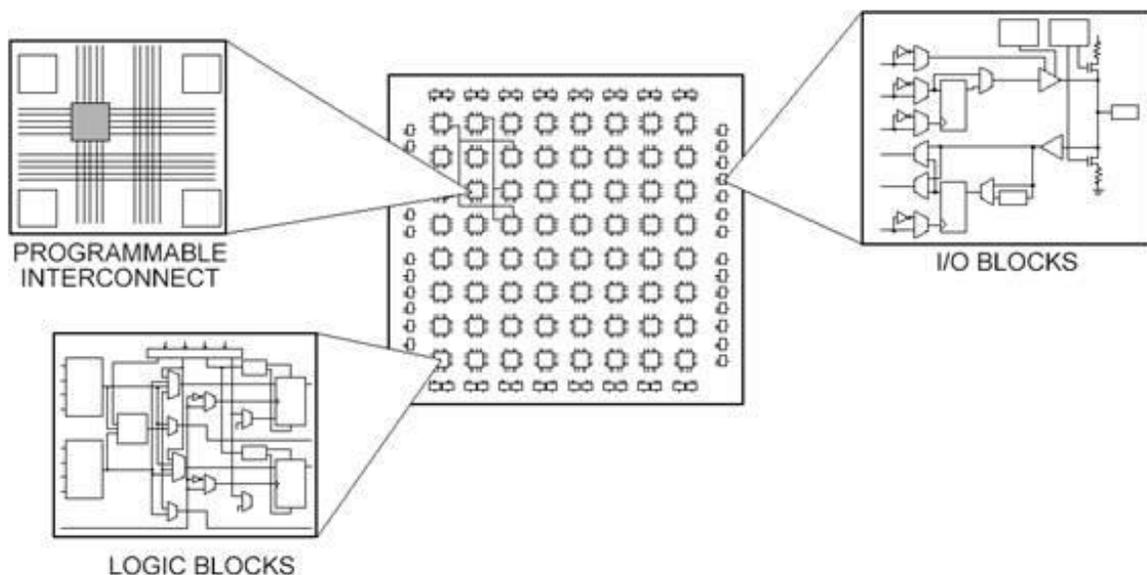


Figura 2.3.1.1. Diferentes partes de un FPGA, arquitectura de la familia XC4000 de Xilinx [11]

Los FPGAs consisten en 3 elementos importantes, los cuales son los bloques de lógica configurable (CLBs), bloques de entrada y salida y canales de comunicación.

Los bloques de lógica configurable (CLBs) son la unidad de lógica básica de un FPGA, estos bloques están hechos de dos componentes básicos los cuales son

flip-flops y tablas de consulta (LUTs) fundamentales para la síntesis de funciones lógicas, vea figura 2.3.1.2.

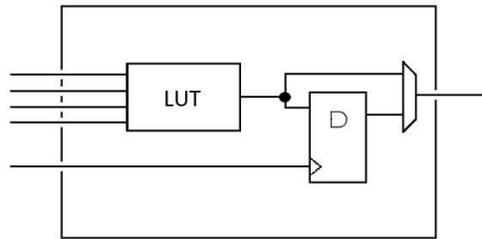


Figura 2.3.1.2 Constitución de CLB

Las LUTs contienen celdas de almacenamiento que sirven para implementar una pequeña función lógica, éstas son implementadas con multiplexores debido a que pueden sintetizar funciones lógicas. Esto se logra mediante un teorema propuesto por Claude Shannon” [12] [13].

En la salida de cada LUT son conectados los flip-flops los cuales se usan para almacenar el valor de su entrada bajo el control de un reloj, vea figura 2.3.1.3.

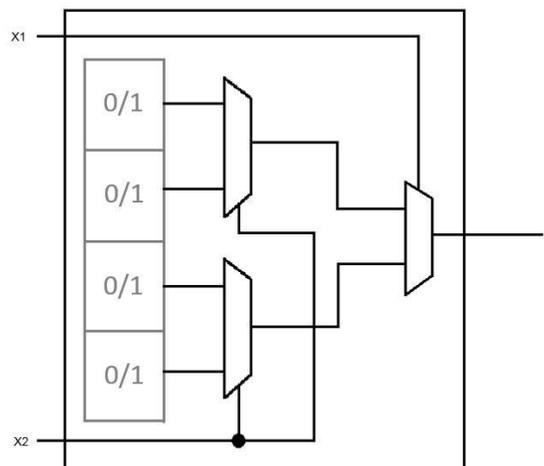


Figura 2.3.1.3 Implementación de LUTs

Los CLB están ordenados en arreglos de matrices programables, la matriz se encarga de dirigir las salidas de un bloque a otro.

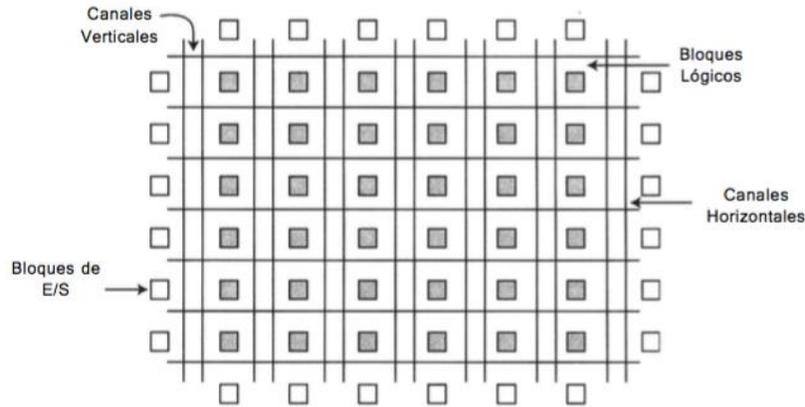


Figura 2.3.1.4 Matrices de bloques Lógicos [11]

Las principales ventajas que ofrecen los FPGAs son:

**Rendimiento:** Se aprovecha el paralelismo del hardware es decir los FPGAs a diferencia de los microprocesadores y DSPs no siguen una ejecución secuencial, por lo que logran una mayor cantidad de procesos en cada ciclo de reloj. [14]

**Tiempo en llegar al mercado:** La tecnología que ofrecen los FPGAs presentan flexibilidad y capacidades para un rápido desarrollo de prototipos.

**Mantenimiento a largo plazo:** Los circuitos de FPGA son reprogramables por lo que se puede dar seguimiento y actualizaciones al diseño funcional sin modificar el hardware.

Las especificaciones más importantes al seleccionar un FPGA para una aplicación en particular son el número de bloques de lógica configurable, número de bloques de lógica de función fijos como multiplicadores y el tamaño de los recursos de memoria como RAM en bloques embebidos.

## **2.3.2 HDL**

Un lenguaje de descripción de hardware (HDL, hardware description language) sirve para describir de manera específica la estructura y comportamiento de un circuito lógico digital.

Los HDL más usados son los que están respaldados como norma por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE, por sus siglas en Inglés), los lenguajes estandarizados más usados son el VHDL (lenguaje de descripción de hardware de circuitos de muy alta velocidad) y el Verilog [15] [16].

Debido a la estandarización de estos lenguajes se tiene una independencia en la tecnología es decir que el lenguaje puede ser programado en hardware de diferentes vendedores, esto significa que puede implementarse en diferentes tipos de chips y herramientas de diseño sin cambiar la especificación en HDL

## **2.4 Bloques de Propiedad Intelectual y SOFT CORE**

En el diseño electrónico para la implementación de funciones y sistemas en PLDs se tiene la opción de usar bloques de propiedad intelectual (IP, por sus siglas en Inglés) los cuales son bloques reusables que son patentados, los IPs tienen un profundo impacto en los sistemas en un Circuito Integrado (SoC, por sus siglas en Inglés) ya que son descripciones funcionales de diseños de circuitos electrónicos que conllevan a un rápido desarrollo en dispositivos de electrónica lógica programable.

## 2.4.1 SOFT CORE

Un Soft Core es un núcleo de microprocesador que es implementado en su totalidad mediante lenguaje de hardware descriptivo (HDL, por sus siglas en inglés) específico. Se implementa en diferentes dispositivos semiconductores que contienen lógica programable (por ejemplo, FPGA, CPLD). [17]

EL uso de Soft Cores tiene muchas ventajas para el diseño de sistemas embebidos, ya que pueden ser personalizados para aplicaciones en específico con una relativa sencillez debido a que solo se modifican partes específicas y no se realiza el diseño desde cero. [18]

La mayoría de los sistemas, utilizan solo un Soft Core, sin embargo, algunos diseñadores usan más de uno en un FPGA. [19]. Aunque muchas personas ponen exactamente un procesador en un FPGA, la capacidad de este es suficientemente grande por lo que puede contener dos o más procesadores, lo que resulta en el diseño de un sistema multi-núcleo, estos sistemas son diseñados para poder alcanzar un mayor procesamiento en paralelo. El número de procesadores en un solo FPGA sólo está limitado por el tamaño de la FPGA. [20] Algunas personas han puesto decenas o cientos de Soft Cores en un solo FPGA. [21] [22] [23] [24] [25]

Estos Soft Cores pueden ser adquiridos a través de los fabricantes de FPGAs o comunidades de código abierto, los principales Soft Core son Nios II, MicroBlaze, PicoBlaze y Xtensa desarrollados por Altera, Xilinx y Tensilica respectivamente. Estos pueden ser implementados con relativa facilidad usando sus entornos de desarrollo respectivamente, los entornos facilitaran su implementación, configuraciones, módulos a programar, puertos de entrada-salida y buses de comunicación. Para la adquisición de estos Soft Cores se tiene que obtener su licencia debido a que son bloques de IP, vea tabla 2.4.1.1.

Tabla 2.4.1.1 Soft-Cores

Procesador	Dsarrollador	Codigo Abierto	Lenguaje de descripcion
ZPUino	Álvaro Lopes	Si	VHDL
ZPU	Zylin AS	Si	VHDL
Zet	Zeus Gómez Marmolejo	Si	Verilog
YASEP	Yann Guidon	Si AGPLv3	VHDL
PicoBlaze	Xilinx	Si	VHDL, Verilog
MicroBlaze	Xilinx	No	VHDL, Verilog
OpenFire	Virginia Tech CCM Lab	SI	Verilog
RISC-V	UC Berkeley	Si	Chisel
Navré	Sébastien Bourdeauducq	Si	Verilog
OpenSPARC T1	Sun	Si	Verilog
AEMB	Shawn Tan	Si	Verilog
PacoBlaze	Pablo Bleyer	Si	Verilog
OpenRISC	OpenCores	Si	Verilog
SYNPIC12	Miguel Angel Ajo Pelayo	Si MIT	VHDL
MCL86	MicroCore Labs	No	
JOP	Martin Schoeberl	Si	VHDL
SecretBlaze	LIRMM, University of Montpellier / CNRS	Si	VHDL
LatticeMico32	Lattice	Si	Verilog
xr16	Jan Gray	No	Schematic
CPU86	HT-Lab	Si	VHDL
LEON2(-FT)	ESA	Si	VHDL
ERIC5	Entner Electronics	No	VHDL
eSi-RISC	EnSilica	No	Verilog
pAVR	Doru Cuturela	Si	VHDL
Cortex-M1	ARM	No	Verilog

ARC	ARC International, Synopsys	No	Verilog
TSK3000A	Altium	No Royalty- Free	
TSK51/52	Altium	No Royalty- Free	
Nios, Nios II	Altera	No	Verilog
LEON3/4	Aeroflex Gaisler	Si	VHDL

## **2.5 BUSES DE COMUNICACIÓN: BUS I2C, Hardware I2C, Protocolo de comunicación I2C, Comunicación SPI y UART**

### **2.5.1 BUS I2C**

El Bus Entre-Circuitos Integrados (IIC o I2C, por sus siglas en inglés) es un protocolo destinado a permitir múltiples circuitos integrados digitales "esclavos" para comunicarse con uno o más circuitos integrados "maestro".

La velocidad de comunicación inicial se definió con un máximo de 100 kbit por segundo, actualmente se puede tener velocidades de 3.4 Mbit por segundo.

### **2.5.2 Hardware del I2C.**

El bus de I2C consiste en dos señales: SCLK y SDA, el SCLK es el reloj de la señal, y el SDA es la información, la señal de reloj se genera siempre por el maestro; algunos dispositivos esclavos pueden forzar a bajar el reloj para retrasar el envío de más datos del maestro al esclavo.

Las salidas de los buses de comunicación I2C son de "open drain" lo que significa que ellos "tiran" la línea de la señal a un valor lógico bajo, por lo tanto, cada línea de la señal tiene que tener resistencias de pull-up para tener un valor lógico alto y distinguir las señales de I2C, vea figura 2.5.2.1.

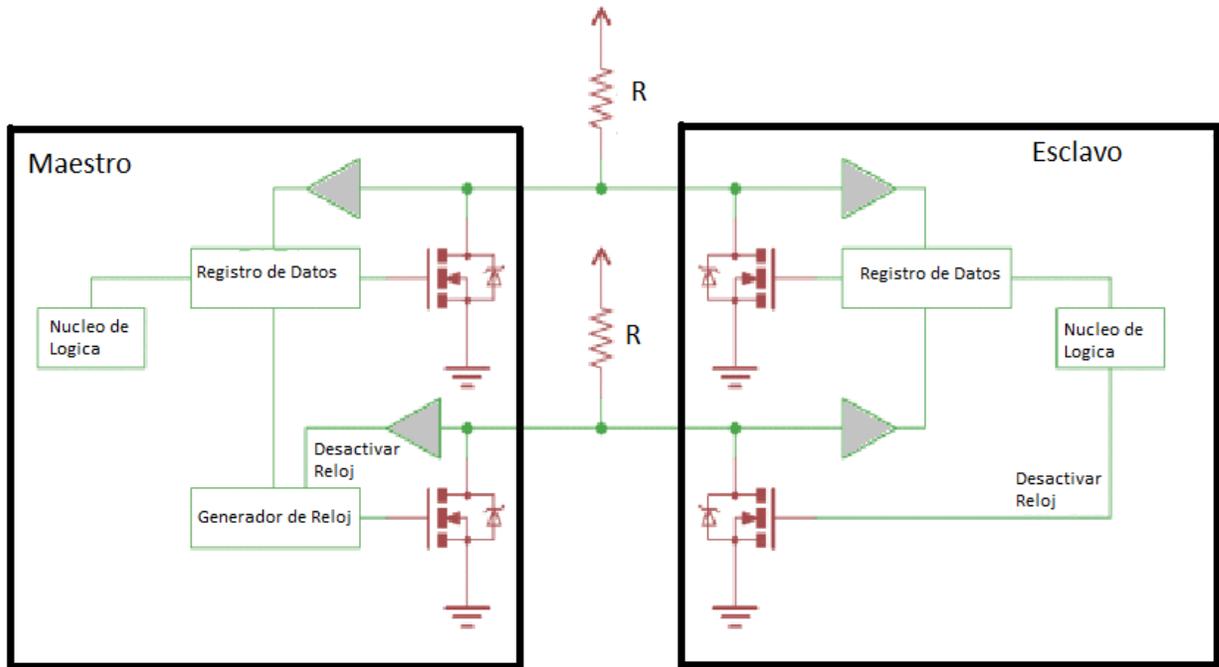


Figura 2.5.2.1 Conexión en hardware del protocolo I2C

La selección de las resistencias de “pull-up” es una consideración importante ya que con ellas se limita la corriente que es capaz de drenar el transistor FET que controla los flancos de subida y bajada de la línea de datos en el bus.

Para determinar la resistencia mínima de “pull-up” se calcula:

$$R_{p(min)} = \frac{V_{CC} - V_{OL(max)}}{I_{OL}}$$

Donde:

$V_{CC}$ : es la alimentación de la fuente

$V_{OL(max)}$  : Es el voltaje máximo, que puede ser leído por el dispositivo como un valor bajo

$I_{OL}$  : Es la corriente de drenado del FET en [mA] especificada por el fabricante

Para el cálculo de la resistencia de “pull up” máxima se tiene que está limitada por la capacitancia de bus y del tiempo de subida de las señales SDA y SCLK donde [26]:

$$Rp(max) = \frac{tr}{0.8473 \times C_b}$$

$C_b$ ; Capacitancia del bus

$tr$ ;Tiempo máximo de subida

Los valores  $C_b$  y  $tr$  son proporcionados por el fabricante de los dispositivos con interfaz I2C, por lo que el valor de las resistencias varían conforme al dispositivo que se emplea.

### 2.5.3 Protocolo de comunicación I2C

El protocolo de comunicación I2C comienza con una condición de inicio, la cual consiste en un flanco de bajada en la línea de SDA cuando la línea SCLK este en un valor alto y es seguida por el envío de la dirección del esclavo de 8 bits, posteriormente la dirección de registro de 16 bits a escribir, la cual se da por su dirección alta en 8 bits seguida de su dirección baja en 8 bits, y por último se envían las tramas de 8 bits para escritura o lectura, al final un bit de stop el cual se da poniendo en un valor alto la señal del SDA para terminar la comunicación, vea la figura 2.5.3.1, en la cual se observa el protocolo de comunicación.

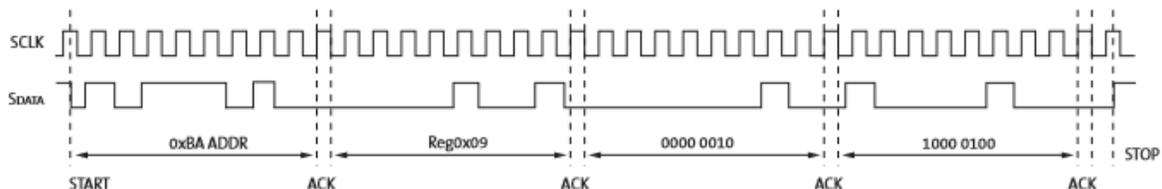


Figura 2.5.3.1 Protocolo de comunicación I2C [26]

## 2.5.4 COMUNICACIÓN SPI

La Interfaz Periférica Serial (SPI, por sus siglas en inglés) es un bus usado para el control de dispositivos periféricos y tiene ventajas sobre el I2C, debido a su simplicidad y generalidad, además de una mayor velocidad de transferencia de datos.

El método de comunicación es ilustrado en la figura 2.5.4.1, el maestro y el esclavo están atados con cuatro señales, reloj serial (SCLK, por sus siglas en inglés), salida de datos del Esclavo y entrada al Maestro (MISO, por sus siglas en inglés), salida de datos del Maestro y entrada de datos al Esclavo (MOSI, por sus siglas en inglés) y Selector de Esclavo (SS, por sus siglas en inglés).

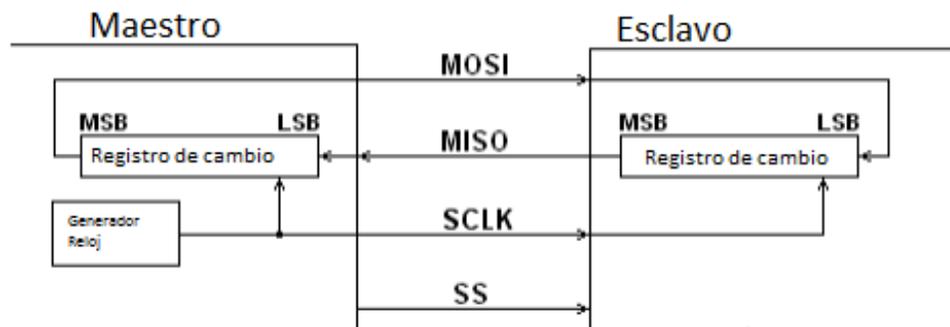


Figura 2.5.4.1 Conexión en hardware del protocolo SPI

En la comunicación SPI la transferencia de los datos son sincronizadas por la línea de reloj, estos tienen 2 bits de configuración llamados Polaridad de Reloj (CPOL) y Fase de Reloj (CPHA), donde CPOL=0 indica flanco de bajada y CPOL=1 indica flanco de subida, CPHA nos determina en que transición se envía la información, si CPHA=0 esta envía la información en cada transición del reloj de bajo a alto, si CPHA=1 esta envía la información en cada transición de alto a bajo.

En la Figura 2.5.4.2 se ilustra cómo funciona la comunicación SPI, dada para una polarización de CPOL=0,1 y CPHA=0,1.

El protocolo de comunicación del SPI se describe principalmente por el comienzo de la activación del dispositivo el cual se da cuando en la línea del SS se pone en un nivel lógico bajo, una vez hecho se empieza la comunicación de los buses MISO y MOSI de manera paralela los cuales envían las tramas de datos de 8 bits al dispositivo maestro y esclavo, las comunicaciones en ambos buses se ejecutan al mismo tiempo, a este tipo de comunicación serial se le llama full dúplex.

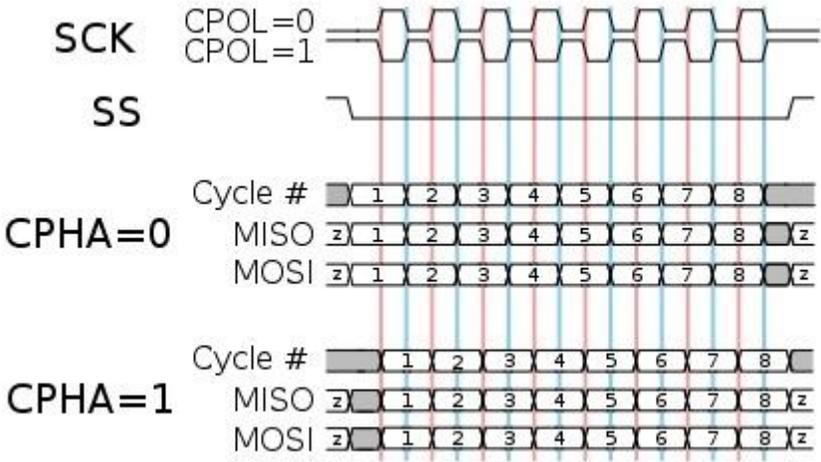


Figura 2.5.4.3 Protocolo de comunicación SPI

Gracias a la línea de selección SS se puede trabajar con múltiples dispositivos esclavos controlados por un dispositivo maestro.

Cuando hay más de un SPI esclavo estos se conectan en paralelo, como se muestra en la Figura 2.5.4.4.

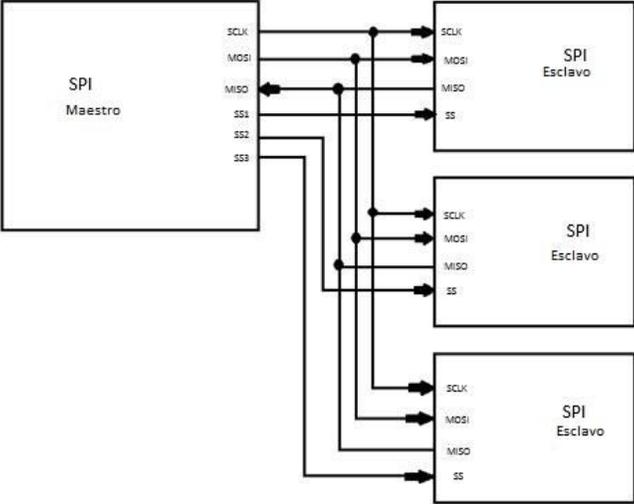


Figura 2.5.4.4 Conexión de múltiples esclavos SPI

Los dispositivos conectados al bus son definidos como maestros y esclavos, un dispositivo configurado como maestro es aquel que inicia la transferencia de información sobre el bus y genera las señales de reloj y selección, un dispositivo esclavo es controlado por el maestro por medio de la línea selectora SS por un nivel lógico bajo.

## 2.5.5 UART

El periférico transmisor-receptor asíncrono universal (UART, por sus siglas en ingles), es un protocolo de comunicación half-duplex, el UART es un componente clave del subsistema de comunicaciones serie de una computadora, el UART toma bytes de datos y transmite los bits individuales de forma secuencial

Para su funcionamiento a nivel de hardware tiene que tener una línea de transmisión 'TX', recepción 'Rx' y una de referencia la cual es la de tierra 'GND', para la comunicación entre dos dispositivos UART, las conexiones deben realizarse cruzadas para que uno pueda recibir información mientras el otro envía información adecuadamente y viceversa esto se muestra en la figura 2.5.5.1.

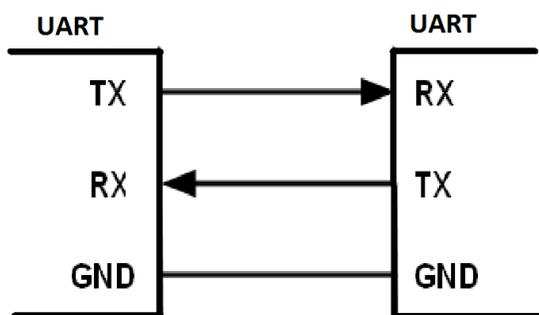


Figura 2.5.5.1 Hardware en UART

Descripción del protocolo.

EL protocolo UART para transmitir se da estableciendo un bit de inicio posteriormente se envía la información de 5 a 8 bits según se requiera, posteriormente se puede establecer opcionalmente un bit de paridad y para finalizar se pone un bit de STOP.

Para recibir información por UART se recibe el bit de inicio posteriormente la trama de 5 a 8 bits dependiendo la selección, se lee el bit de paridad si se seleccionó previamente en la transmisión y por último el bit de STOP [27], vea figura 2.5.5.2.

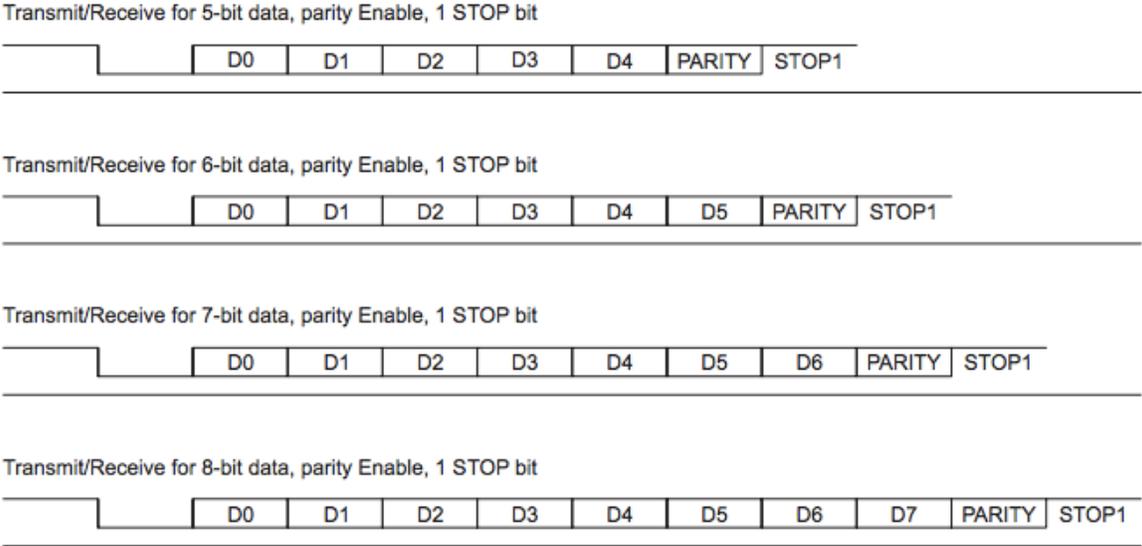


Figura 2.5.5.2 Descripción en bits del protocolo UART [27]

En las telecomunicaciones y la electrónica los baudios son la unidad de velocidad de símbolos o la tasa de modulación de símbolos por segundo. Es el número de cambios de símbolos distintos realizados en el medio de transmisión por segundo en una señal modulada digitalmente o un código de línea.

El recuento de baudios incluye los bits de información, inicio, STOP y de paridad que se generan por la UART enviar y removidos por el UART receptora.

La velocidad estándar de baud rate más utilizada se muestran en la tabla 2.5.5.1:

Tabla 2.5.5.1 Baud Rate comunicación UART

<b>Baud Rate</b>
110
300
1200
2400
4800
9600
19200
38400
57600
115200
128000
230400
460800
921600

Cada símbolo puede codificar uno o varios dígitos binarios o "bits". El tiempo de duración de cada símbolo "Ts" se puede calcular como:

$$T_s = 1/f_s$$

Donde  $f_s$  es la velocidad de símbolo.

Como ejemplo; se tiene una velocidad en baudios de 1 kBd = 1.000 Bd es decir, una tasa de símbolos de 1.000 símbolos por segundo. En caso de un módem, esto corresponde a 1.000 tonos por segundo, y en el caso de un código de línea, esto corresponde a 1.000 impulsos por segundo. El tiempo de duración del símbolo es  $1 / 1.000$  segundos = 1 milisegundo.

## **2.6 Transceptores para módulos de telemetría**

La telemetría es una tecnología que permite la medición remota de las características y magnitudes físicas de un objeto y/o sistema para su posterior envío a una estación donde son desplegados, guardados y analizados.

La telemetría se usa en grandes sistemas, como lo pueden ser satélites, misiles, sistemas de inteligencia militar, plantas químicas, redes de suministro eléctrico, redes de suministro de gas, monitoreo de energías, equipos médicos, comunicaciones, debido a que facilitan la monitorización, registro de información, así como el envío de señales para su control.

### **2.6.1 Transceptores**

Un transceptor es un dispositivo que permite enviar y recibir información de una señal analógica o digital, por lo que tienen implementado en el mismo circuito un transmisor y receptor

#### Transmisor

La función del transmisor es tomar la señal de información que se envía, y la convierte en una señal de RF que puede transmitirse a través de grandes distancias. Todo transmisor tiene tres funciones básicas. Primera, debe generar una señal de la frecuencia correcta en un punto deseado del espectro. Segunda, debe proporcionar cierta forma de modulación para que la señal de información modifique la señal de la portadora. Tercera, debe efectuar la amplificación de potencia suficiente para asegurar que el nivel de la señal sea lo bastante alto para que recorra eficazmente la distancia deseada.

## Receptor

La función del receptor es ser capaz de aceptar y demodular una señal de radio frecuencia emitida por un transmisor a una frecuencia determinada, a fin de obtener la información. El receptor realiza un proceso inverso al del transmisor ya que descifra e interpreta la información recibida por el transmisor. La señal de entrada al receptor generalmente presenta una amplitud extremadamente baja, un receptor típico debe ser capaz de amplificar la señal de entrada, para que esta tenga suficiente amplitud para ser útil.

Las interfaces más usadas en transceptores comerciales se muestran en la tabla 2.6.1.1 y figura 2.6.1.1.

Tabla 2.6.1.1 Interfaces de comunicación en transceptores

<b><u>Interfaces:</u></b>
I <sup>2</sup> C, UART, CAN-Bus
UART
CAN-Bus
SSMCX antenna
RF SMA or MCX

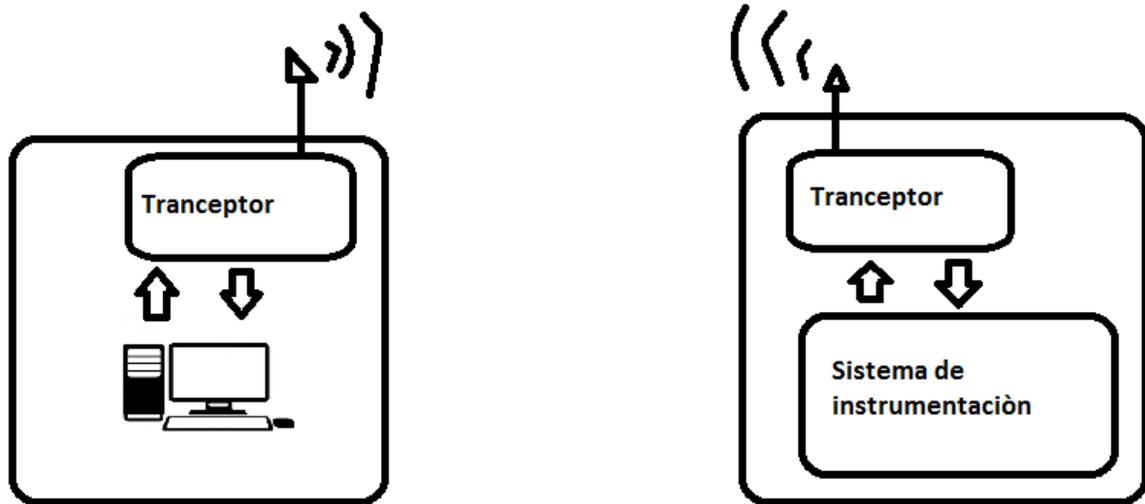


Figura 2.6.1.1 Sistema de trancceptores

## Capítulo 3. ANALISIS Y DISEÑO DE LA PROPUESTA A IMPLEMENTAR

### Metodología de diseño

La metodología de diseño comprende los siguientes pasos:

1. Determinación de requerimientos del microcontrolador y definición de su arquitectura;
2. Selección de un FPGA en el que se implementara un microcontrolador en lenguaje VHDL;
3. Selección de componentes;
4. Diseño e implementación del hardware y software del microcontrolador

### 3.1 Requerimientos Generales del microcontrolador

Un requerimiento, es una exigencia de un usuario que necesita, para la solución de un problema o el alcance de un objetivo. Como se revisó en el estado del arte actualmente la tendencia de microcontroladores está migrando al uso de FPGAs con procesadores embebidos, por lo que al diseñar e implementar un microcontrolador en FPGA, se tienen ventajas en comparación a un microcontrolador de carácter comercial debido a su mayor flexibilidad para migración y adaptabilidad de aplicaciones, tomando en cuenta las características de los microcontroladores comerciales se establecen requerimientos para su diseño en FPGA, los cuales son:

- Elegir un procesador de 32 bits, de código abierto o licenciado en el FPGA.
- Tener instrucciones tipo RISC
- Establecer una arquitectura Harvard
- Elegir módulos de comunicación serial: I2C, SPI, UART; calificados por compañías internacionales o universidades
- Usar un módulo de comunicación paralelo: GPIO
- Que el procesador funcione a 100 MHz
- Seleccionar un módulo que provea una frecuencia de 100 MHz hasta 500 MHz, para electrónica externa.
- Elegir un dispositivo FPGA que se tenga un fácil acceso a documentación.
- Tener un diseño en HLS, para minimizar costos
- Que se pueda desarrollar con herramientas comerciales
- Que se use un Soft-Core para agilizar su desarrollo, calificado por compañías internacionales o universidades
- Que tenga suficiente memoria RAM para su uso y aplicación
- Que se pueda configurar sus buses de comunicación en base a necesidades del diseño
- Que los módulos IP sean o hayan sido probados por otros usuarios en universidades

- Que posean la documentación necesaria para su uso y configuración
- Que puedan ser evaluados por software
- Que se incluyan en algún hardware comercial de preferencia Xilinx
- Que consuman muy baja potencia
- Que se polarice con 3 V de preferencia
- Se tenga soporte para módulos IP para reloj, comunicación I2C, UART, GPIO y SPI

## **3.2 Diseño conceptual del prototipo a realizar**

Para poder implementar esta arquitectura planteada en el punto 1.5 se requiere de un procesador de tipo SoftCore que contenga registros, una unidad aritmética lógica (ALU), una memoria rom para guardar nuestro programa, una memoria cache que facilite el manejo de datos e información, y los principales módulos de comunicación descritos anteriormente, para esto debe contar con un bus de instrucciones capaz de manejar los bits de información de cada módulo de comunicación por lo mínimo de 16 bits, las características del microcontrolador se presentan en la tabla 3.2.1, en la figura 3.2.1 se presenta nuestro diseño del microcontrolador a realizar en el FPGA.

Para estimar el volumen de recursos que se requerirán del FPGA se realizó una estimación preliminar de los principales recursos lógicos del FPGA por medio de la síntesis de la descripción de estos en un entorno de desarrollo VHDL, los resultados se muestran en la Tabla 3.2.1. Los detalles del diseño de los bloques implementados en el FPGA se presentarán en el punto 4.5.

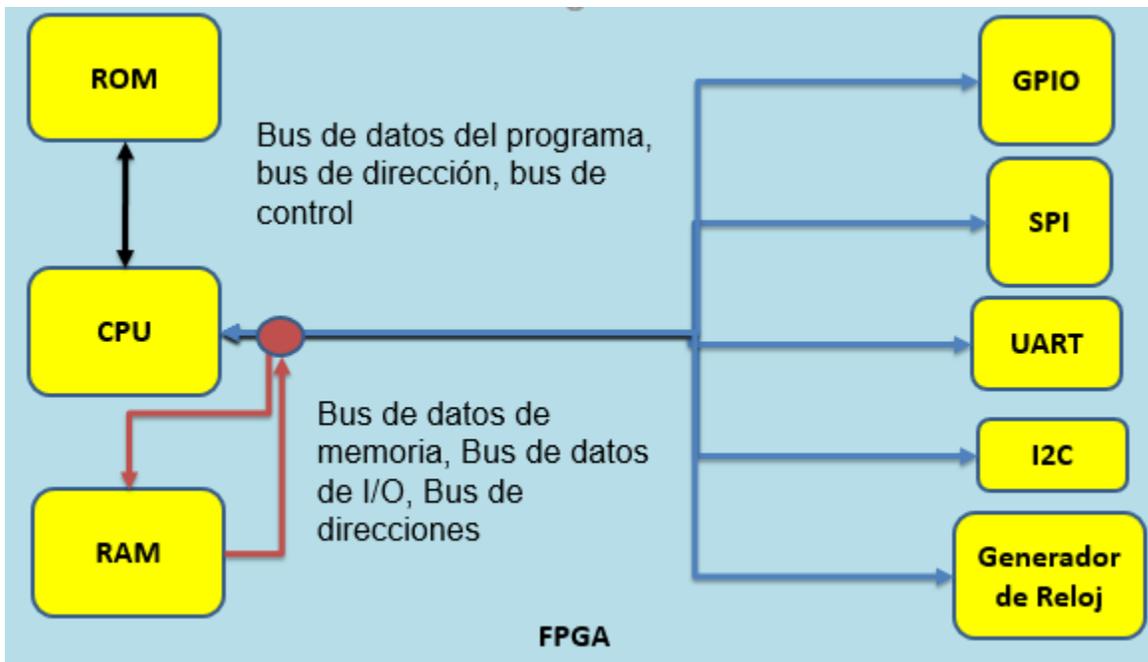


Figura 3.2.1 Diagrama General del microcontrolador

Tabla 3.2.1 Requerimientos generales del microcontrolador

Bloques	Descripción	Número de Puertos Requeridos	Estimación de LUTS requeridas
I2C	Este protocolo de comunicación se usará para la configuración de sensores.	2	415
SPI	El protocolo SPI se usará para configuración de sensores que lo requieran	4	342
GPIO	Estos puertos serán necesarios para obtener las lecturas y escrituras digitales que se puedan necesitar.	18	261

Generador de Reloj	Se implementará un módulo de generación de reloj para propósito general	1	64
UART	El protocolo UART será necesario para poder implementar telemetría y recibir y transmitir órdenes	4	326
Soft Core	Para la implementación del microcontrolador, el cual configurará y gestionará el funcionamiento de los módulos de comunicación a implementar, se eligió el uso de un soft core con el fin de alcanzar el menor tiempo de diseño	4	3,788

### 3.3 Elección de FPGA

Para la elección del FPGA se escogió trabajar con los FPGA de Xilinx ya que se encontró que:

"Científicos de la NASA encontraron que los FPGAs de Xilinx son más adecuados para tareas de alto desempeño debido a su flexibilidad y sus bloques de DSP embebidos comparado con las computadoras de abordo individuales y DSPs. Aparte del incremento en rendimiento, los FPGAs basados en SRAM ofrecen la capacidad de ser reconfigurado." [28], la anterior afirmación es uno de los principales fundamentos por la que se seleccionó los FPGA de Xilinx para el desarrollo del microcontrolador de nuestro proyecto. Muchas universidades recomiendan el uso de este soft-core como núcleo de muchas aplicaciones

industriales, por su gran confiabilidad, desempeño y bajo consumo de energía, además del respaldo que ofrece xilinx a los productos que desarrolla.

Para la elección del FPGA a utilizar se tiene que considerar su capacidad computacional, consumo de potencia, y suficiente capacidad lógica necesaria para los procesos que ejecutará, como se revisó en la tabla 2.4.1.1 se optó por usar el soft-core MicroBlaze que es propio de Xilinx, este es de código cerrado y no podrá ser modificado una vez hecho esto, se realizó una estimación de recursos necesarios, con base a los requerimientos generales, véase la tabla 3.3.1.

Tabla 3.3.1 Estimación en recursos lógicos preliminares del microcontrolador

Recursos	Estimación
LUTs	4,870
Flip-Flops	9,000
BRAM	50

Ya con los datos más relevantes para la implementación en FPGA, se investigaron las familias de Xilinx las cuales tienen las siguientes características, vea las tablas 3.3.2 – 3.3.7.

Tabla 3.3.2 Spartan 6 FPGAs

Spartan-6 LX FPGAs									Spartan-6 LXT FPGAs				
Optimizados para lógica de más bajo costo, DSP y Memoria (1.2V, 1.0V)									Optimizado para lógica de más bajo costo,				
Part	XC6SL	XC6SL	XC6SL	XC6SL	XC6SL	XC6SL	XC6SLX	XC6SLX	XC6SLX	XC6SLX	XC6SLX	XC6SLX1	XC6SLX1
Slices(	600	1,430	2,278	3,758	6,822	11,662	15,822	23,038	3,758	6,822	11,662	15,822	23,038
Celdas	3,840	9,152	14,579	24,051	43,661	74,637	101,261	147,443	24,051	43,661	74,637	101,261	147,443
CLB	4,800	11,440	18,224	30,064	54,576	93,296	126,576	184,304	30,064	54,576	93,296	126,576	184,304
Block	12	32	32	52	116	172	268	268	52	116	172	268	268
DSP48	8	16	32	38	58	132	180	180	38	58	132	180	180

Tabla 3.3.3 Artix 7 FPGAs

Artix®-7 FPGAs						
Optimizados para aplicaciones de bajo costo y consumo de potencia (1.0V, 0.95V, 0.9V)						
Numero de FPGA	XC7A15T	XC7A35T	XC7A50T	XC7A75T	XC7A100T	XC7A200T
Celdas Logicas	16,640	33,280	52,160	75,520	101,440	215,360
LUTs	10,400	20,800	32,600	47,200	63,400	134,600
CLB Flip-Flops	20,800	41,600	65,200	94,400	126,800	269,200
Block RAM/FIFO w/ ECC (36 Kb each)	25	50	75	105	135	365
DSP Slices	45	90	120	180	240	740

Tabla 3.3.4 Kintex 7 FPGAs

Kintex®-7 FPGAs		Optimizados para la relación Rendimiento-Precio (1.0V, 0.95V, 0.9V)						
Numero de FPGA		XC7K70T	XC7K160T	XC7K325T	XC7K355T	XC7K410T	XC7K420T	XC7K480T
EasyPath™ Cost Reduction		—	—	XCE7K325T	XCE7K355T	XCE7K410T	XCE7K420T	XCE7K480T
LUTs		41,000	101,400	203,800	222,600	254,200	260,600	298,600
Celdas Lógicas		65,600	162,240	326,080	356,160	406,720	416,960	477,760
CLB Flip-Flops		82,000	202,800	407,600	445,200	508,400	521,200	597,200
Block RAM/FIFO w/ ECC (36		135	325	445	715	795	835	955
DSP48 Slices		240	600	840	1,440	1,540	1,680	1,920

Tabla 3.3.5 Virtex FPGAs

Virtex 7 FPGAs		Optimizados para el más alto rendimiento y capacidad del sistema (1.0V)									
Numero de FPGA	XC7V585T	XC7V2000T	XC7VX330T	XC7VX415T	XC7VX485T	XC7VX550T	XC7VX690T	XC7VH580T	XC7VH70T		
EasyPath™	XCE7V5	—	XCE7VX33	XCE7VX4	XCE7VX4	XCE7VX5	XCE7VX6	XCE7VX9	—	—	—
LUTs	364,200	1,221,600	204,000	257,600	303,600	346,400	433,200	612,000	712,00	362,80	547,600
Celdas	582,720	1,954,560	326,400	412,160	485,760	554,240	693,120	979,200	1,139,200	580,48	876,160
CLB Flip-Flop	728,400	2,443,200	408,000	515,200	607,200	692,800	866,400	1,224,000	1,424,000	725,60	1,095,20
Block	795	1,292	750	880	1,030	1,180	1,470	1,500	1,880	940	1,410
DSP Slices	1,260	2,160	1,120	2,160	2,800	2,880	3,600	3,600	3,360	1,680	2,520

Tabla 3.3.6 Kintex® UltraScale+™ FPGAs

Numero de FPGA	KU3P	KU5P	KU9P	KU11P	KU13P	KU15P
CLB Flip-Flop (K)	325	434	548	597	683	1,045
LUTs (k)	163	217	274	299	341	523
Total Block Ram	12.7	16.9	32.1	21.1	26.2	34.6
DSP Slices PCIE	1,368	1,824	2,520	2,928	3,528	1,968

Tabla 3.6.7 Virtex® UltraScale+™ FPGAs

Nombre de FPGA	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P
Celdas Logicas (k)	862	1,314	1,724	2,586	2,822	3,763
Flip Flops (k)	788	1,201	1,576	2,364	2,580	3,441
LUTs(k)	394	601	788	1,182	1,290	1,720
Total Block Ram (Mb)	25.3	36	50.6	76	71	94.5
DSP Slices	2,280	3,474	4,560	6,840	8,928	11,904

En las tablas se observa en sombreado los FPGA en los que se pueden implementar en su totalidad el microcontrolador comparando las familias se decidió usar la familia artix-7 ya que aparte de tener un coste de adquisición más bajo y ser utilizado en aplicaciones de bajo consumo de potencia, esta nos puede proveer los suficientes bloques de celdas lógicas para poder implementar el hardware descriptivo necesario, para nuestra aplicación. Dentro de la familia Artix-7 el XC7A75T tiene la capacidad suficiente para implementar el diseño del microcontrolador.

La síntesis de la implementación de los módulos requeridos en el FPGA seleccionado, proporciona el volumen de recursos utilizados en el FPGA los cuales se muestran en la Tabla 3.3.8.

Tabla 3.3.8 Recursos lógicos usados del microcontrolador

Recursos	Recursos utilizados
LUTs	3,883 (4870 estimados)
Flip-Flop	8,538 (9000 estimados)
BRAM	46 (50 estimados)

Se observa una diferencia entre los recursos lógicos utilizados en el FPGA seleccionado y los recursos lógicos estimados, esta diferencia estimada es muy cercana a la implementada, aun esta diferencia mínima se puede deber a que en la síntesis e implementación de todo el diseño en HDL se tiene una optimización por parte del IDE de Xilinx en las conexiones por lo que se ocupan menos recursos lógicos con respecto a los recursos lógicos que se requieren para cada módulo de manera individual.

### **3.4. Estrategia de desarrollo por modulo**

Para implementar los módulos presentados en el punto 3.2, por medio de hardware descriptivo se usará una herramienta de HLS proporcionada por Xilinx, este entorno de desarrollo es VIVADO Design Suite, este nos permite crear nuestros diseños de VHDL y Verilog así como también usar bloques de propiedad intelectual propios de Xilinx, la cual como principal característica de esta herramienta es el desarrollo de sistemas complejos y robustos en un tiempo menor de desarrollo.

Para la implementación de nuestro sistema se usarán bloques IP para cada módulo del diseño presentado. Al usar IPs tenemos una gran ventaja, ya que estos están en constante mantenimiento evitando la obsolescencia lo cual nos garantiza un diseño confiable en cuestión de funcionalidad además de ser posible la realización y el debug de varias pruebas de diseño sin tanta modificación a nivel de hardware descriptivo.

Para la implementación de los bloques de IPs se tiene que estudiar su documentación técnica de cada uno, así como sus APIs los cuales van a ser útiles para su programación, estos los proporciona el fabricante, así como las modificaciones necesarias que se pudieran hacer a los IPs para cumplir con los requisitos propuestos. [29] [30]

### **3.5. Diseño del hardware en el FPGA**

La implementación de hardware descriptivo se realizará en VIVADO Design Suite el cual es un entorno de desarrollo en HLS y la programación de los bloques implementados fue realizada en Software Development Kit (SDK, por sus siglas en ingles).

En el entorno de desarrollo integrado (IDE, por sus siglas en ingles), como se muestra en la figura 3.5.1, nos ayudara a la creación de Hardware descriptivo para sistemas embebidos en FPGA de Xilinx, en esta interfaz se puede elegir el componente FPGA a utilizar y además de poder utilizar bloques IPs propios de Xilinx, bloques IPs de código abierto o también códigos propios en VHDL, para su integración en HLS nos proporciona un entorno de programación visual por medio de bloques, en el cual se va utilizar en el programa una opción de integración de bloques IP, posteriormente se realizaran sus conexiones basándonos en las notas técnicas de cada IP, se asignaran las salidas y entradas al FPGA se finalizara el diseño de hardware descriptivo con la validación del diseño por medio de la síntesis y la generación del bitstream que es un archivo de programación del FPGA.

Una vez teniendo la validación del hardware descriptivo se procederá a programar el microcontrolador por medio del SDK, su lenguaje de programación va a ser en C, en este nos permite programar el procesador MicroBlaze con una biblioteca de APIs específicamente diseñados para el microprocesador, además de incluir códigos de ejemplo y notas técnicas de soporte, vea figura 3.5.2.

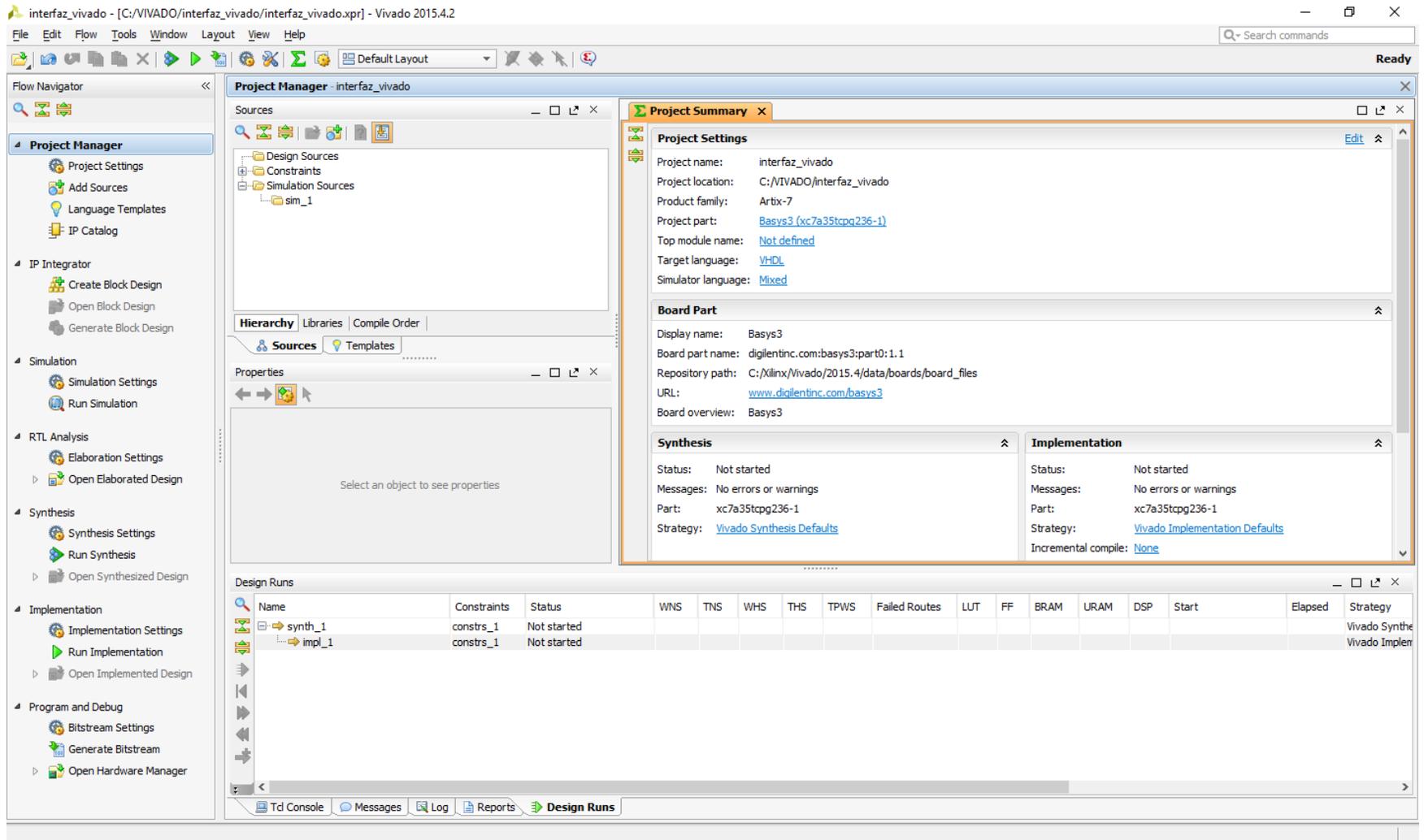


Figura 3.5.1 Vivado design suite

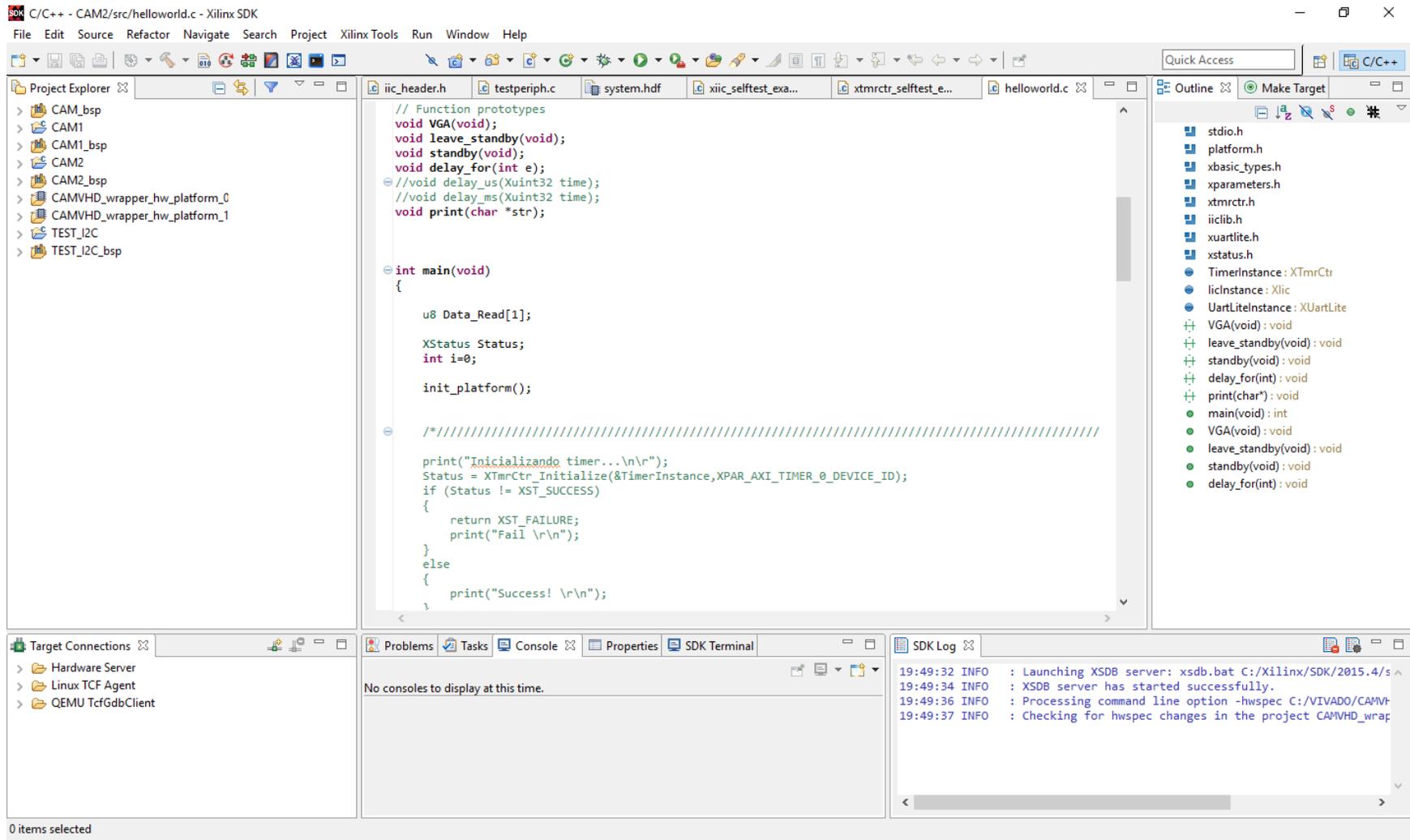


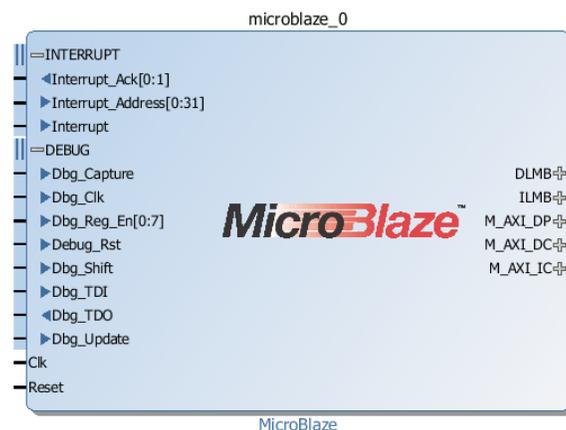
Figura 3.5.2 SDK

En el diseño en hardware del microcontrolador en FPGA, se tiene que tener en cuenta la implementación de la lógica necesaria que cumpla con los requerimientos generales en el punto 3.2.

El FPGA debe tener los siguientes bloques IP y lógica descriptiva implementada:

-Procesador Soft core: se decidió usar MicroBlaze, este es un Soft core optimizado para su uso en FPGAs de Xilinx, la ventaja de utilizar este Soft core y no un procesador embebido, como Zync o PowerPC ambos embebidos como hardcore en FPGAs de Xilinx, es debido a su fácil integración en cualquier hardware de FPGAs de Xilinx, además MicroBlaze esta optimizado como un procesador de 32-bits con arquitectura RISC [31], el cual proporciona conjuntos de instrucciones optimizadas para aplicaciones embebidas con opciones configurables por el usuario, además de tener características avanzadas como cache de instrucciones y datos con interfaz AXI, unidad de punto flotante, unidad de administración de memoria (MMU, por sus siglas en ingles) y soporte a tolerancia a fallas, en la figura 3.5.3 inciso a) se muestra el bloque IP del Soft-Core, en la figura 3.5.3 inciso b) se muestra el diagrama a bloques de MicroBlaze.

a)



b)

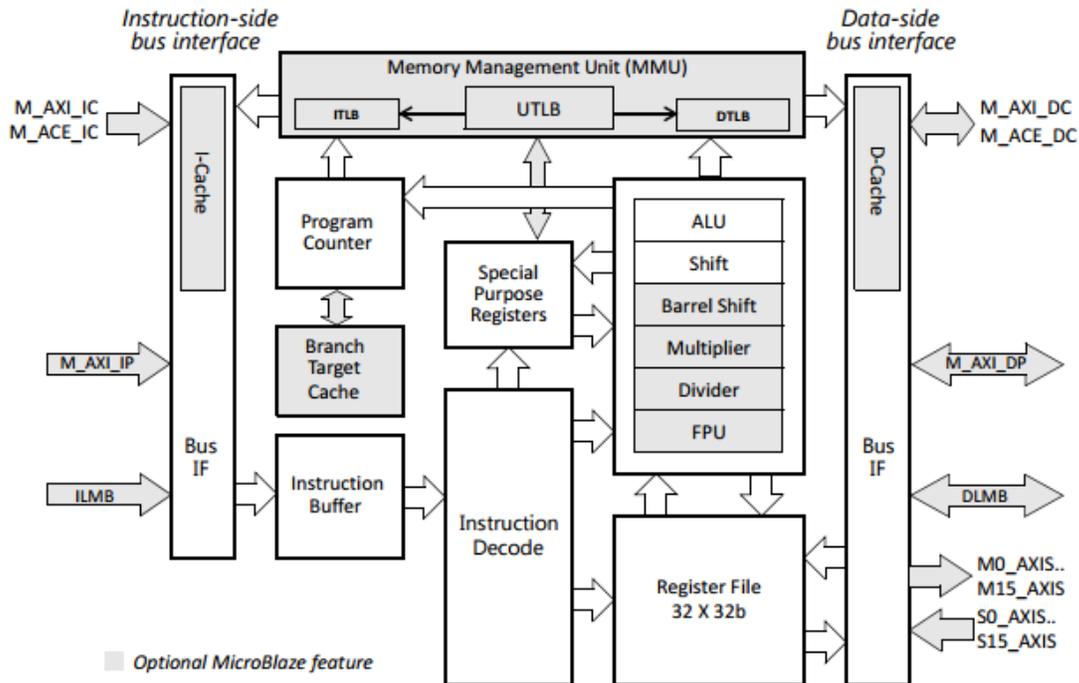


Figura 3.5.3 Diagrama a bloques de MicroBlaze

Este procesador cuenta con 32 registros de propósito general de 32 bits y 18 registros de propósito específico de 32 bits, vea tabla 3.5.1, en los registros de propósito específico se cuenta con [30]:

- Program Counter (PC): Es una dirección de 32 bits de ejecución de una instrucción.
- Machine Status Register (MSR): Este registro contiene el control y estado de bits del procesador; como el ACC, VMS, VM, UMS, UM, PVR, EIP, EE, DCE, DZO, ICE, FSL, BIP, AC, IE.
- Exception Address Register (EAR):
- Exception Status Register (ESR):
- Branch Target Register (BTR):
- Floating Point Status Register (FSR):
- Exception Data Register (EDR):

- Process Identifier Register (PID):
- Zone Protection Register (ZPR):
- Translation Look-Aside Buffer Low Register (TLBLO):
- Translation Look-Aside Buffer High Register (TLBHI):
- Translation Look-Aside Buffer Index Register (TLBX):
- Translation Look-Aside Buffer Search Index Register (TLBSX):
- Processor Version Register (PVR):

Mientras tanto para los 32 registros de propósito general. Estos registros se clasifican como volátiles, no volátiles y dedicados [32].

- Los registros volátiles (también conocidos como llamadas guardadas) se utilizan como registros temporales y no conservan valores a través de llamadas de función. Los registros R3 a R12 son volátiles, de los cuales R3 y R4 se usan para devolver valores a la función del llamante, si los hay. Los registros R5 a R10 se utilizan para pasar los parámetros entre las subrutinas.

- Los registros R19 a R31 conservan su contenido a través de las llamadas de función y, por lo tanto, se denominan como registros no volátiles. Se espera que la función de llamada guarde los registros no volátiles que están siendo utilizados.

- Ciertos registros se utilizan como registros dedicados y no se espera que los programadores los utilicen para ningún otro propósito.

- ◆ Los registros R14 a R17 se utilizan para almacenar la dirección de retorno de interrupciones, subrutinas y excepciones en ese orden. Las subrutinas se llaman usando la instrucción de ramificación y enlace, que guarda el contador de programa actual (PC) en el registro R15.

- ◆ Los punteros de datos pequeños se utilizan para acceder a determinadas ubicaciones de memoria con un valor inmediato de 16 bits. Estas áreas son

secciones de memoria vea [32]. El registro R2 (Read-Only) del área de datos pequeños (SDA) es de sólo lectura se utiliza para tener acceso a las constantes. El otro registro SDA R13 (lectura-escritura) se utiliza para acceder a los valores en la sección de lectura y escritura de datos pequeños.

- ◆ El registro R1 almacena el valor del puntero de la pila y se actualiza al entrar y salir de las funciones.
- ◆ El registro R18 se utiliza como un registro temporal para las operaciones del ensamblador.

MicroBlaze incluye registros de propósito especial y de propósito general, los registros de propósito especial son diferentes a los registros de propósito específico [32].

Tabla 3.5.1 Convenciones para el uso de registros en MicroBlaze

Registro	Tipo	Implementación	Descripción
R0	Dedicado	HW	Valor 0
R1	Dedicado	SW	Stack Pointer
R2	Dedicado	SW	Read-only small data area anchor
R3-R4	Volátil	SW	Regresa parámetros/ registro temporal
R5-R10	Volátil	SW	Envía parámetros / registro temporal
R11-R12	Volátil	SW	Registro Temporal
R13	Dedicado	SW	Read-write small data area anchor
R14	Dedicado	HW	Regresa dirección de una interrupción

R15	Dedicado	SW	Regresa dirección para subrutina
R16	Dedicado	HW	Regresa dirección para trampa (Depurador)
R17	Dedicado	HW/SW	Regresa dirección para excepciones
R18	Dedicado	SW	Reservado para ensamblador
R19-R31	No- volatil	SW	Must be saved across function calls. Callee-save
RPC	Especial	HW	Program counter
RMSR	Especial	HW	Machine Status Register
REAR	Especial	HW	Exception Address Register
RESR	Especial	HW	Exception Status Register
RFSR	Especial	HW	Floating Point Status Register
RBTR	Especial	HW	Branch Target Register
REDR	Especial	HW	Exception Data Register
RPID	Especial	HW	Process Identifier Register
RZPR	Especial	HW	Zone Protection Register
RTLBLO	Especial	HW	Translation Look-Aside Buffer Low Register

RTLBHI	Especial	HW	Translation Look-Aside Buffer High Register
RTLBX	Especial	HW	Translation Look-Aside Buffer Index Register
RTLBSX	Especial	HW	Translation Look-Aside Buffer Search Index
RPVR0- RPVR11	Especial	HW	Processor Version Register 0 through 11

Para la implementación del generador de señales de reloj el cual va a ser el encargado de generar señales de reloj hasta de 500 MHz requerida por propósito general, se implementó por medio del bloque “Clocking wizard” como se muestra en la figura 3.5.4 inciso a), el cual va a generar señales de reloj según se requieran, en el inciso b) se muestra su diagrama a bloques del IP . Los principales motivos en la elección de este bloque IP son:

- Permite crear una red de relojes con frecuencia, fase y ciclo de trabajo con efecto de jitter reducido.
- Reducción de la interferencia electromagnética.

a)

Component Name: partes\_clk\_wiz\_0\_1

The phase is calculated relative to the active input clock.

Output Clock	Output Freq (MHz)		Phase (degrees)		Duty Cycle (%)
	Requested	Actual	Requested	Actual	Requested
<input checked="" type="checkbox"/> clk_out1	100.000	100.000	0.000	0.000	50.000
<input type="checkbox"/> clk_out2	100.000	N/A	0.000	N/A	50.000
<input type="checkbox"/> clk_out3	100.000	N/A	0.000	N/A	50.000
<input type="checkbox"/> clk_out4	100.000	N/A	0.000	N/A	50.000
<input type="checkbox"/> clk_out5	100.000	N/A	0.000	N/A	50.000
<input type="checkbox"/> clk_out6	100.000	N/A	0.000	N/A	50.000
<input type="checkbox"/> clk_out7	100.000	N/A	0.000	N/A	50.000

USE CLOCK SEQUENCING:

Output Clock Sequence Number:

Output Clock	Sequence Number
clk_out1	1
clk_out2	1
clk_out3	1
clk_out4	1
clk_out5	1
clk_out6	1
clk_out7	1

Enable Optional Inputs / Outputs:  Reset Type: \_\_\_\_\_

b)

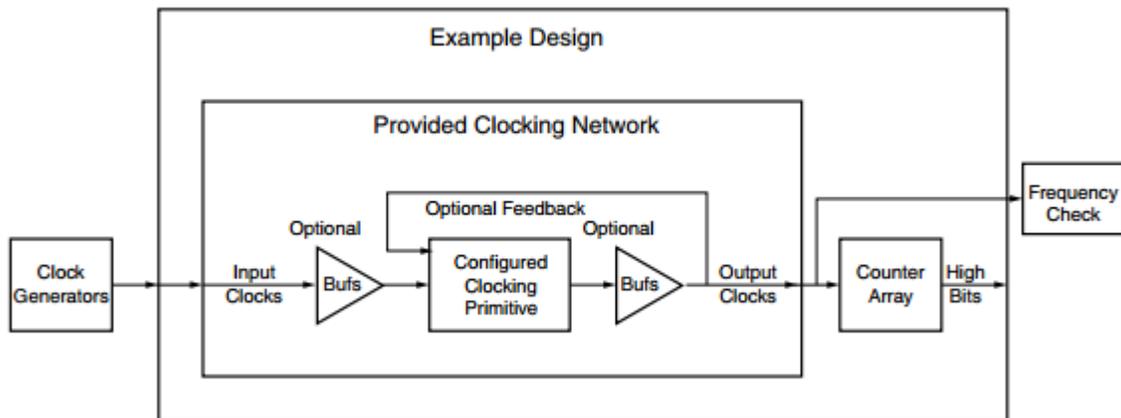


Figura 3.5.4 IP Clocking Wizard

Las funciones de este bloque IP también incluyen:

- Síntesis de frecuencias. Esta característica permite que los relojes de salida tengan frecuencias diferentes que el reloj de entrada.

- Espectro ensanchado. Esta característica ofrece relojes de salida modulados que reducen la densidad espectral de la interferencia electromagnética (EMI) generada por dispositivos electrónicos.
- Alineación de fases. Esta característica permite que el reloj de salida esté sincronizado como el pin de reloj de entrada para un dispositivo.
- Desplazamiento de fase dinámico. Esta función le permite cambiar la relación de fase en la salida de diferentes salidas de reloj.

Para la implementación de los módulos GPIO se optó usar bloques de IP GPIO los cuales van a ser controlados por MicroBlaze, estos bloques IPs se agregaron del catálogo de Xilinx y se configuraron en HLS, estos van a ser utilizados para recibir señales digitales e interrupciones, estos bloques implementados se muestran en la figura 3.5.5 inciso a).

a)

#### AXI GPIO (2.0)

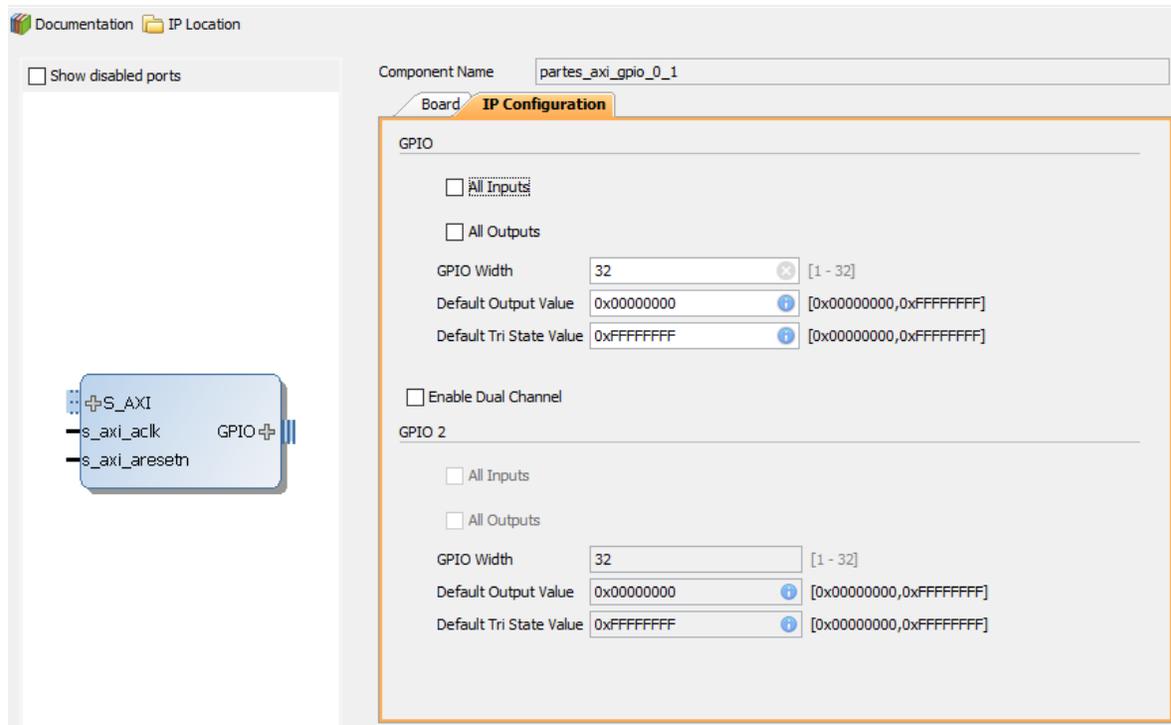


Figura 3.5.5 Bloque GPIO

b)

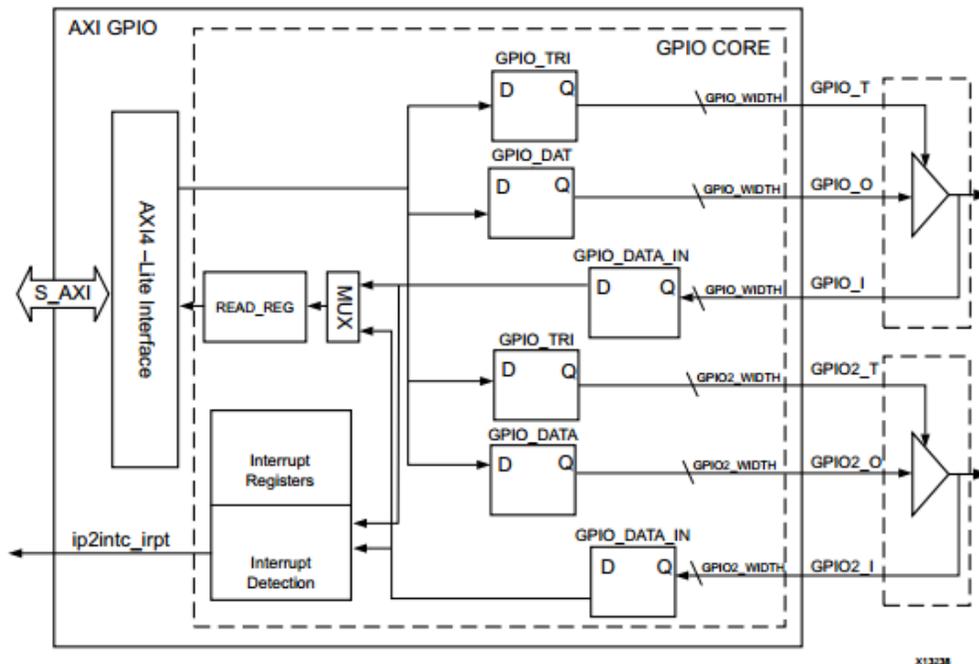


Figura 3.5.5 Bloque GPIO

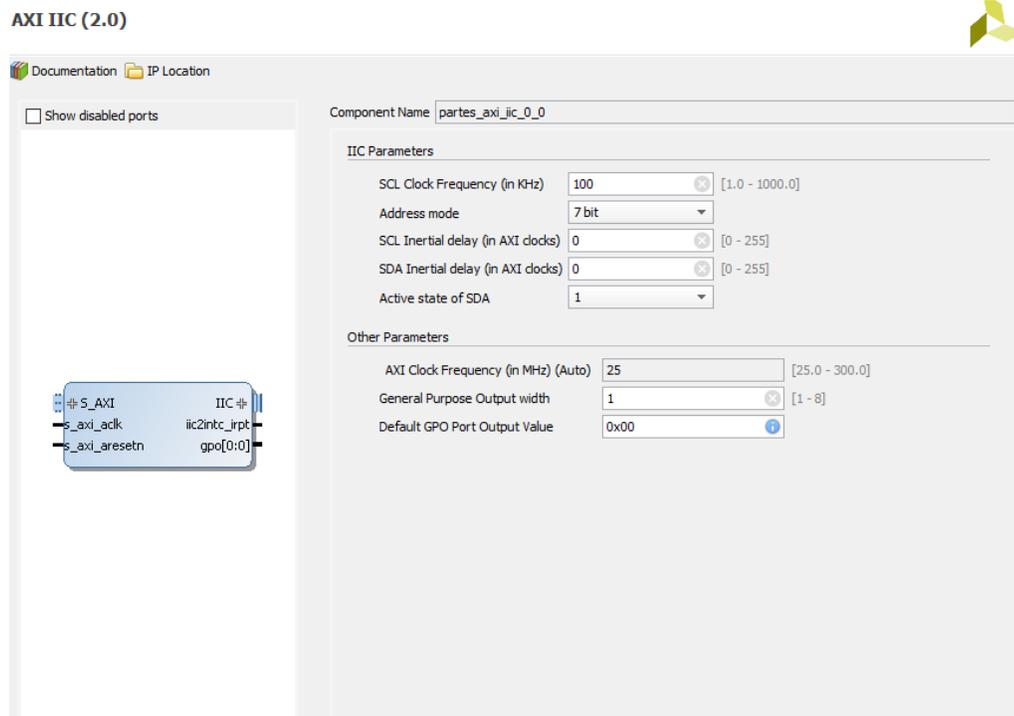
En la figura 3.5.5 inciso b) se muestra el diagrama a bloques del módulo, donde la conexión de este bloque al procesador se realiza por el bus AXI4, en esta configuración se podrá configurar los puertos GPIO como entrada / salida de propósito general a una interfaz.

Como característica de este bloque tiene que:

- Soporta una configuración de frecuencia máxima para el FPGA seleccionado de 120 MHz
- Soporta la interfaz AXI4-Lite, la cual es compatible con MicroBlaze
- Soporta el ancho de canal configurable para puertos GPIO de 1 a 32 bits
- Soporta la programación dinámica de cada GPIO como entrada o salida
- Soporta la configuración individual de cada canal
- Soporta valores de restablecimiento independientes para cada bit de todos los registros
- Soporta programación de interrupciones

Para la implementación de la interfaz I2C que es esencial para la configuración de sensores que usen este bus de comunicación. Se implementó por medio de un bloque IP IIC seleccionado del catálogo de Xilinx el cual va a ser controlado por el microprocesador, vea Figura 3.5.6 inciso a).

a)



b)

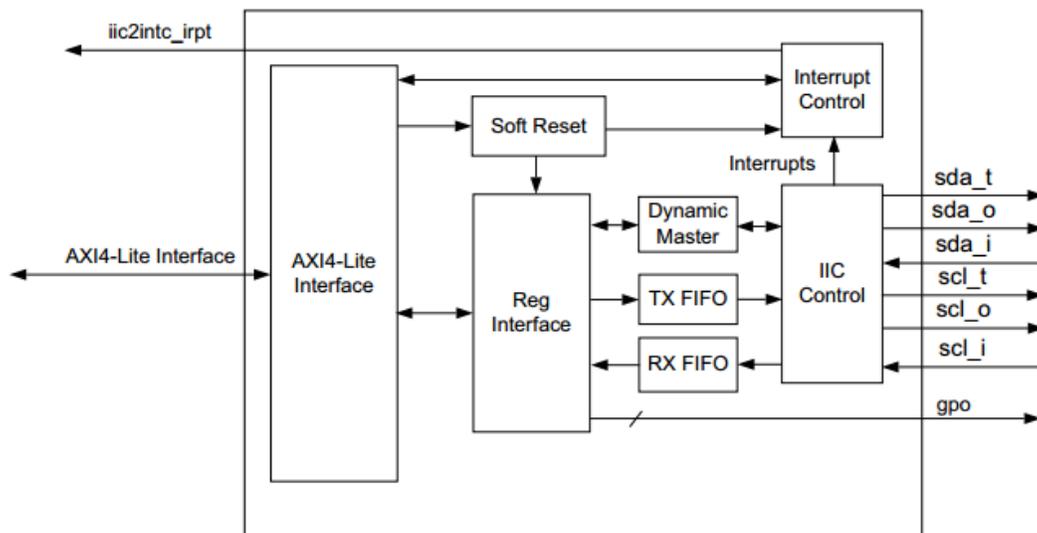


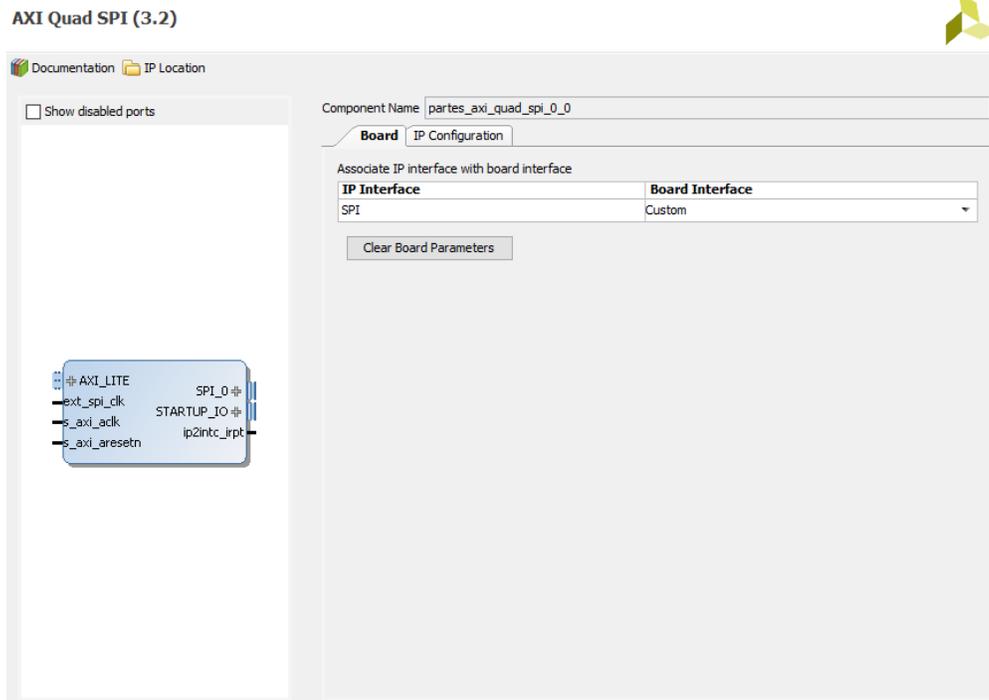
Figura 3.5.6 IP I2C

En el diagrama de bloques de la figura 3.5.6 inciso b) se observa sus características donde son las siguientes:

- Cumple con el protocolo I2C estándar de la industria
- Compatible con la interfaz AXI4-Lite de MicroBlaze
- Funcionamiento programable de maestro o esclavo
- Operación de varios maestros
- Bit de confirmación seleccionable por software
- Detección y generación de señal de “START” y “STOP”
- Generación de señal START repetida
- Reconoce la generación y detección de bits
- Detección de ocupado del bus
- Configuración de modo “Fast-plus” 1 MHz y modo rápido 400 kHz, o modo estándar de operación de 100 kHz
- Direccinamiento de 7 o 10 bits
- Transmitir y recibir FIFOs - 16 bytes de profundidad
- Salida de uso general, de 1 a 8 bits de ancho
- Generación dinámica de arranque y paro

-Comunicación SPI: el bloque SPI va ser necesario para la ejecución de comandos y control de sensores que lo necesiten y utilicen este protocolo de comunicación, su implementación fue por medio de un bloque IP SPI como se muestra en la Figura 3.5.7 inciso a).

a)



b)

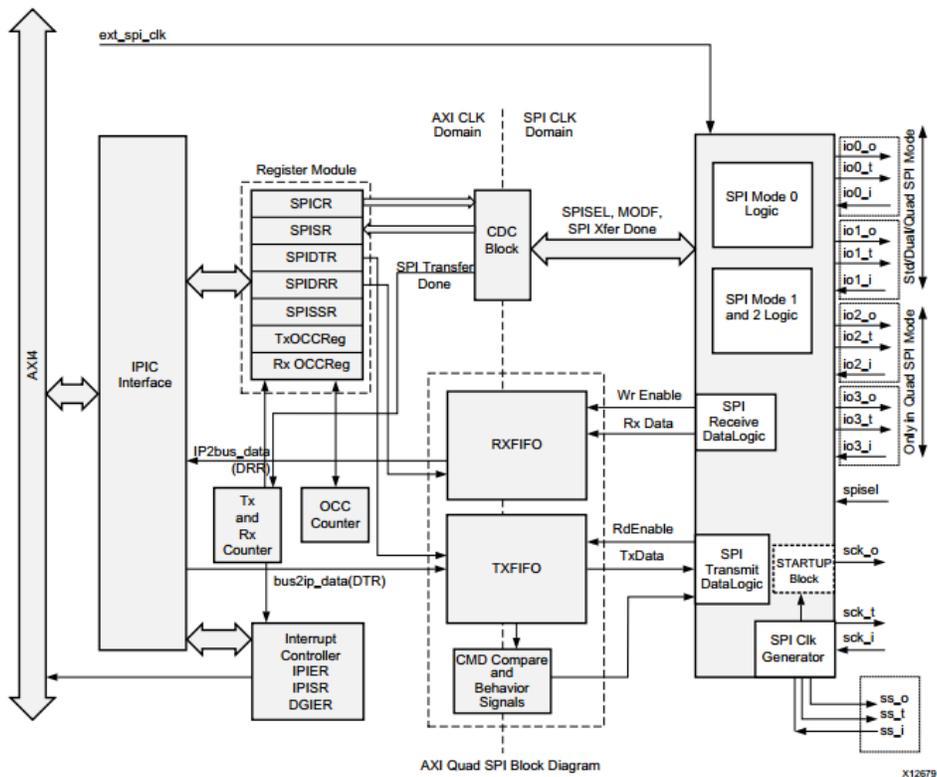


Figura 3.5.7 Bloque IP SPI

En la figura 3.5.7 inciso b) se muestra su diseño a bloques. El bloque IP de protocolo SPI cuenta con lo siguiente:

- Interfaz configurable AXI4
- Se conecta como un esclavo de 32 bits en AXI4-Lite o interfaz AXI4
- Modos SPI configurables:
  - Modo SPI estándar
  - Modo SPI dual
  - Modo SPI Cuádruple
- Fase y polaridad programables del reloj SPI
- Profundidad FIFO configurable (16 o 256 elementos profundidad en modo Dual / Quad / SPI estándar) y profundidad FIFO fijo de 64.

Para la implementación del módulo UART se usó el bloque IP Uartlitle de Xilinx, este bloque puede configurarse con un baudrate de 110 a 921600, con una trama de datos de 5 a 8 bits y un bit de paridad, para fines prácticos se configuro a un baudrate de 9600 con una trama de datos de 8 bits y sin bit de paridad, una vez terminada la configuración del bloque IP se procedió a utilizar la conexión automática de Xilinx para conectar el bloque IP UART con el softcore, vea la Figura 3.5.8 inciso a), en la figura 4.5.8 inciso b) se muestra su diagrama a bloques del módulo IP.

a)

## AXI Uartlite (2.0)



Documentation IP Location

Show disabled ports

Component Name partes\_axi\_uartlite\_0\_1

Board **IP Configuration**

AXI CLK Frequency (Auto) 100 [10-300]MHz

Baud Rate 9600

Data Bits 8 [5 - 8]

Parity

No Parity  Odd  Even



b)

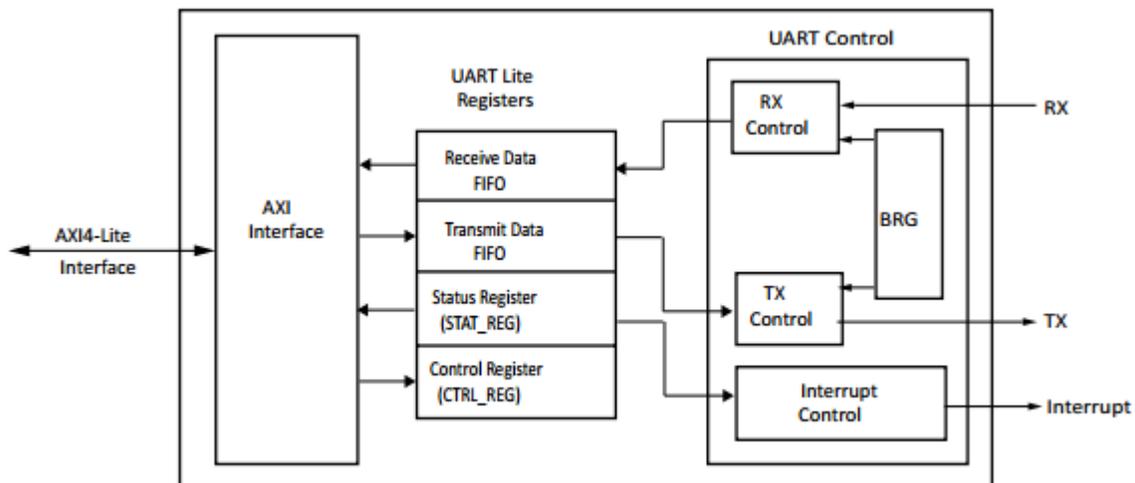


Figura 3.5.8 Bloque UART

También es necesario implementar dos tipos de memoria, una para instrucciones y otra para datos, para poder tener una arquitectura tipo Harvard, para esto se usó el bloque IP Block Memory Generator, vea figura 3.5.9, este bloque IP tiene las siguientes características:

- Soporta protocolos de interfaz AXI4 y AXI4-Lite compatibles con MicroBlaze
- Canales independientes de lectura y escritura
- Transferencias en ráfaga AXI estrechas y no alineadas
- Configuraciones primitivas de RAM de doble puerto simple
- Rendimiento de hasta 300 MHz
- Soporta ancho de datos hasta 256 bits y profundidad de memoria de 1 a 1M palabras

#### Block Memory Generator (8.3)

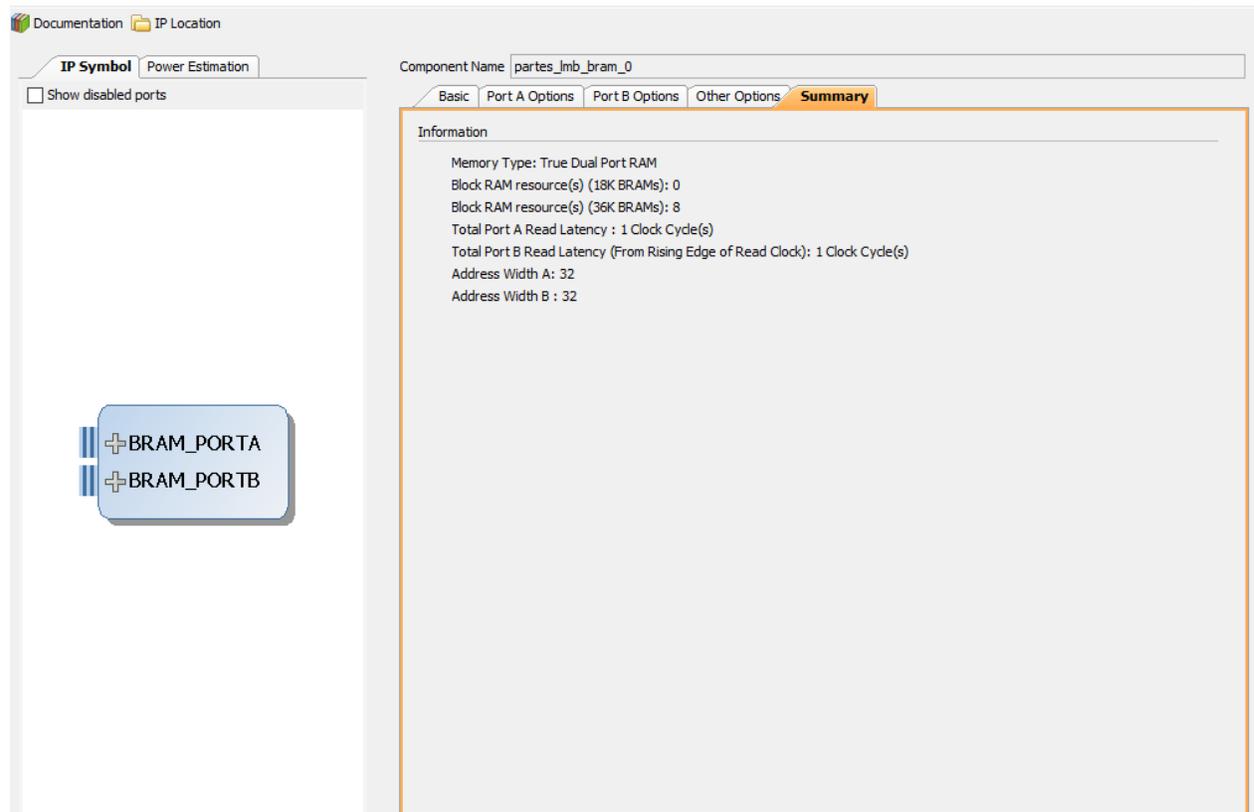


Figura 3.5.9 Bloque de Memoria

Por lo que nuestro diseño del microcontrolador con los módulos necesarios en el FPGA se muestra en la figura 3.5.10, se puede observar que el diseño final atiende a los requerimientos generales, implementando los principales módulos de comunicación para operar como un microcontrolador diseñado a la medida como se planteó como requerimientos generales en el punto 3.1, con la característica de tener un conjunto de instrucciones tipo RISC dadas por el SoftCore, al implementar el tipo de instrucciones RISC ponemos nuestro microcontrolador en la tendencia actual entre los que destacan el uso de procesadores ARM debido a que emplean el mismo conjunto de instrucciones RISC.

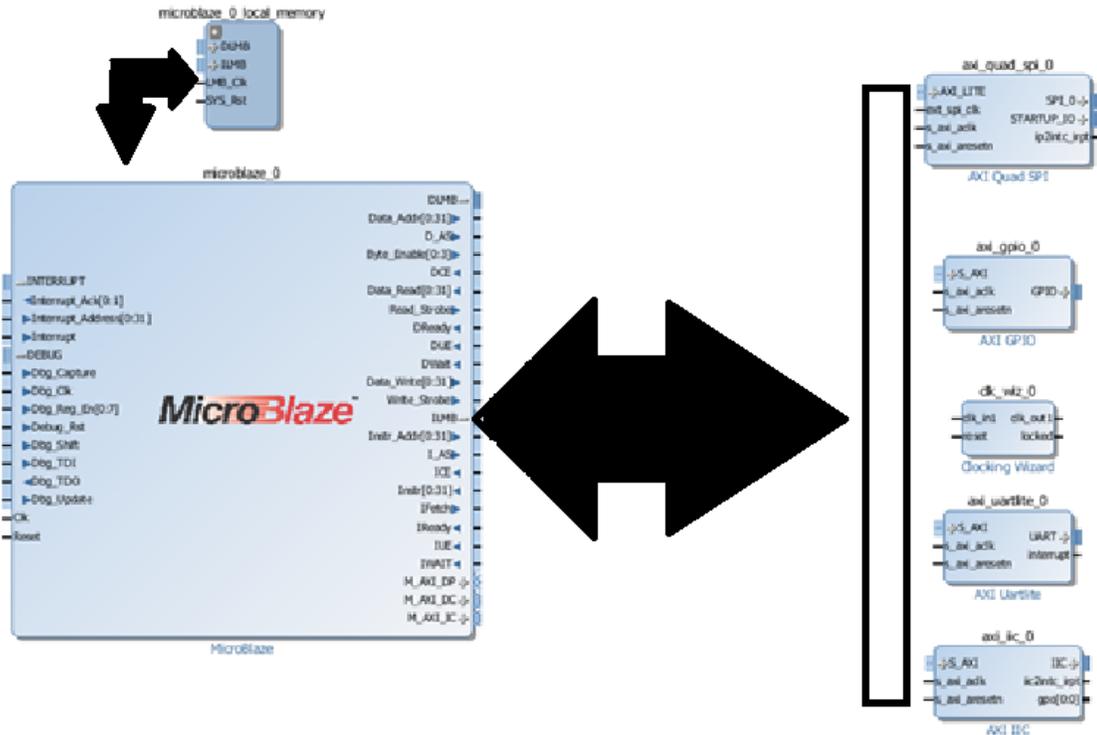


Figura 3.5.10 Implementación total de manera conceptual del microcontrolador en HDL

## Capítulo 4. DESARROLLO Y CONSTRUCCION

### 4.1 Implementación del núcleo de la computadora, pruebas y resultados

Para la implementación del Soft-Core se integra el bloque IP de MicroBlaze al entorno de desarrollo HLS como se muestra en la figura 3.5.1 inciso a), para lograr el funcionamiento de este soft core se requiere implementar lógica adicional al IP, esta lógica adicional va a ser una memoria local “Block Memory Generator”, la cual es un bloque funcional IP que su principal característica es el uso de los bloques RAM embebidos en los FPGAs de Xilinx, esta memoria se encargara de guardar el programa a ejecutar por el microprocesador, y la memoria cache para manipular los datos del programa de forma rápida , además de ser necesario la implementación de un módulo “debug” el cual nos auxiliará en la programación y depuración del programa, un bloque de sincronía “Processor System Reset” para implementar las conexiones entre cada bloque descrito anteriormente, por lo que se implementaron y configuraron de la siguiente manera, en la configuración del MicroBlaze se seleccionó un reloj de 100MHz el cual es la frecuencia máxima soportada por el IP, para la configuración de las memorias se decidió usar la máxima capacidad la cual es 128 KB de memoria local y 64 KB de memoria de cache.

Para operar el “Soft core” se tiene que conectar a las señales de reloj y reset, estas señales se conectaron al IP “Clocking Wizard” este es un IP generador de señales el cual nos ayuda a mantener la sincronía entre todos los bloques IP como lo son el Soft core y el Processor System Reset, así mismo nos ayuda a evitar problemas de impedancia si se requiere usar más conexiones de reloj y poder generar más señales de reloj a partir de la de entrada, en este IP se configuro la señal de entrada a 100 MHz el cual es el reloj físico de la tarjeta y una señal de salida del bloque de 100 MHz este bloque sirve como buffer para dar sincronía a los demás bloques IP y el reset se configuro como flanco de bajada.

Para las conexiones de cada bloque IP se auxilió del IDE de Xilinx y de las notas técnicas proporcionadas por Xilinx para realizar las conexiones necesarias de los IPs estas notas pueden ser consultadas en las referencias [30] [31]. La implementación del Soft core en HLS se muestra en la figura 4.1.1, en la figura 4.1.2 se muestra el total de recursos usados del FPGA en la implementación del núcleo.

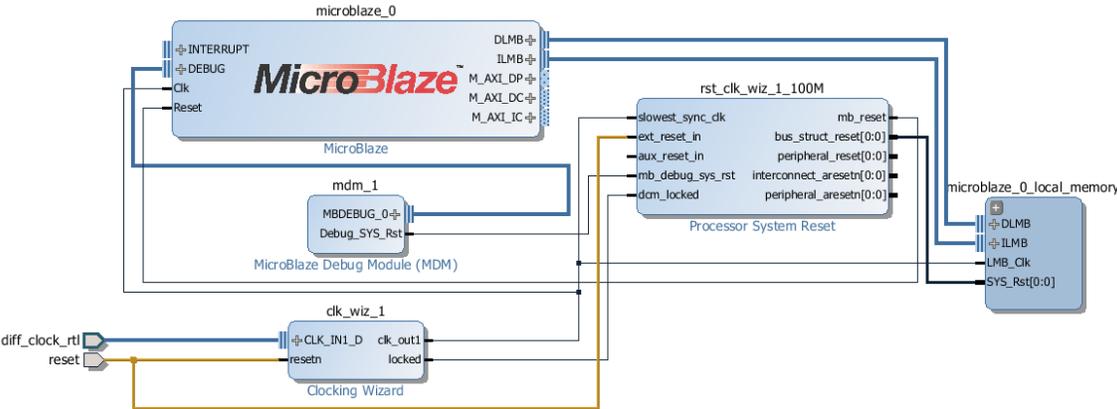


Figura 4.1.1 Implementación del Soft-Core

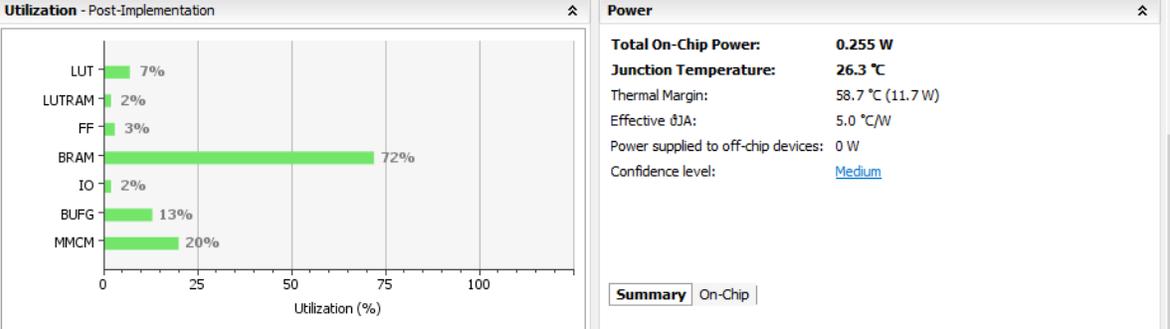
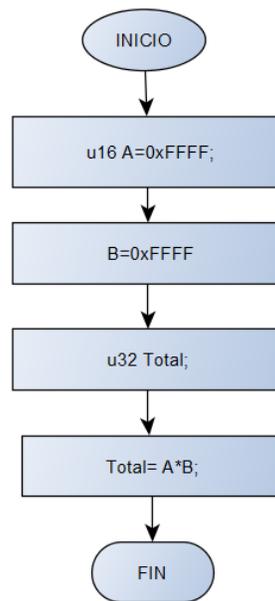


Figura 4.1.2 Implementación del Soft-core

Para demostrar el funcionamiento del Soft-Core implementado se realizó una multiplicación de dos números de 16 bits donde el resultado de esta operación sería un número de 32 bits su diagrama de flujo se muestra en la figura 4.1.3 inciso a) y su código en lenguaje C se muestra en la figura 4.1.3 inciso b), esta operación también comprueba que el procesador es de 32 bits, los resultados prácticos se muestran en las figuras 4.1.4 y 4.1.5.

a)



b)

```
#include <stdio.h>
#include "platform.h"
#include "stdint.h"

void print(char *str);

int main()
{
    uint16_t A=0xFFFF;
    uint16_t B=0xFFFF;
    uint32_t Total=0x00000000;
    Total=A*B;
    return 0;
}
```

Figura 4.1.3 Diagrama de la operación a realizar

Name	Value
(x)= A	0xffff
(x)= B	0xffff
(x)= Total	0x0

Figura 4.1.4 Datos antes de la multiplicación

Name	Value
(x)= A	0xffff
(x)= B	0xffff
(x)= Total	0xffff0001

Figura 4.1.5 Resultados después de la multiplicación

Además, se realizó una segunda prueba donde se cargaron valores de 32 bits a dos registros de propósito general los cuales se sumaron y el resultado se guardó en un registro diferente, en la figura 4.1.6 se muestra su diagrama de flujo de la programación en la figura 4.1.7 se muestra el código y en la figura 4.1.8 se muestra los resultados.

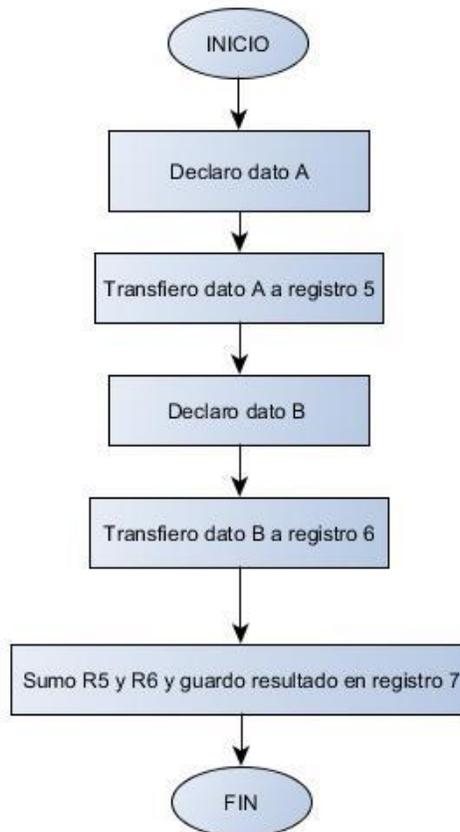


Figura 4.1.6 Diagrama de flujo de suma de dos números de 32 bits

```

#include <stdio.h>
#include "platform.h"

void print(char *str);
int main()
{
int a=0xAAAAAAAA;
asm("ADD r5,r3,r0"); // load word immediate
a=0x11111111;
asm("ADD r6,r3,r0"); // load word immediate
asm("ADD r7,r5,r6");
}

```

Figura 4.1.7 Código en c y ensamblador para la suma de 2 números de 32 bits

Name	Value
 r4	0
 r5	0xaaaaaaaa
 r6	0x11111111
 r7	0xbbbbbbbb

Figura 4.1.5 Resultados de una suma de 32 bits

## 4.2. Construcción y prueba del módulo UART, pruebas y resultados

En el diseño del microcontrolador en FPGA se implementó un bloque IP UART como se muestra en la figura 3.5.6, este bloque IP va a ser controlado por el Soft-Core, se observó que para la conexión al procesador es necesario implementar un bloque proporcionado por Xilinx llamado “axi interconnect” el cual se encarga de realizar múltiples conexiones lógicas de los módulos IP de entrada o salida con el

procesador mediante un bus configurable de 8,16,32 bits, este bus llamado Interconnect como se observa en la figura 4.2.1 va a direccionar los datos enviados por el procesador ya sea de escritura o lectura hacia los bloques esclavos, soportando modos de acceso síncronos y asíncronos, las conexiones de los bloques se hizo con la nota técnica del bloque y del IDE de Xilinx, como resultado de la conexión del bloque IP UART, se muestra en la figura 4.2.2.

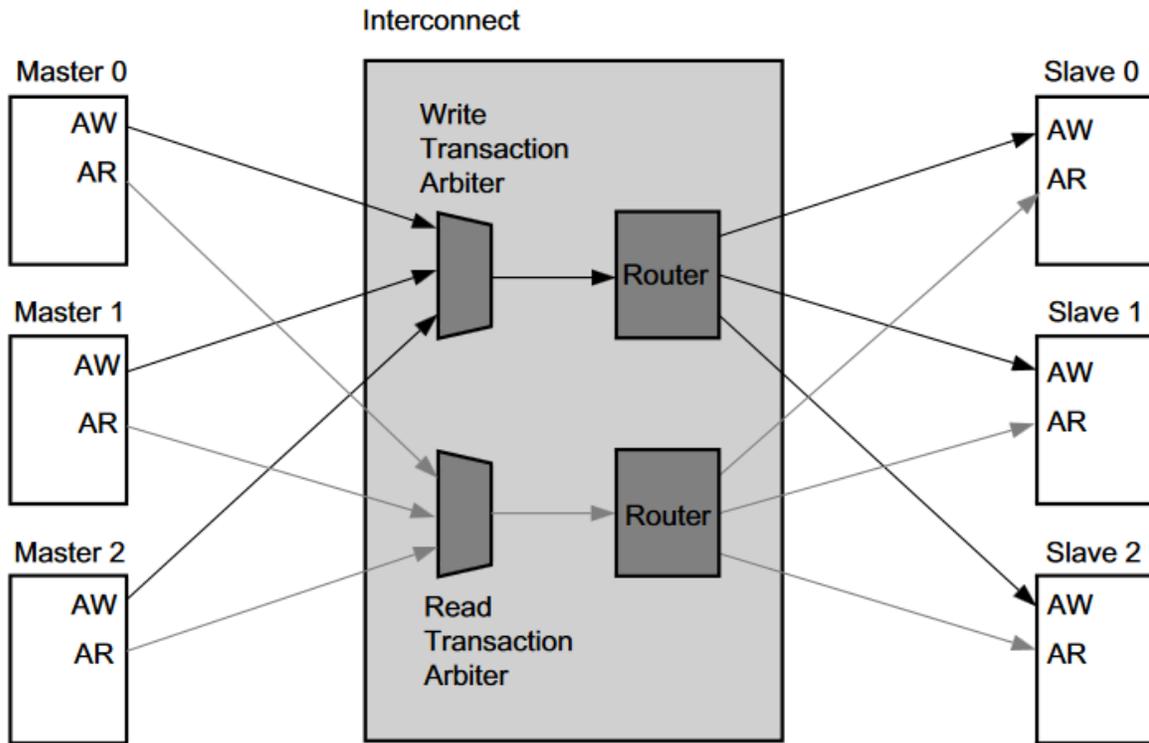


Figura 4.2.1 Bloque AXI Interconect

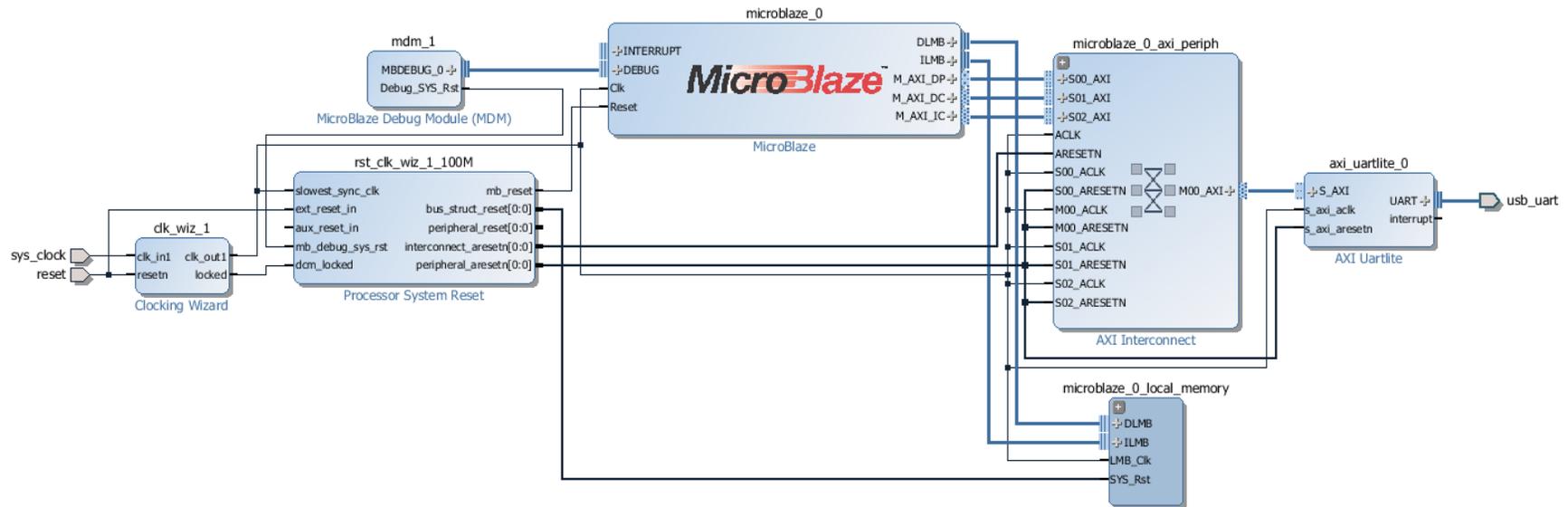
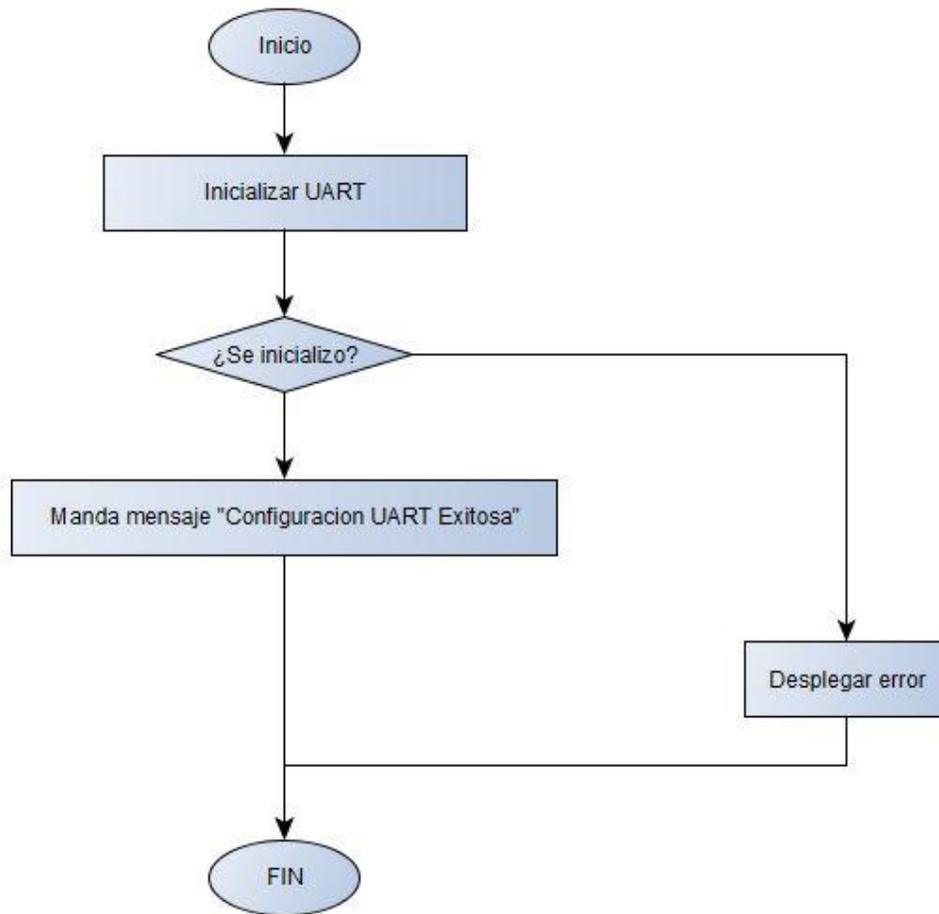


Figura 4.2.2 Implementación del módulo UART

Para la correcta validación de este módulo IP se realizó en software una ejecución de comandos en el SDK para verificar el correcto funcionamiento de la conexión del módulo y verificación del mismo en la terminal de la computadora, vea el diagrama en la figura 4.2.3 inciso a). En la figura 4.2.3 inciso b) se muestra su código en C y en la figura 4.2.4 se muestra los resultados.

a)



b)

```
#include <stdio.h>
#include "platform.h"

void print(char *str);

int main()
{
    init_platform();
    print("Configuración UART Exitosa\n\r");
    cleanup_platform();
    return 0;
}
```

Figura 4.2.3 Programación en SDK para validar módulo UART

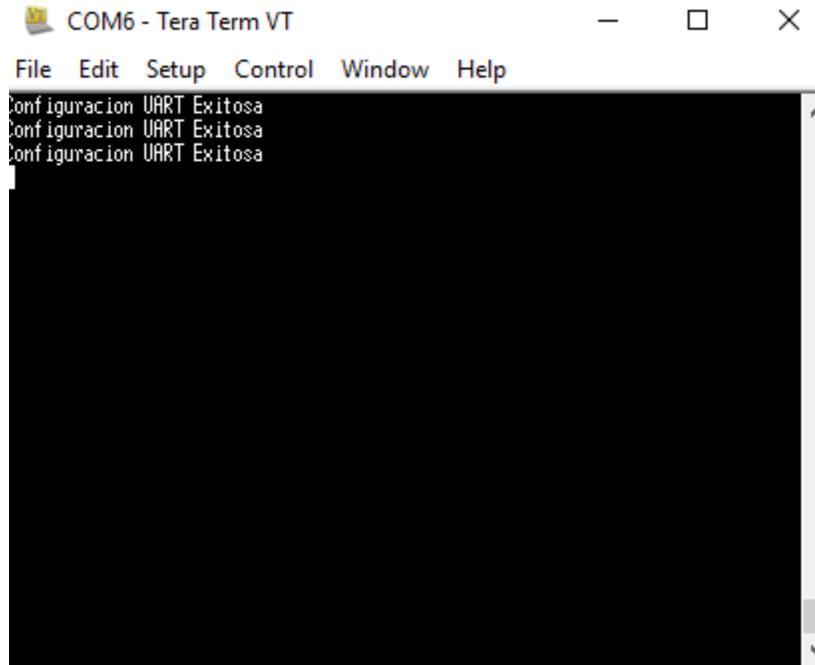


Figura 4.2.4 Resultados de prueba UART

### 4.3 Desarrollo y prueba de módulo GPIO e interrupciones

Para la implementación de las funcionalidades de GPIO e interrupciones primero se tiene que habilitar desde el IP de MicroBlaze la función de interrupción, esto quiere decir que al IP del procesador se le agrego esta función, al habilitarla MicroBlaze solamente puede manejar 1 interrupción y esta requiere de un bloque que detecte esta señal el bloque IP utilizado es "axi interrupt controller" vea figura 4.3.1, este bloque además ofrece un modo de interrupción rápida, configuración de niveles lógicos en la interrupción, si se llegara a necesitar más de una interrupción se tiene que usar un vector de interrupciones, este vector de interrupciones se puede implementar con el bloque IP "concat" vea figura 4.3.2, este bloque es usado para concatenar señales de entrada en un bus de salida por lo que en las interrupciones se usa de tal forma que ejecuta las interrupción mediante una lista FIFO.

Estos dos bloques ayudan a gestionar interrupciones entre los bloques IP GPIO y el Soft core, para las conexiones de estos bloques se auxilió del IDE de Xilinx y de las notas técnicas de los bloques IP y se muestra en la figura 4.3.3. La interrupción se habilita en HLS cuando se asigna al bloque GPIO que requiera manejar interrupciones.

### AXI Interrupt Controller (4.1)



The screenshot displays the configuration window for the AXI Interrupt Controller. On the left, a block diagram shows the component with ports: s\_axi, s\_axi\_aclk, s\_axi\_aresetn, intr[1:0], processor\_clk, processor\_rst, and an interrupt output. The main configuration area is divided into tabs: Basic, Advanced, and Clocks. The 'Basic' tab is active, showing the following settings:

- Component Name:** uart\_int\_microblaze\_0\_axi\_intc\_0
- Interrupt Usage:** Number of Peripheral Interrupts (Auto) set to 2.
- Fast Interrupt Mode:**  Enable Fast Interrupt Logic. Interrupt Vector Address Register reset value (Auto) set to 0x00000010.
- Peripheral Interrupts Type:** Three rows of configuration:
  - Interrupts type - Edge or Level: Radio buttons for Auto (selected), Edge, or Level. Value: 0xFFFFFFFF.
  - Level type - High or Low: Radio buttons for Auto (selected), High, or Low. Value: 0xFFFFFFFF.
  - Edge type - Rising or Falling: Radio buttons for Auto (selected), Rising, or Falling. Value: 0xFFFFFFFF.
- Processor Interrupt Type:** Interrupt type set to Level Interrupt, Level type set to Active High.

Figura 4.3.1 Bloque AXI interrupt controller

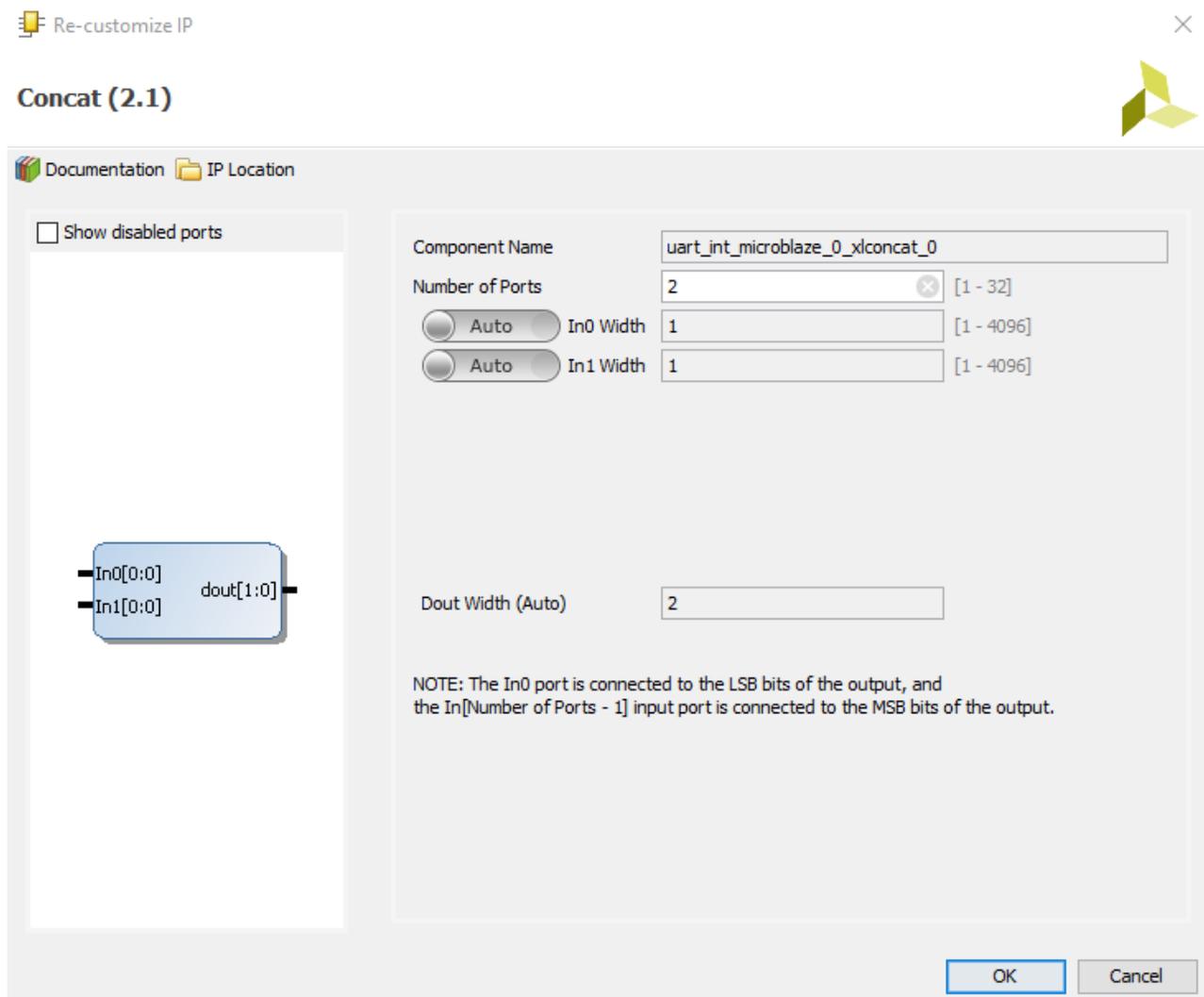


Figura 4.3.2 Bloque Concat

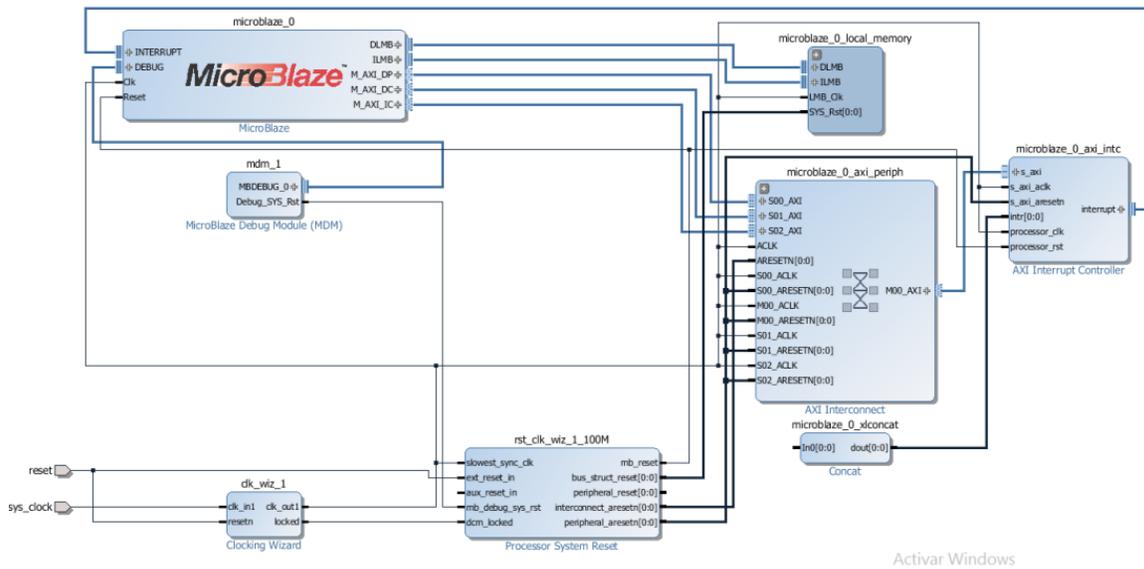


Figura 4.3.3 Implementación de interrupciones en MicroBlaze

Para implementar los bloques GPIOs se utilizó el bloque IP GPIO, para su conexión se realizó conforme a las notas técnicas y con el IDE de Xilinx, vea figura 4.3.4 su conexión se realizó mediante la comunicación de las líneas de control AXI hacia el bloque AXI Interconnect una vez realizado esto se configura en HLS el bloque IP para establecer el número de puentes GPIO deseados.

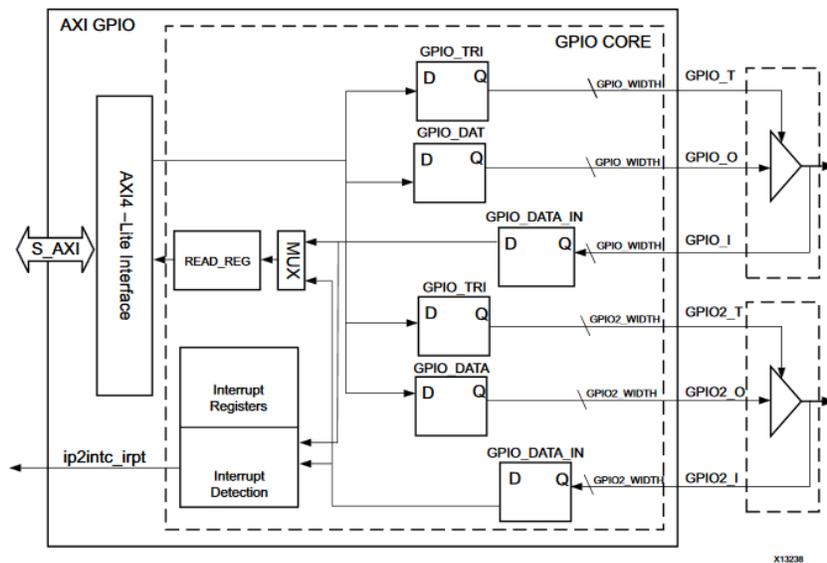


Figura 4.3.4 Diagrama a bloques del bloque IP GPIO

Por lo que los módulos GPIO junto con su vector de interrupciones y el Soft core quedarían como se muestra en la figura 4.3.5.

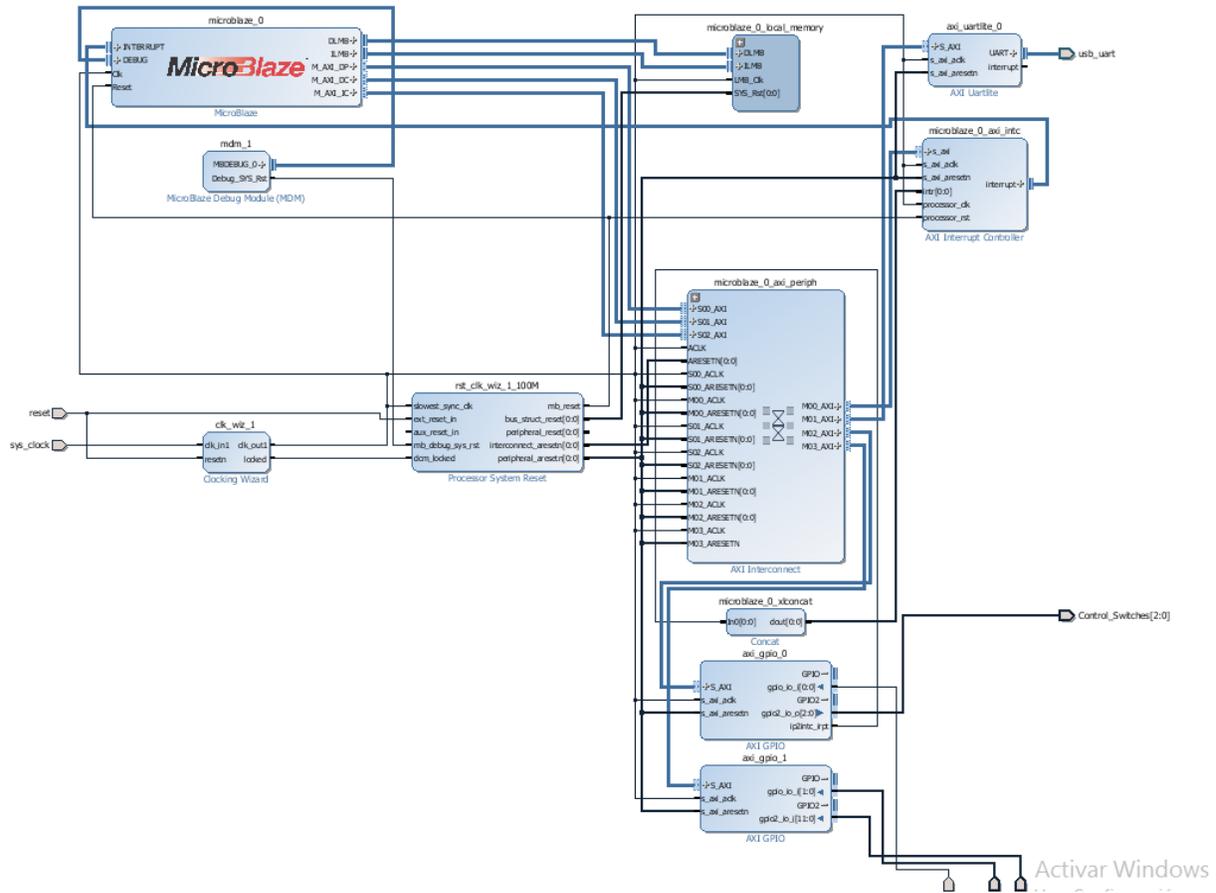


Figura 4.3.5 Bloques IP de GPIO con interrupciones en MicroBlaze

para verificar el correcto funcionamiento de los puertos GPIO conectados al procesador se validó el hardware mediante la ejecución de comando en el SDK, en la figura 4.3.6 se observa el método de validación en software, en la figura 4.3.7 se muestra su código en C, sus resultados fueron desplegados por comunicación UART como se muestra en la figura 4.3.8.

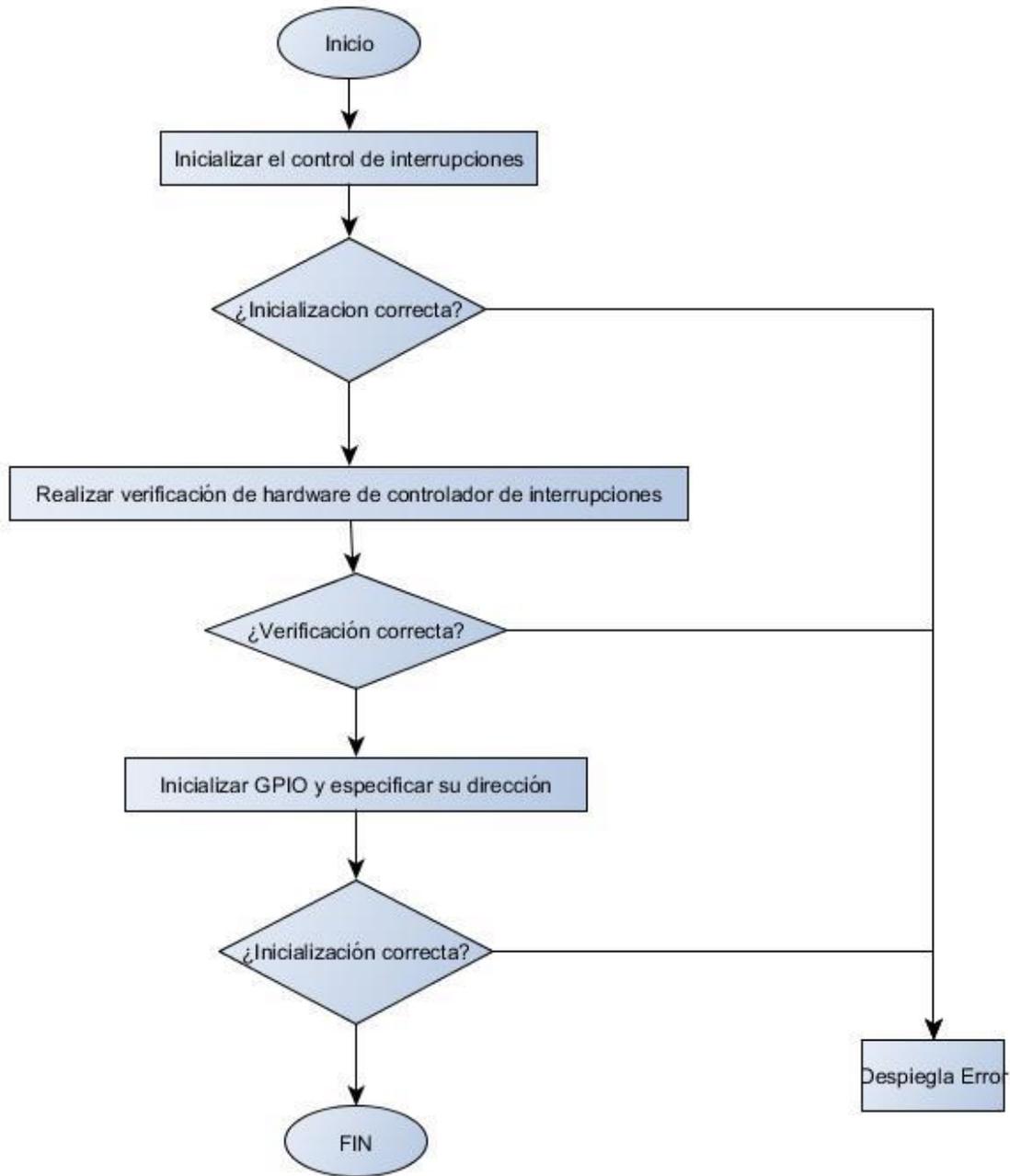


Figura 4.3.6 Validación de hardware en SDK de GPIO

```

#include <stdio.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "xintc.h"
#include "intc_header.h"
#include "xgpio.h"
#include "gpio_header.h"
int main()
{
    static XIntc intc;
    Xil_ICacheEnable();
    Xil_DCacheEnable();
    print("---Programa de validacion de modulos GPIO e interrupciones---\n\r");
    {
        int status;
        print("\r\n Validadndo comunicaci3n con modulo
microblaze_0_axi_intc...\r\n");
        status = IntcSelfTestExample(XPAR_MICROBLAZE_0_AXI_INTC_DEVICE_ID);
        if (status == 0) {
            print("Modulo de interrupcion VALIDADO\r\n");
        }
        else {
            print("Modulo de interrupcion ERROR\r\n");
        }
    }
    {
        int Status;
        Status = IntcInterruptSetup(&intc, XPAR_MICROBLAZE_0_AXI_INTC_DEVICE_ID);
        if (Status == 0) {
            print("Configuracion de interrupcion VALIDADO\r\n");
        }
        else {
            print("Configuracion de interrupcion ERROR\r\n");
        }
    }
    {
        u32 status;
        print("\r\nValidando modulos GPIO como entradas...\r\n");
        status = GpioOutputExample(XPAR_AXI_GPIO_0_DEVICE_ID,32);
        if (status == 0) {
            print("Etradas GPIO VALIDADA.\r\n");
        }
        else {
            print("Etradas GPIO ERROR.\r\n");
        }
    }
    print("---Programa Finalizado---\n\r");
    Xil_DCacheDisable();
    Xil_ICacheDisable();
    return 0;
}

```

Figura 4.3.7 C3digo en C

```

COM9 - Tera Term VT
File Edit Setup Control Window Help
---Programa de validacion de modulos GPIO e interrupciones---
Validadndo comunicacion con modulo microblaze_0_axi_intc...
Modulo de interrupcion VALIDADO
Configuracion de Interrupcion VALIDADA
Validando modulos GPIO como entradas...
Entradas GPIO VALIDADA.
---Programa Finalizado---

```

Figura 4.3.8 Despliegue de resultados de la validación en hardware del boque IP GPIO e interrupciones

Para demostrar el funcionamiento de los puertos GPIO se realizó un programa el cual prendiera y apagara leds como se muestra en la figura 4.3.9 y en la figura 4.3.10 se muestra su código

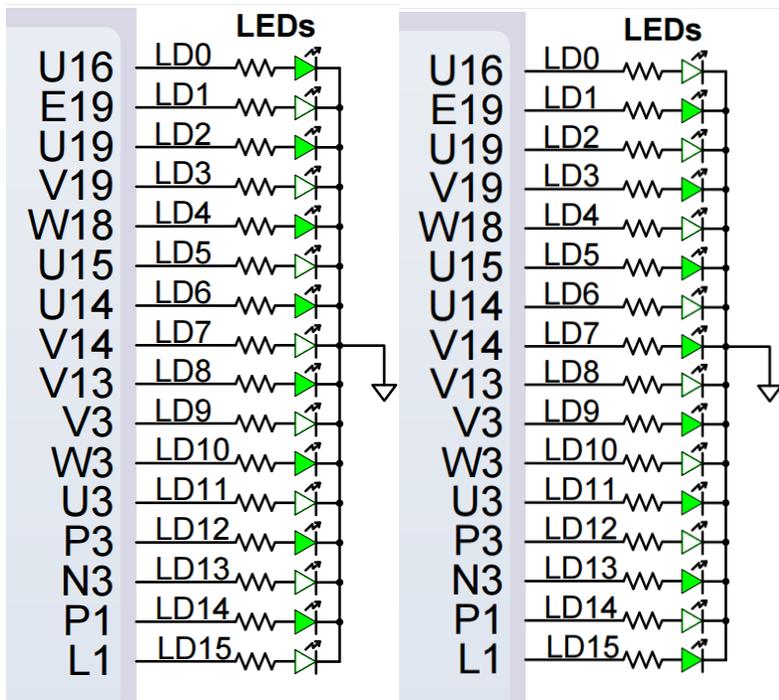


Figura 4.3.9 Ejemplo de GPIO

```

#include <stdio.h>
#include "platform.h"
#include "xgpio.h"
#include "xparameters.h"
#include "microblaze_sleep.h"

void print(char *str);

XGpio GPIO_0;
XGpio_Config GPIO_0_conf;

int main()
{
    GPIO_0_conf.BaseAddress = XPAR_AXI_GPIO_0_BASEADDR;
    GPIO_0_conf.DeviceId = XPAR_AXI_GPIO_0_DEVICE_ID;
    GPIO_0_conf.InterruptPresent = XPAR_GPIO_0_INTERRUPT_PRESENT;
    GPIO_0_conf.IsDual = XPAR_GPIO_0_IS_DUAL;

    XGpio_CfgInitialize(&GPIO_0, &GPIO_0_conf, GPIO_0_conf.BaseAddress);

    while(1){

        XGpio_DiscreteWrite(&GPIO_0, 2, 0x0000AAAA);

        MB_Sleep(100);
        XGpio_DiscreteWrite(&GPIO_0, 2, 0x00005555);///  

        MB_Sleep(100);
    }

    return 0;
}

```

Figura 4.3.10 Código de ejemplo de GPIOs

## 4.4. Evaluación y prueba del módulo I2C

Para implementar la comunicación I2C se agregó al diseño el bloque IP AXI IIC, para su conexión del bloque, vea figura 4.4.1 en sus conexiones se conectaron como periféricos al Soft core por medio del bloque AXI Interconnect que se mencionó anteriormente, esta conexión se muestra en la figura 4.4.2., posteriormente en el entorno de desarrollo se configuró el bloque IP con fines de pruebas con una frecuencia de reloj (SCLK) de 100 KHz, siendo que este bloque también permite que su frecuencia de reloj pueda ser configurada con 1KHz a 1MHz.

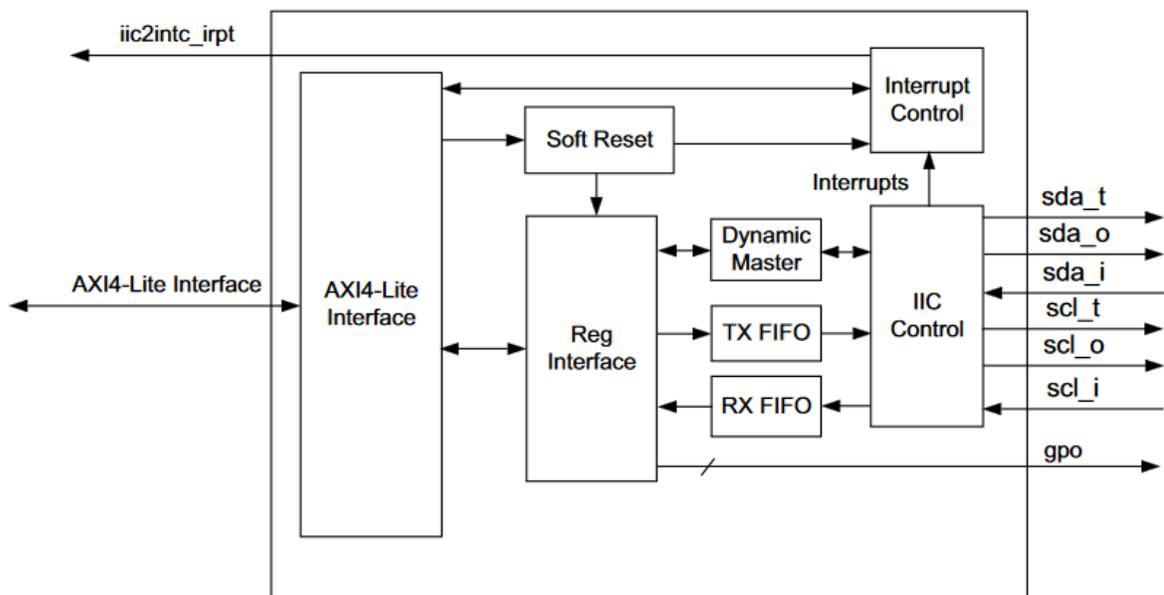


Figura 4.4.1 Diagrama a bloques del IP IIC

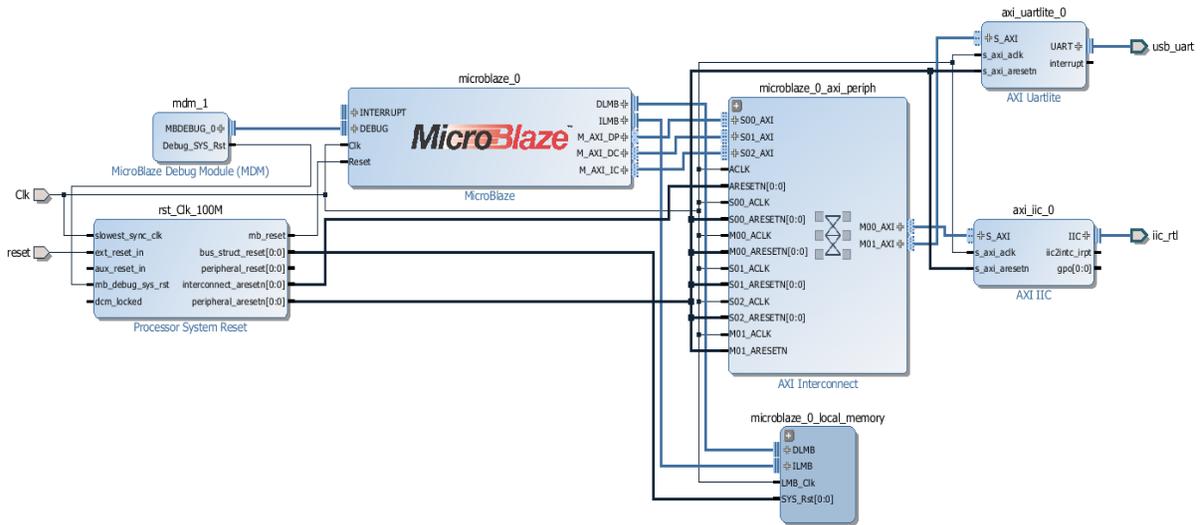


Figura 4.4.2 Configuración del módulo I2C

Para la correcta validación de este módulo IP se realizó en software una ejecución de comandos en el SDK para verificar el correcto funcionamiento de la conexión del módulo I2C al Soft Core, esto se observa en la figura 4.4.3, en la figura 4.4.4 se muestra el código escrito, el resultado de estas validaciones se desplegó por comunicación UART, el módulo de comunicación UART sólo es usado para validar las instrucciones enviadas como se muestra en la figura 4.4.5.

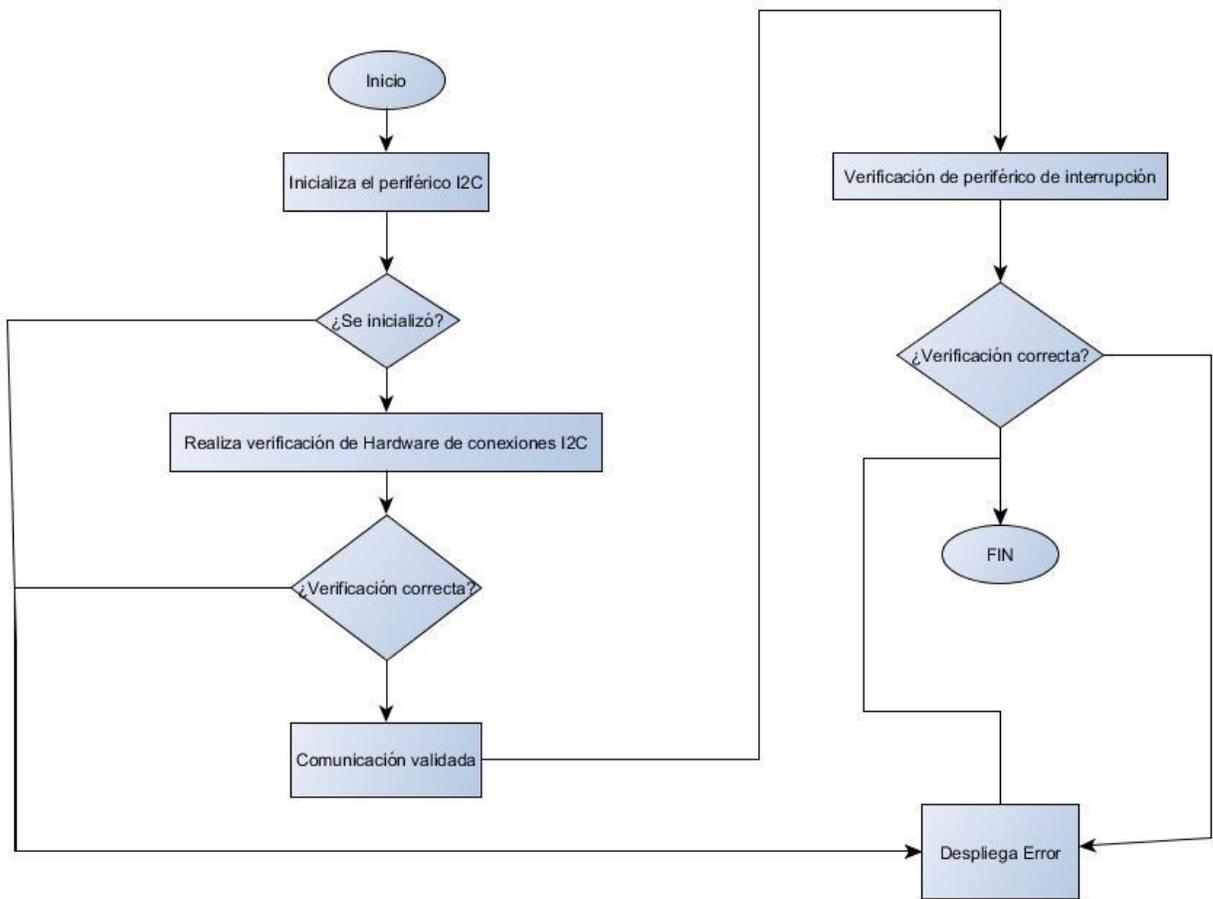


Figura 4.4.3 Programación en SDK para validar módulo I2C

```

#include <stdio.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "iic_header.h"
#include "xtmrctr.h"
#include "tmrctr_header.h"
int main()
{
    Xil_ICacheEnable();
    Xil_DCacheEnable();
    print("---Programa de validacion de modulo I2C---\n\r");

    {
        int status;

        print("\r\n Validando comunicacion con el modulo axi_iic_0...\r\n");

        status = IicSelfTestExample(XPAR_AXI_IIC_0_DEVICE_ID);

        if (status == 0) {
            print("Modulo I2C VALIDADO\r\n");
        }
        else {
            print("IicSelfTestExample FAILED\r\n");
        }
    }
    {
        int status;

        print("\r\n Validando interrupcion por I2C...\r\n");

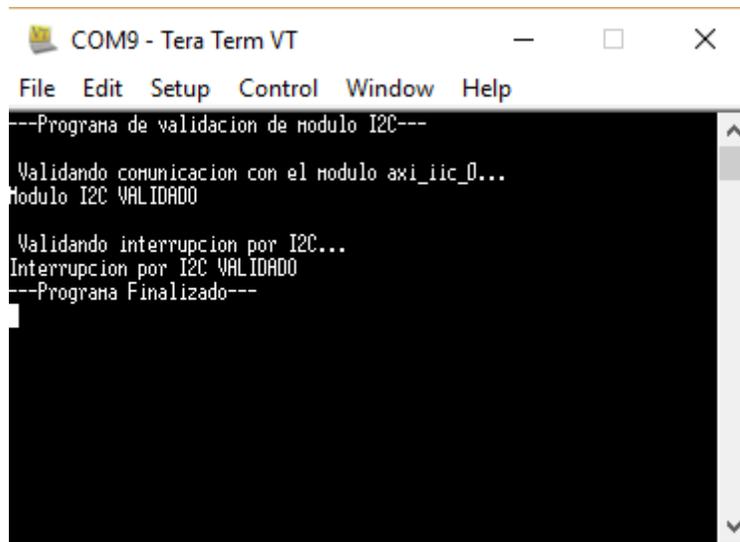
        status = TmrCtrSelfTestExample(XPAR_AXI_TIMER_0_DEVICE_ID, 0x0);

        if (status == 0) {
            print("Interrupcion por I2C VALIDADO\r\n");
        }
        else {
            print("TmrCtrSelfTestExample FAILED\r\n");
        }
    }

    print("---Programa Finalizado---\n\r");
    Xil_DCacheDisable();
    Xil_ICacheDisable();
    return 0;
}

```

Figura 4.4.4 Código de validación módulo AXI IIC



```
COM9 - Tera Term VT
File Edit Setup Control Window Help
---Programa de validacion de modulo I2C---
Validando comunicacion con el modulo axi_iic_0...
Modulo I2C VALIDADO
Validando interrupcion por I2C...
Interrupcion por I2C VALIDADO
---Programa Finalizado---
```

Figura 4.4.5 Despliegue de resultados de la validación en hardware del protocolo I2C

Una vez validado el hardware se validaron los códigos necesarios para el uso con MicroBlaze, para esto se estudió el API del SDK, por lo que se realizó una prueba práctica escribiendo un valor aleatorio en una memoria I2C, la memoria usada es 24FC512 este valor fue 0x23h, escrito en el registro 0 de la memoria, el resultado se observó en el osciloscopio en la figura 4.4.6 se muestra el resultado.

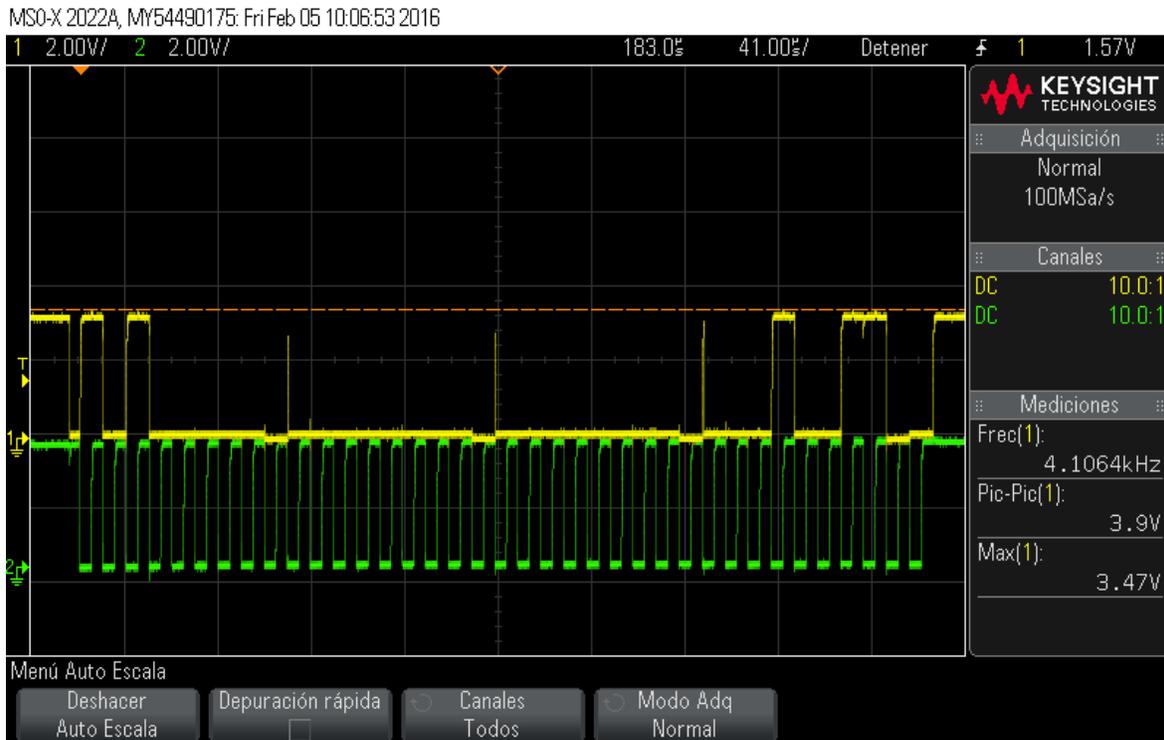


Figura 4.4.6 Validación comunicación I2C en osciloscopio

Se observa en la figura la correcta comunicación por parte del protocolo I2C

## 4.5 Prueba de módulo SPI

Para la implementación y validación del módulo SPI en el FPGA se implementó el bloque IP AXI SPI, como se muestra en la figura 4.5.1, configurándolo a una velocidad de reloj máxima de 60 MHz como se indica en las notas técnicas, para verificar su correcta implementación, por medio de programación en Software en el SDK se validó su conexión de bloques del SPI al soft core ejecutando comandos de SPI, éstos se ilustran en la figura 4.5.2, en la figura 4.5.3 se muestra su código de validación y el resultado de esta implementación fue desplegada por comunicación UART, se observa el resultado de esta implementación en la figura 4.5.4.

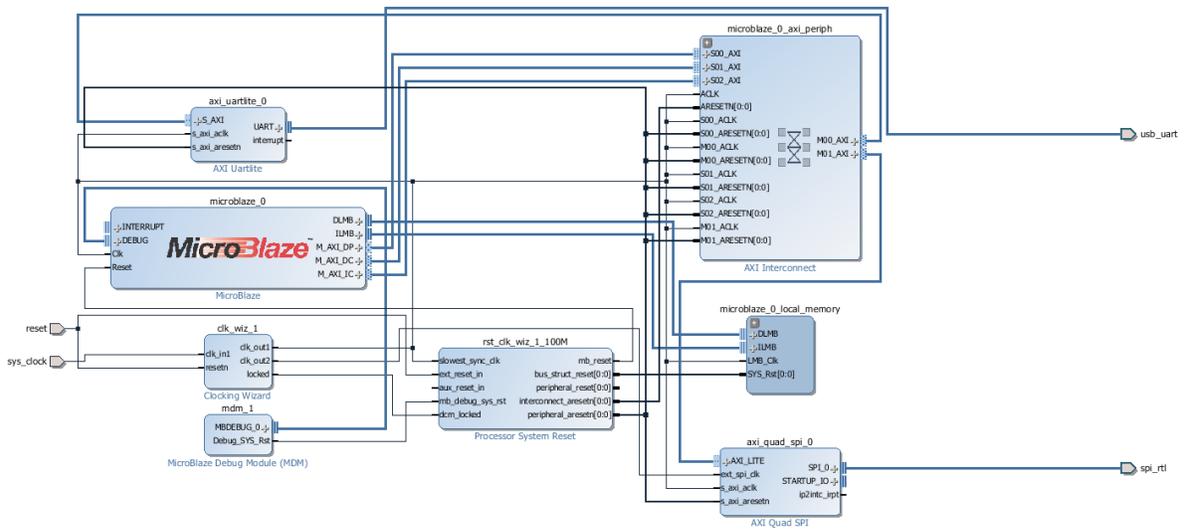


Figura 4.5.1 Implementación de comunicación SPI

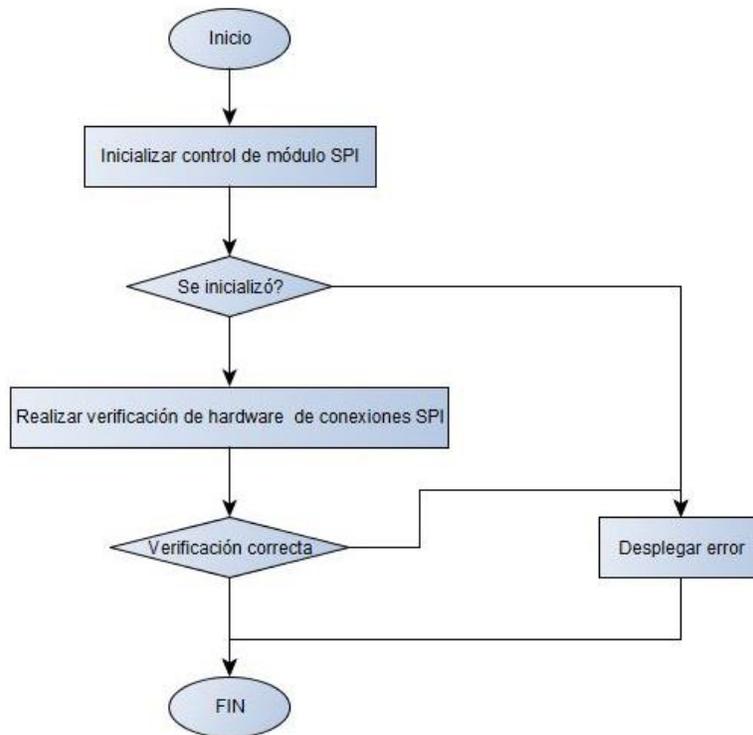


Figura 4.5.2 Programación en SDK para validar módulo SPI

```

#include <stdio.h>
#include "xparameters.h"
#include "xil_cache.h"
#include "xspi.h"
#include "spi_header.h"
int main() {
    Xil_ICacheEnable();
    Xil_DCacheEnable();
    print("---Ejecutando programa---\n\r");
    {
        XStatus status;
        print("\r\n Validando comunicacion SPI en hardware...\r\n");
        status = SpiSelfTestExample(XPAR_AXI_QUAD_SPI_0_DEVICE_ID)
        if (status == 0) {
            print("Estado: Validado\r\n");
        }
        else {
            print("Estado: Error en Hardware\r\n");
        }
    }
    print("---Programa finalizado---\n\r");
    Xil_DCacheDisable();
    Xil_ICacheDisable();
    return 0;
}

```

Figura 4.5.4 Código de validación del módulo SPI

The screenshot shows a terminal window titled "COM4 - Tera Term VT". The menu bar includes "File", "Edit", "Setup", "Control", "Window", and "Help". The terminal output is as follows:

```

---Ejecutando programa---

Validando comunicacion SPI en hardware...
Estado: Validado
---Programa finalizado---

```

Figura 4.5.4 Validación del módulo SPI

## 4.6 Prueba del microcontrolador mediante simulación por software

En esta simulación del microcontrolador por software se basa en la conexión del hardware descrito que se desarrolló e implementó en el FPGA validando sus conexiones y una vez que sea ha validado se simula los datos de los sensores que se tienen pensados implementar mandando su información a través del microcontrolador a un centro de monitoreo, esta función la realizará un PC mediante el despliegue de resultados en LabView. En esta simulación es el equivalente al sistema físico a implementar.

Las ventajas que proporciona esta simulación en Software es:

Permite realizar una simulación previa y en paralelo con el desarrollo del Sistema a implementar, de forma que se pueden ir sustituyendo las partes simuladas por las ya implementadas físicamente conforme estén disponibles.

Permiten reducir el costo de desarrollo, se evitan los prototipos parciales de los elementos del sistema y el costo de verificación, se evitan las averías de un sistema real.

La simulación contempla la correcta implementación de las comunicaciones seriales y un monitoreo de datos, la cual se encarga el microcontrolador de enviar esta información simulada de los transductores al centro de monitoreo que va a ser un PC con LabView. La simulación implementada de los transductores es; monitoreo de un sensor de gas, sensor de temperatura, sensor de nivel de flujo y presión como primera instancia.

El código del sistema de monitoreo del microcontrolador se muestra en la figura 4.6.1, en este código sólo verifica la correcta implementación de los módulos de

comunicación serial mediante las APIs de los IPs y simula datos de los sensores a implementar y los envía por comunicación serial UART, su vista grafica se muestra en la figura 4.6.3.

Al recibir los datos de la comunicación serial del microcontrolador el centro de monitoreo, despliega los datos en una interfaz gráfica la cual va a ser LabView, el código de LabView de monitoreo se muestra en la figura 4.6.2 en este código se obtienen los datos de comunicación UART mediante la conexión del microcontrolador a la computadora usando el bloque NI VISA resource, los datos de la comunicación UART pasan a un ciclo while para leerlos de forma constante, esta lectura se realiza por medio de los bloques visa read los cuales separan toda la información recibida por sus números y caracteres para posteriormente desplegarla de manera visual, en esta simulación sólo se usaron los módulos GPIO y UART del Hardware del FPGA de manera demostrativa para probar el microcontrolador por medio de software, y los módulos I2C y SPI fueron probados con códigos de prueba del fabricante lo cual asegura su funcionalidad.

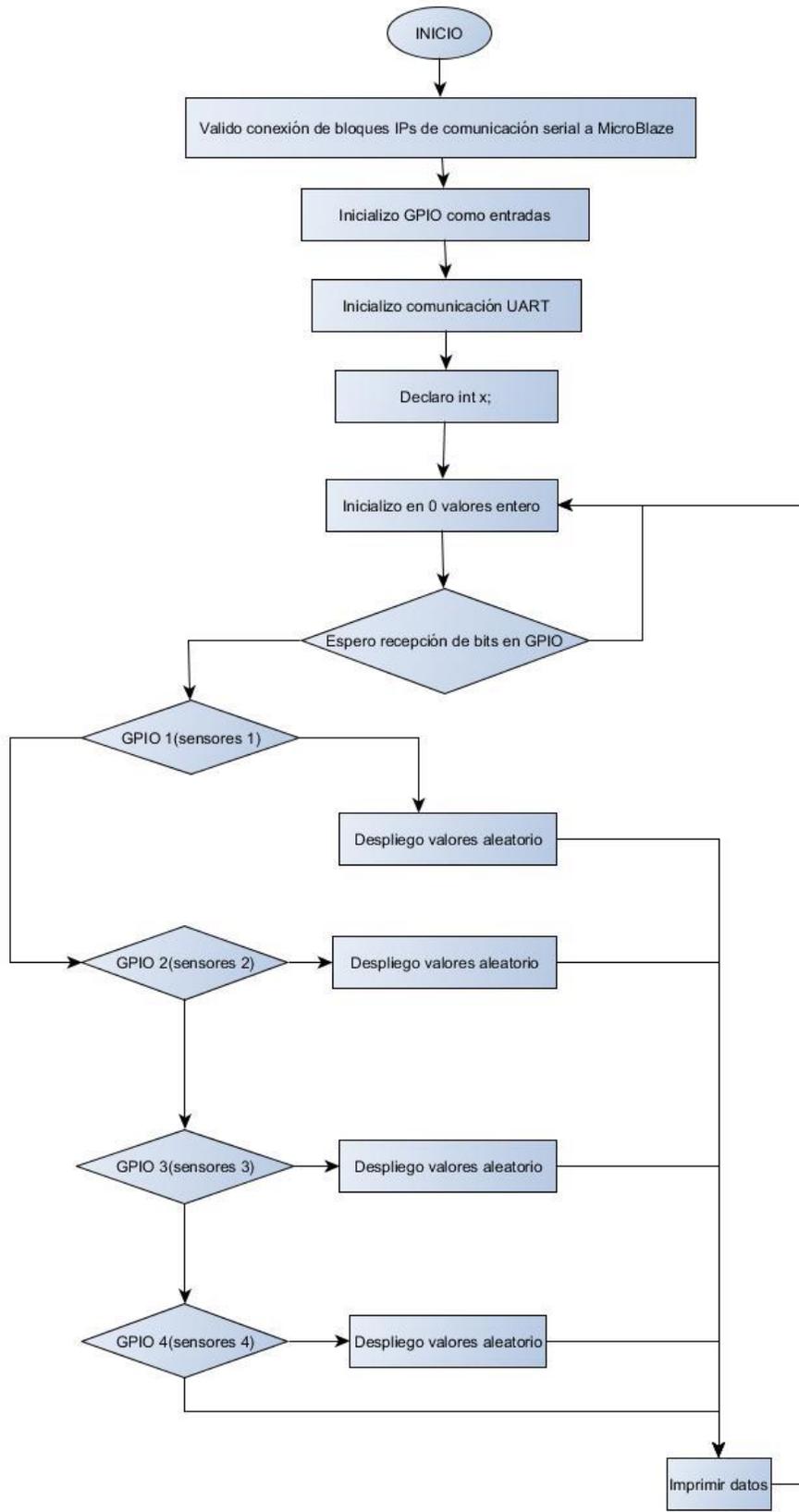


Figura 4.6.1. Código de simulación de monitoreo en el microcontrolador

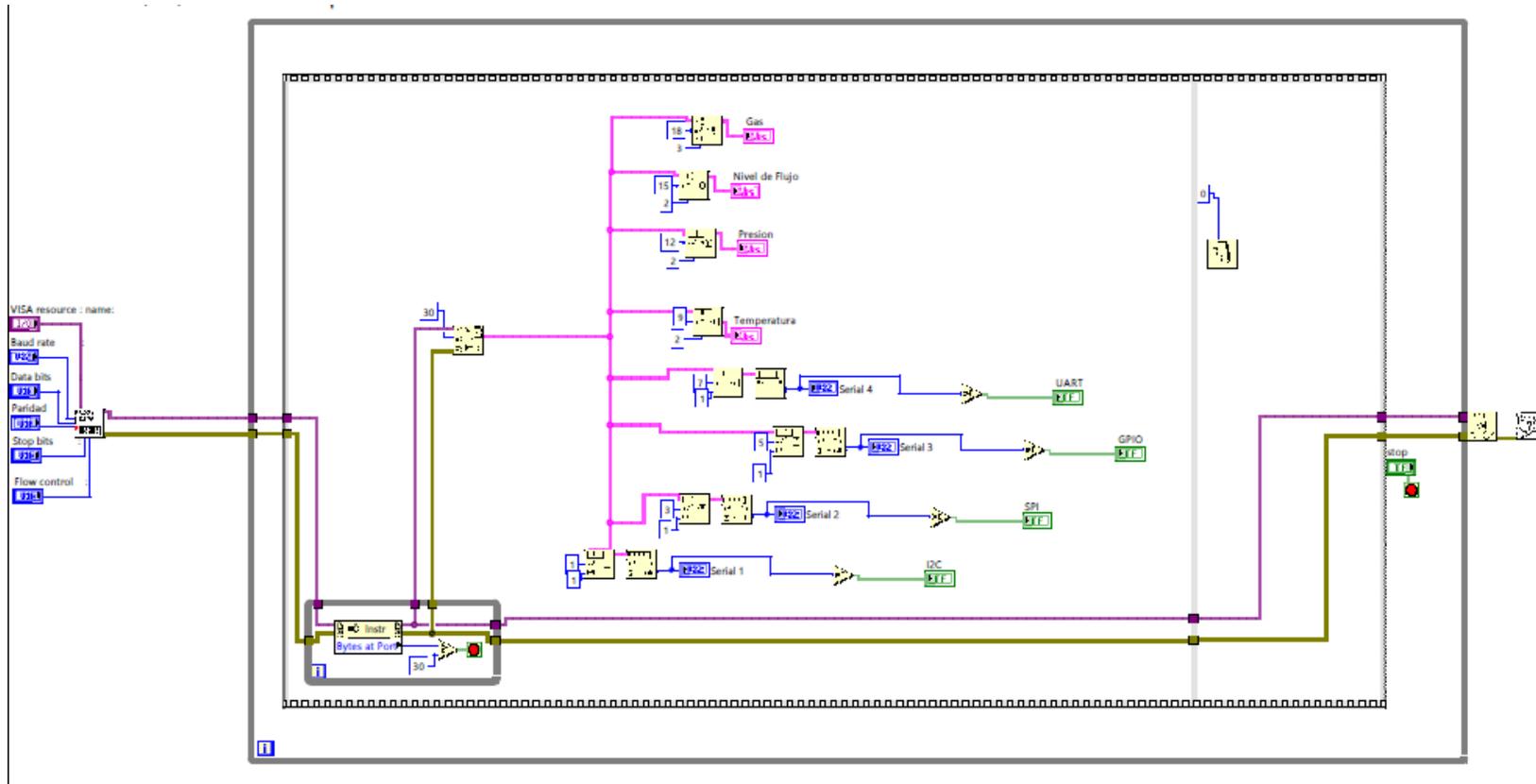


Figura 4.6.2 Código en LabView de recepción de información

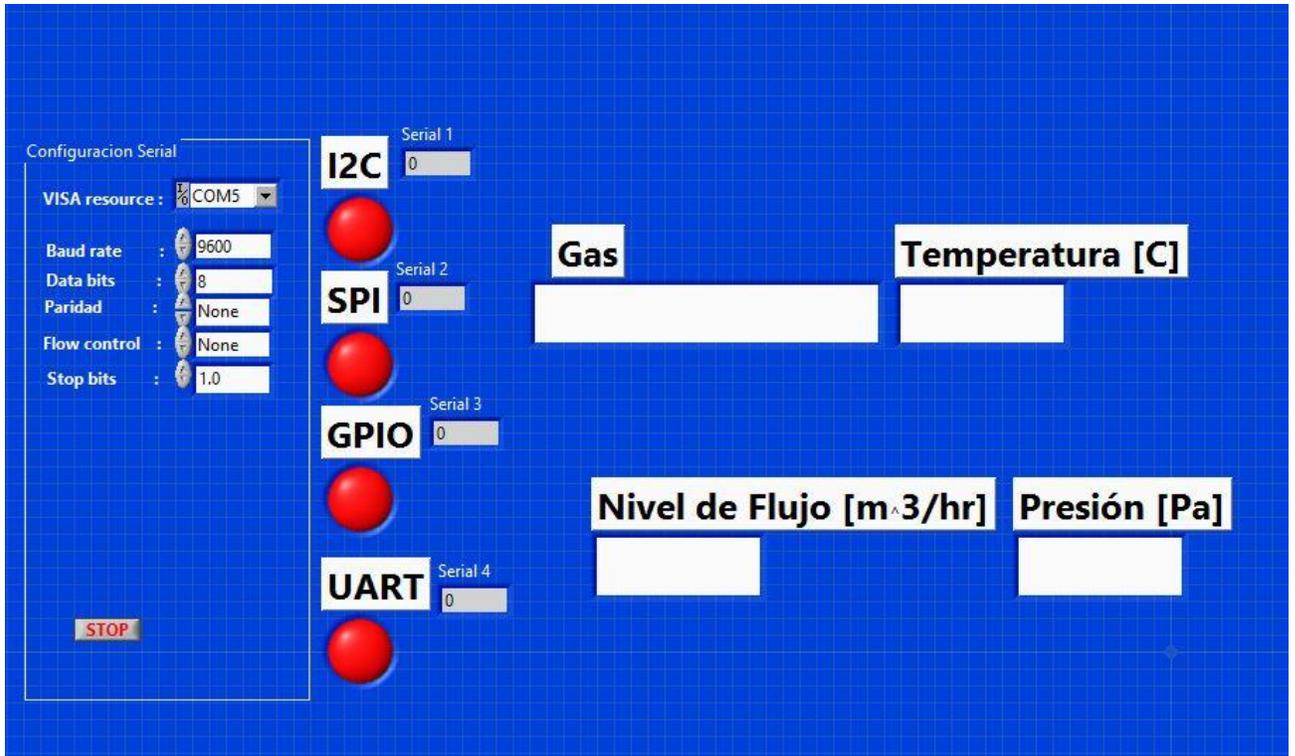


Figura 4.6.3. Despliegue gráfico del centro de monitoreo

## 4.7 Pruebas de Telemetría

Para validar el funcionamiento y configuración de módulos de telemetría en nuestro microcontrolador se pretende usar un módulo transceivers para el envío de datos por comunicación UART a una computadora de monitoreo, para esto se usó con fines de prueba un transceptor modelo HC11 cuyo alcance son 100m.

### Configuración del transceptor

Se usó el dispositivo HC11 Wireless, este dispositivo como características tiene, trabaja en la banda de frecuencia de 434 MHz (UHF), bajo consumo de potencia, vea figura 4.7.1.

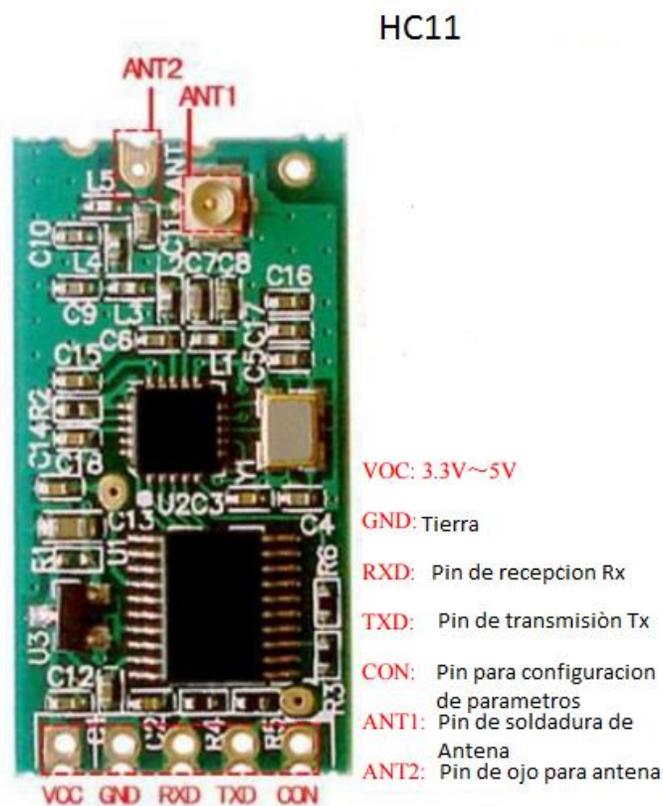


Figura 4.7.1 Características técnicas del HC11

Para poder transmitir datos por telemetría desde nuestro microcontrolador a nuestra computadora que tendría la función de centro de monitoreo, se usaron dos módulos SCMI y un conversor de UART a USB este conversor fue el PL2303, se realizó la conexión de la siguiente forma como se muestra en la figura 4.7.2.

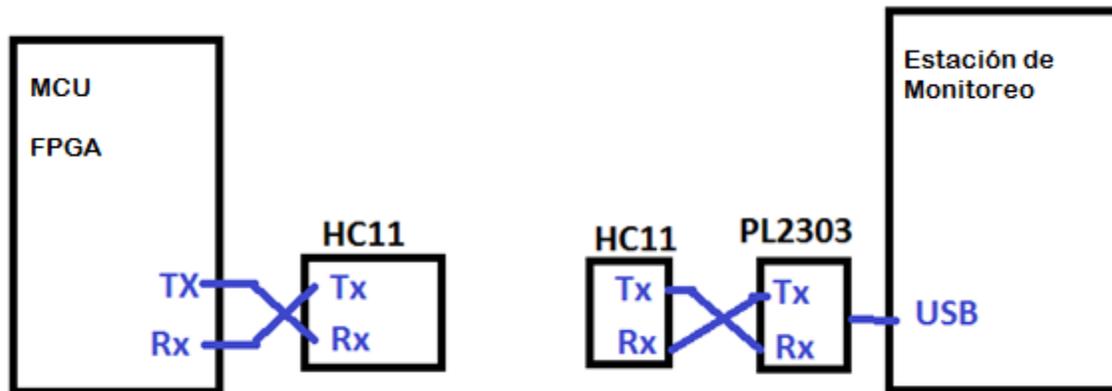


Figura 4.7.2. Conexión de módulos HC11

Los módulos HC11 se utilizan en pares, y transmiten los datos por medio el tipo de comunicación Half-dúplex. La velocidad de transmisión, el canal de comunicación de los módulos debe ser el mismo para poder transmitir y recibir información, en este caso están configurados con un baudrate de 9600.

Tras validar su funcionamiento se realizaron pruebas en donde el microcontrolador envía información hacia la estación de monitoreo con la implementación gráfica de LabView, teniendo resultados exitosos como se observa en la figura 4.7.3.

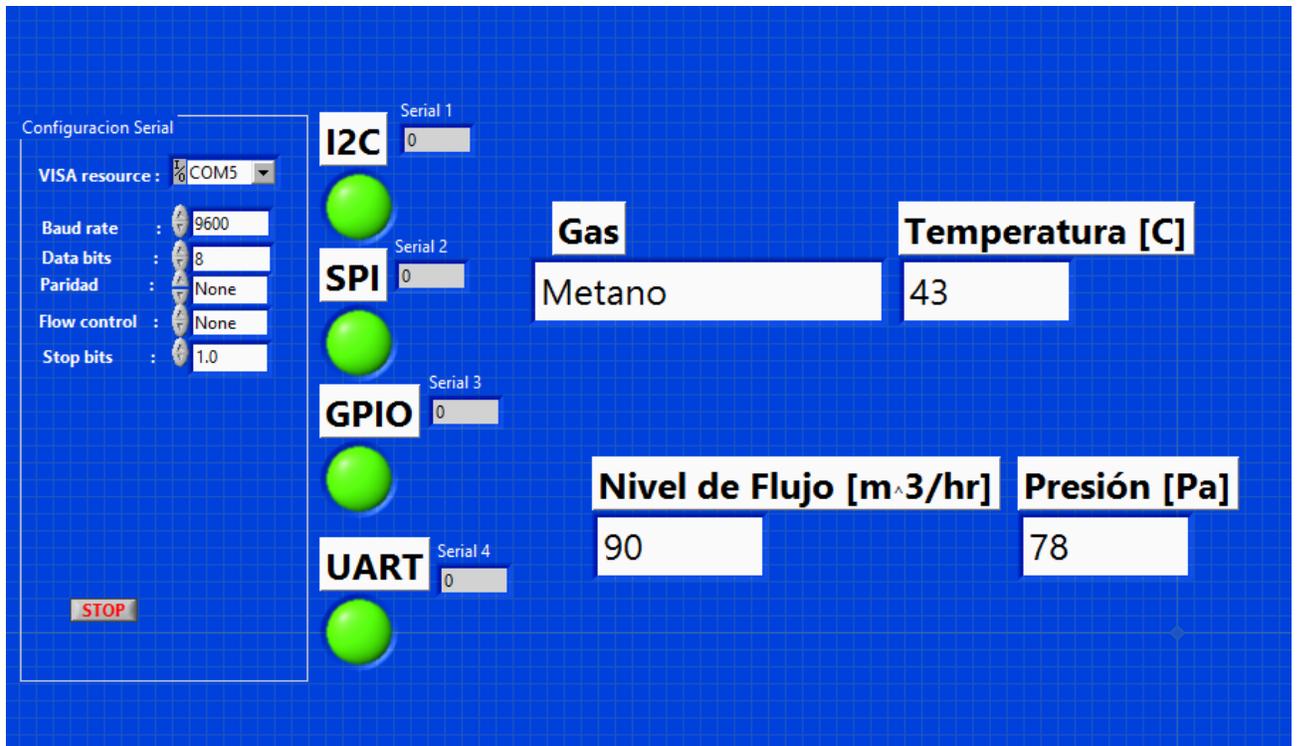


Figura 4.7.3 Funcionamiento de módulos de telemetria

## CONCLUSIONES.

- Los objetivos de esta tesis se cubrieron porque se tiene un microcontrolador con una arquitectura Harvard funcional como se demuestra en las pruebas realizadas.
- El uso de programas de High level syntesis como vivado, nos facilita el desarrollo, diseño e implementación de sistemas complejos diseñados a la medida, en este caso enfocados al área industrial, brindando innovación y mantener una calidad deseada.
- El empleo de FPGA es conveniente para el procesamiento de datos en los sistemas de comando y manejo de información, ya que se aprovecha su característica de paralelismo; además, hay que considerar la flexibilidad ofrecida por éstos para realizar cambios en su funcionalidad cuando se requiera.
- Se logró validar correctamente a nivel hardware y software la implementación de los módulos esenciales de un microcontrolador por medio de IPs, concluyendo que el uso de IPs facilita la elaboración de manera rápida un diseño complejo sin embargo se observó que en ciertos aspectos que se requiere una descripción personalizada y/o funciones específicas a realizar donde se necesita un mantenimiento constante es necesario crear módulos propios en VHDL implementándolos en VIVADO.
- El diseño de este microcontrolador basado en FPGA nos pone en un marco competitivo con respecto a los microcontroladores basados en hardcores debido a que en el FPGA se tiene una implementación a la medida y capaz de ser escalable a cualquier requisito a futuro, además este desarrollo en FPGA nos da una base para implementar futuros proyectos más robustos en el sector industrial.
- Como trabajo a futuro es necesario la creación de un estándar de referencia para microcontroladores, para el desarrollo de proyectos multidisciplinarios y adquisición de protocolos industriales licenciados los cuales destacan; CAN, LVDS, USB los cuales ofrecen una comunicación serial más robusta por lo que hay que tomar en cuenta sobre sus manejo y licencias.

## BIBLIOGRAFIA

- [1] <http://www.atmel.com/products/microcontrollers/avr/default.aspx>, diciembre 2016
- [2] <http://www.atmel.com/products/microcontrollers/arm/default.aspx>, diciembre 2016
- [3] <http://www.atmel.com/applications/industrialautomation/default.aspx>, diciembre 2016
- [4] [http://www.ti.com/lscs/ti/microcontrollers\\_16-bit\\_32-bit/applications.page](http://www.ti.com/lscs/ti/microcontrollers_16-bit_32-bit/applications.page), diciembre 2016
- [5] [http://www.ti.com/lscs/ti/microcontrollers\\_16-bit\\_32-bit/products.page#](http://www.ti.com/lscs/ti/microcontrollers_16-bit_32-bit/products.page#), diciembre 2016
- [6] <http://www.microsemi.com/applications/industrial>, diciembre 2016
- [7] <http://www.xilinx.com/applications/industrial.html>, diciembre 2016
- [8] [https://www.xilinx.com/publications/prod\\_mktg/Industrial-Automation-Solutions-Product-Brief.pdf](https://www.xilinx.com/publications/prod_mktg/Industrial-Automation-Solutions-Product-Brief.pdf), diciembre 2016
- [9] Linda Null; Julia Lobur (2010), The essentials of computer organization and architecture (en inglés) (3ra edición), Jones & Bartlett Learning, pp. 36,199-203
- [10] "VHDL El arte de programar sistemas digitales", David G. Maxinez, editorial continental
- [11] <http://www.ni.com/white-paper/6983/es/>
- [12] C.E. Shannon, "Symbolic Analysis of Relay and Switching Circuits", Transactions AIEE57 (1930), pp. 713-723

- [13] "Fundamentos de lógica digital con diseño vhdl", Brown, 2ed
- [14] BDTI Focus Report: FPGAs for DSP, Second Edition, BDTI Benchmarking, 2006
- [15] Estandar del 1076-2008, "IEEE Standard VHDL Language Reference Manual"
- [16] Estandar del 1364-2001, "IEEE estándar Verilog hardware description language"
- [17] <http://www.dailycircuitry.com/2011/10/zet-soft-core-running-windows-30.html> "Zet soft core running Windows 3.0" by Andrew Felch 2011
- [18] Soft-Core Processors for Embedded Systems, J. G. Tong ; University of Windsor - Department of Electrical and Computer Engineering, Research Centre for Integrated Microsystems, Windsor, Ontario, Canada
- [19] <http://www.embedded.com/columns/showArticle.jhtml?articleID=192700615> "FPGA Architectures from 'A' to 'Z'" by Clive Maxfield 2006
- [20] MicroBlaze Soft Processor: Frequently Asked Questions
- [21] István Vassányi. "Implementing processor arrays on FPGAs". 1998.
- [22] Zhoukun WANG and Omar HAMMAMI. "A 24 Processors System on Chip FPGA Design with Network on Chip".
- [23] John Kent. "Micro16 Array - A Simple CPU Array"
- [24] Kit Eaton. "1,000 Core CPU Achieved: Your Future Desktop Will Be a Supercomputer". 2011.
- [25] "Scientists Squeeze Over 1,000 Cores onto One Chip". 2011.
- [26] "I2C Bus Pullup Resistor Calculation", Application Report, Texas Instruments

[27]KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART), User guide, Texas Instruments p 16

[28] Siegle, F., Vladimirova, T., Ilstad, J., & Emam, O. (2015). Mitigation of radiation effects in SRAM-based FPGAs for space applications. ACM Computing Surveys (CSUR), 47(2), 37.

[29] Vivado Desing Suite User Guide,Getting Started, UG910(v2014.1)

[30] <https://www.xilinx.com/products/design-tools/vivado.html?resultsTablePreSelect=documenttype:SeeAll#documentation>, enero 2017

[31] <http://www.xilinx.com/products/design-tools/microblaze.html>, enero 2017

[32] [https://www.xilinx.com/support/documentation/sw\\_manuals/mb\\_ref\\_guide.pdf](https://www.xilinx.com/support/documentation/sw_manuals/mb_ref_guide.pdf), enero 2017

[33] Embedded Systems: Architecture, Programming and Design,Raj Kamal,McGraw-Hill Education,3ra ed, pp. 20-60