



Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú

UNIVERSITAT POLITÈCNICA DE CATALUNYA

PROYECTO FINAL DE MASTER

**TÍTULO:IMPLEMENTACIÓN DE UN KERNEL LINUX SOBRE UN
PROCESADOR TIPO SOFTWARE UTILIZANDO UNA FPGA**

AUTOR: VAQUERIZO CID, DANIEL

FECHA DE PRESENTACIÓN: Febrero, 2017

APELLIDOS: VAQUERIZO CID

NOMBRE: DANIEL

**TITULACIÓN: MÁSTER UNIVERSITARIO EN SISTEMAS AUTOMÁTICOS Y
ELECTRÓNICA INDUSTRIAL**

PLAN:

DIRECTOR: MARIANO LÓPEZ GARCÍA

DEPARTAMENTO: INGENIERÍA ELECTRÓNICA

CALIFICACIÓN DEL PFM

TRIBUNAL

PRESIDENTE

SECRETARIO

VOCAL

FECHA DE LECTURA: 09/02/2017

Este Proyecto tiene en cuenta aspectos medioambientales: Sí x No

RESUMEN

En el mercado existen multitud de microprocesadores capaz de realizar multitud de tareas muy variadas, sin embargo, todos ellos tienen una limitación, el hardware, ya que éste no puede ser modificado una vez se fabrica la placa.

En este aspecto, las FPGA tienen mucho que decir al respecto, dado que gracias a lo que se conoce como softcore, es posible diseñar un microprocesador en lenguaje VHDL capaz de comportarse de la forma deseada, incluyendo en este diseño los periféricos y utilidades deseadas como memoria caché, unidad de gestión de memoria, coma flotante, etc.

Además, cuando se emplea el término FPGA, normalmente se piensa en un sistema embebido, capaz de ejecutar una aplicación una y otra vez a una velocidad elevadísima, aquí es donde entra en juego el softcore Microblaze, diseñado por Xilinx, ya que permite añadir a una simple FPGA la capacidad de ejecutar un sistema operativo como puede ser Linux.

Este proyecto se centra en el diseño de un softcore capaz de ejecutar un kernel Linux y los periféricos asociados al mismo, pasando para ello por diferentes puntos del diseño, que van desde la creación del procesador propiamente dicho, pasando por la inclusión de periféricos de utilidad, y la creación de un sistema operativo muy básico capaz de ser cargado en la memoria RAM de la FPGA y ejecutado para tener un sistema operativo completamente funcional, que finalmente ejecute una aplicación de control sobre los diferentes periféricos.

Palabras clave (máximo 10):

FPGA	Xilinx	Linux	Microblaze
Control	Softcore	RAM	ISE
Kernel	Microcontrolador		

ABSTRACT

Nowadays the market is full of different solutions in the market of microprocessors, they allow the user to perform a lot of functions and control the different peripherals associated to them, however, they came all with a huge limitation, the hardware could not be modified once the PCB came out of the factory.

This limitation is overpassed by the FPGA chips, the reason is that they allow the use of soft-core, which is an embedded microprocessor capable of being modified and interact with different peripherals which could be added by the user during the design of the hardware in VHDL language. Using a softcore processor allows the user to change the cache memory, peripherals, memory management unit, floating point unit, etc.

Besides, when talking about FPGAs, someone could think about the embedded systems and how they execute an application in a very high speed environment, but, the fact is that they could be used to execute an operating system by integrating the softcore in them, which could be for instance the Microblaze softcore of Xilinx, used in this project.

This project is focused in the design of a full customizable microprocessor and the peripherals to the microprocessor, for doing it, the idea is to use a softcore and add the different necessary units and peripherals in order to be able to execute a minimal operating system using a Linux kernel compiled specially for the designed hardware. This Linux kernel will be executing a control, application that ensures that all the peripherals and the softcore Microblaze works as expected.

Keywords (10 maximum):

FPGA	Xilinx	Linux	Microblaze
Control	Softcore	RAM	ISE
Kernel	Microcontroller		

ÍNDICE

CAPÍTULO 1: INTRODUCCIÓN.....	8
1.1 ESTADO DEL ARTE.....	9
1.2 SISTEMA PROPUESTO.....	15
1.3 OBJETIVOS.....	17
CAPÍTULO 2: DISEÑO DEL SISTEMA MICROPROCESADOR	20
2.1 SOFTWARE EMPLEADO PARA EL DESARROLLO DEL HARDWARE	20
2.2 DISEÑO DE LA PARTE HARDWARE	20
CAPÍTULO 3: DISEÑO DE LA PARTE SOFTWARE.....	38
3.1 SOFTWARE EMPLEADO PARA EL DESARROLLO DEL KERNEL	38
3.2 CREACIÓN DEL KERNEL LINUX PARA MICROBLAZE	38
CAPÍTULO 4: IMPLEMENTACIÓN Y TEST DEL DISEÑO REALIZADO.....	50
4.1 IMPLEMENTACIÓN DEL DISEÑO.....	50
4.2 TEST DE LA PARTE HARDWARE Y SOFTWARE.....	38
CAPÍTULO 5: DISEÑO DE LA APLICACIÓN LINUX.....	66
5.1 APLICACIÓN PROPUESTA.....	66
5.2 DISEÑO DE LA APLICACIÓN	67
5.3 TEST DE LA APLICACIÓN	76
CAPÍTULO 6: CONCLUSIONES.....	86
6.1 ANÁLISIS DE OBJETIVOS	86
6.2 CONCLUSIONES.....	86
6.3 LÍNEAS DE MEJORA.....	87
BIBLIOGRAFÍA.....	89
AGRADECIMIENTOS	90
ANEXO.....	91

SUMARIO DE FIGURAS

FIGURA 1: MARKET SHARE FABRICANTES FPGAS FINAL 2015 (FUENTE: IHS)	9
FIGURA 2: DIAGRAMA DE BLOQUES DEL SOFTCORE NIOS II	10
FIGURA 3: DIAGRAMA DE BLOQUES DEL SOFTCORE MICO32	11
FIGURA 4: DIAGRAMA DE BLOQUES DE SOFTCORE RV32IM	12
FIGURA 5: DIAGRAMA DE BLOQUES DE MICROBLAZE	13
FIGURA 6: CAPAS DEL SISTEMA PROPUESTO	15
FIGURA 7: SOFTCORE MICROBLAZE	15
FIGURA 8: DIAGRAMA HARDWARE DEL SISTEMA PROPUESTO	16
FIGURA 9: HERRAMIENTA XILINX XPS	20
FIGURA 10: XPS - SELECCIÓN DEL TIPO DE BUS DEL SISTEMA	21
FIGURA 11: XPS - SELECCIÓN DE LA PLACA DE DESARROLLO	21
FIGURA 12: XPS - CONFIGURACIÓN INICIAL DEL SISTEMA PROCESADOR	22
FIGURA 13: XPS - SELECCIÓN INICIAL DE PERIFÉRICOS	22
FIGURA 14: XPS - CONFIGURACIÓN MEMORIA CACHÉ	23
FIGURA 15: XPS - PANEL DE CONFIGURACIÓN DE MICROBLAZE	23
FIGURA 16: XPS - MICROBLAZE CONFIGURACIÓN GENERAL	24
FIGURA 17: XPS - MICROBLAZE CONFIGURACIÓN DE EXCEPCIONES	24
FIGURA 18: XPS - MICROBLAZE CONFIGURACIÓN DE LA MMU	25
FIGURA 19: XPS - MICROBLAZE CONFIGURACIÓN DEL PVR	25
FIGURA 20: XPS - PERIFÉRICOS INICIALES MICROBLAZE	26
FIGURA 21: XPS - IPS DISPONIBLES PARA MICROBLAZE	26
FIGURA 22: XPS - PERIFÉRICOS FINALES MICROBLAZE	26
FIGURA 23: XPS - DIRECCIÓN DE MEMORIA DE LOS ELEMENTOS DEL SOFTCORE	27
FIGURA 24: FICHERO DE DESCRIPCIÓN .MHS - PUERTOS	27
FIGURA 25: FICHERO DE DESCRIPCIÓN .MHS – MICROBLAZE Y BUSES	28
FIGURA 26: FICHERO DE DESCRIPCIÓN .MHS – PERIFÉRICOS ASOCIADOS A MICROBLAZE	29
FIGURA 27: FICHERO DE DESCRIPCIÓN .MHS – PERIFÉRICOS ASOCIADOS A MICROBLAZE II	30
FIGURA 28: FICHERO DE DESCRIPCIÓN .MHS – ELEMENTOS DE MICROBLAZE	31
FIGURA 29: PLACA DE DESARROLLO AVNET VIRTEX-5 LX50	32
FIGURA 30: FICHERO DE RESTRICCIONES .UCF – GPIOs DE ELEMENTOS SOLDADOS EN LA PLACA	33
FIGURA 31: FICHERO DE RESTRICCIONES .UCF – GPIO Y UART DEL USUARIO	33
FIGURA 32: FICHERO DE RESTRICCIONES .UCF – ETHERNET, RAM, RESET Y CLOCK	34
FIGURA 33: XPS - EXPORT DEL DISEÑO A SDK	35
FIGURA 34: XPS - ERRORES EN EL EXPORT DEL DISEÑO A SDK	35
FIGURA 35: XPS - FINAL DE COMPILACIÓN DEL DISEÑO	35
FIGURA 36: XPS - RESUMEN DE RECURSOS EMPLEADOS DE LA FPGA	36
FIGURA 37: SDK - RESUMEN DEL HARDWARE A IMPLEMENTAR	36
FIGURA 38: KERNEL LINUX 3.8-R1 - MENÚ DE CONFIGURACIÓN	38
FIGURA 39: SDK - BSPS ANTES/DESPUÉS AÑADIR EL GENERADOR DE DEVICE-TREE	39
FIGURA 40: SDK - CONFIGURACIÓN DEL BSP GENERADOR DE DEVICE-TREE	40
FIGURA 41: SDK - OBTENCIÓN DEL FICHERO DE DESCRIPCIÓN .DTS	40
FIGURA 42: FICHERO DE DESCRIPCIÓN HW .DTS - DESCRIPCIÓN DE MICROBLAZE	41
FIGURA 43: FICHERO DE DESCRIPCIÓN HW .DTS - DESCRIPCIÓN DE PERIFÉRICOS I	42
FIGURA 44: FICHERO DE DESCRIPCIÓN HW .DTS - DESCRIPCIÓN DE PERIFÉRICOS II	43
FIGURA 45: SISTEMA DE FICHEROS PARA LINUX (XILINX WIKI)	44
FIGURA 46: KERNEL 3.8-R1 - MENÚ DE CONFIGURACIÓN	45
FIGURA 47: KERNEL 3.8-R1 - AJUSTE DEL SISTEMA DE FICHEROS	46
FIGURA 48: KERNEL 3.8-R1 - AJUSTES DE LA PLATAFORMA EMPLEADA	46
FIGURA 49: KERNEL 3.8-R1 - AJUSTE DEL SOPORTE PARA RED Y ETHERNET	47
FIGURA 50: KERNEL 3.8-R1 – AJUSTE DE LOS DRIVERS ETHERNET	47
FIGURA 51: KERNEL 3.8-R1 – AJUSTE DEL SOPORTE PARA GPIOs	48
FIGURA 52: KERNEL 3.8-R1 – FINAL DE COMPILACIÓN	48
FIGURA 53: SDK - PROGRAMACIÓN DE LA FPGA	50
FIGURA 54: XMD - CONEXIÓN AL MÓDULO DE DEBUG	51
FIGURA 55: XMD - DESCARGA DEL KERNEL SOBRE LA RAM	51

FIGURA 56: TERMINAL LINUX – CONSOLA DE ARRANQUE	52
FIGURA 57: TERMINAL LINUX - GPIOs USER SPACE	53
FIGURA 58: TERMINAL LINUX - OBTENCIÓN DE INFORMACIÓN DEL GPIOCHIP252	53
FIGURA 59: TERMINAL LINUX - CONFIGURACIÓN Y TEST DE LOS SWITCH BUTTONS	54
FIGURA 60: TERMINAL LINUX - OBTENCIÓN DE INFORMACIÓN DEL GPIOCHIP245	54
FIGURA 61: TERMINAL LINUX - CONFIGURACIÓN Y TEST DE LOS PUSH BUTTONS	55
FIGURA 62: TERMINAL LINUX - OBTENCIÓN DE INFORMACIÓN DEL GPIOCHIP248	55
FIGURA 63: CONFIGURACIÓN Y TEST DE LOS LEDS	56
FIGURA 64: PLACA DE DESARROLLO - TEST DEL GPIO DE LOS LEDS	56
FIGURA 65: OBTENCIÓN DE INFORMACIÓN DEL GPIOCHIP241	56
FIGURA 66: CONFIGURACIÓN Y TEST DE LOS GPIOs	57
FIGURA 67: PLACA DE DESARROLLO - TEST DEL GPIO ASIGNADOS A JP11	57
FIGURA 68: TERMINAL LINUX - HERRAMIENTAS DE BUSYBOX	58
FIGURA 69: TERMINAL LINUX – INFORMACIÓN DEL PROCESADOR	59
FIGURA 70: TERMINAL LINUX – INFORMACIÓN DE LA MEMORIA	59
FIGURA 71: TERMINAL LINUX – INFORMACIÓN DEL KERNEL	60
FIGURA 72: TERMINAL LINUX – INTERFACES DE RED	60
FIGURA 73: TERMINAL LINUX – CONFIGURACIÓN DE ETH0	60
FIGURA 74: TERMINAL LINUX – PING HACIA EL ROUTER	61
FIGURA 75: TERMINAL CMD WINDOWS – DIRECCIÓN IP	61
FIGURA 76: TERMINAL LINUX – PING HACIA EL ORDENADOR WINDOWS	61
FIGURA 77: TERMINAL LINUX – ARP DE LA IP DEL ROUTER	62
FIGURA 78: TERMINAL CMD WINDOWS – PING HACIA LA FPGA	62
FIGURA 79: TERMINAL LINUX – PARÁMETROS DEL DISPOSITIVO TTYUL1	62
FIGURA 80: TERMINAL LINUX - CONFIGURACIÓN DEL DISPOSITIVO TTYUL1	63
FIGURA 81: TERMINAL LINUX – VI READ.SH	63
FIGURA 82: TERMINAL LINUX – EJECUCIÓN DEL SCRIPT READ.SH	63
FIGURA 83: FLUJOGRAMA DE FUNCIONAMIENTO DE LA APLICACIÓN LINUX	68
FIGURA 84: SENSORES - AM2302	69
FIGURA 85: SENSORES - HC-SR501	69
FIGURA 86: SENSORES - FINAL DE CARRERA MECÁNICO	69
FIGURA 87: SENSORES - LDR (LIGHT DEPENDENT RESISTOR)	69
FIGURA 88: ACTUADORES - HL-52 V1.0	70
FIGURA 89: ARDUINO NANO	70
FIGURA 90: AVNET LX50 VIRTEX5	70
FIGURA 91: REPETIDOR WIFI NETGEAR EX2700	70
FIGURA 92: ESQUEMA DE SENSORES Y ACTUADORES	71
FIGURA 93: APLICACIÓN EN ARDUINO – FRAGMENTO DE ENVÍO DE LA TEMPERATURA ACTUAL	72
FIGURA 94: APLICACIÓN WEB – FRAGMENTO DE ARRANQUE Y MUESTRA DE VALORES	73
FIGURA 95: APLICACIÓN WEB – FRAGMENTO DE COMPARACIÓN	73
FIGURA 96: APLICACIÓN WEB – FRAGMENTO DE CONFIGURACIÓN DE LA CALEFACCIÓN	74
FIGURA 97: APLICACIÓN WEB – FRAGMENTO DE ALMACENAJE DE PARÁMETROS DEL USUARIO	74
FIGURA 98: ESTADO INICIAL DE LOS RELÉS	76
FIGURA 99: WEB - CALEFACCIÓN MANUAL ON	77
FIGURA 100: WEB - CALEFACCIÓN AUTOMÁTICA, T AMBIENTE SUPERIOR A DESEADA	77
FIGURA 101: WEB - CALEFACCIÓN AUTOMÁTICA, T AMBIENTE INFERIOR A DESEADA	78
FIGURA 102: WEB - CALEFACCIÓN AUTOMÁTICA, T AMBIENTE SUPERIOR A DESEADA (II)	78
FIGURA 103: WEB - PERSIANAS EN AUTOMÁTICO ARRIBA, DE DÍA	79
FIGURA 104: WEB - PERSIANAS EN AUTOMÁTICO ARRIBA, DE NOCHE	79
FIGURA 105: WEB – PERSIANAS EN AUTOMÁTICO ABAJO, DE NOCHE	80
FIGURA 106: WEB – PERSIANAS EN AUTOMÁTICO ABAJO, DE DÍA	80
FIGURA 107: WEB - PERSIANAS MODO MANUAL ABAJO, PERSIANA SUBIDA	81
FIGURA 108: WEB - PERSIANAS MODO MANUAL ABAJO, PERSIANA BAJADA	81
FIGURA 109: WEB - PERSIANAS MODO MANUAL ARRIBA, PERSIANA BAJADA	82
FIGURA 110: WEB - PERSIANAS MODO MANUAL ARRIBA, PERSIANA SUBIDA	82
FIGURA 111: WEB – LUCES ACTIVADAS	83
FIGURA 112: WEB – LUCES DESACTIVADAS	83

GLOSSARIO DE SIGNOS, SÍMBOLOS, ABREVIACIONES, ACRÓNIMOS Y TÉRMINOS

FPGA	<i>Field Programmable Gate Array</i>
ALU	<i>Arithmetic Logic Unit</i>
AXI	<i>Advanced eXtensible Interface</i>
PLB	<i>Processor Local Bus</i>
MMU	<i>Memory Management Unit</i>
RAM	<i>Random Access Memory</i>
IP	<i>Intellectual Property</i>
RISC	<i>Reduced Instruction Set Computer</i>
VHDL	<i>Very High Speed Integrated Circuit Hardware Description Language</i>
UART	<i>Universal Asynchronous Receiver-Transmitter</i>
JTAG	<i>Join Test Action Group</i>
SoC	<i>System on Chip</i>
GPIO	<i>General Purpose Input Output</i>
XMD	<i>Xilinx Microprocessor Debugger</i>

CAPÍTULO 1

INTRODUCCIÓN

En el mundo de los microprocesadores existen muchos dispositivos diferentes, cada uno de ellos cuenta con multitud de periféricos, entre estos periféricos existen varios tipos, para comunicación, cálculo en coma flotante, operaciones lógicas, gestión de memoria, etc.

Algunos de estos microprocesadores cuentan con un número mayor de bytes para almacenar código y variables, entre otras cosas, sin embargo, ninguno de ellos permite modificar el hardware con el que vienen de fábrica. Esto es, añadir y eliminar comunicación serie, aumentar la memoria destinada a código, añadir un controlador para GPIO, e incluso modificar la velocidad a la que ejecutan el código. Y aquí es donde gana ventaja un microprocesador softcore, ya que permite añadir y eliminar periféricos, así como modificar multitud de parámetros que van directamente asociados con el hardware empleado en el microprocesador para adaptarlo a la aplicación concreta para la que esté siendo diseñado. Además, según la capacidad de la FPGA es posible implementar diferentes microprocesadores pudiendo asignar a cada uno las tareas y periféricos que se deseen.

Este proyecto pretende dar a conocer las posibilidades de desarrollo que permite el empleo de una FPGA a la hora de implementar diferentes soluciones, diseñando para ello un procesador tipo softcore. Se tratan los puntos básicos del diseño del procesador para que éste pueda ser implementado sobre una FPGA e incluya el hardware necesario para permitir la ejecución de un kernel Linux. Para ejecutar el sistema operativo Linux sobre el softcore es necesaria la creación de un kernel personalizado que sea compatible con los diferentes periféricos que se van a incluir en el diseño del sistema microprocesador.

En este primer capítulo se muestra el estado actual en cuanto a los diferentes procesadores softcore disponibles, los diferentes tipos de FPGA compatibles existentes y las capacidades de cada uno de ellos.

Seguidamente se presenta la idea que se pretende desarrollar en este trabajo y posteriormente cuales son los objetivos del proyecto y la forma en la que se pretenden alcanzar.

1.1 ESTADO DEL ARTE

Las alternativas a los microprocesadores hardware son los anteriormente mencionados softcores, éstos son sistemas microprocesadores basados en una descripción hardware de alto nivel, ya sea VHDL o Verilog. Dicho código puede ser implementado en una FPGA.

De éste modo un softcore no es más que un código muy complejo que al ejecutarse a una velocidad altísima en la FPGA es capaz de realizar las tareas propias de un microprocesador, con la ventaja principal de ser completamente configurable.

Dado que los softcore deben ejecutarse en una FPGA, a continuación, se analizan las principales empresas que se encargan de diseñar y producir sus propios chips. De acuerdo con la siguiente figura, que muestra los datos financieros del año 2015, Xilinx sería la empresa que mayor presencia tiene en el mercado, seguida de Altera y el resto de fabricantes.

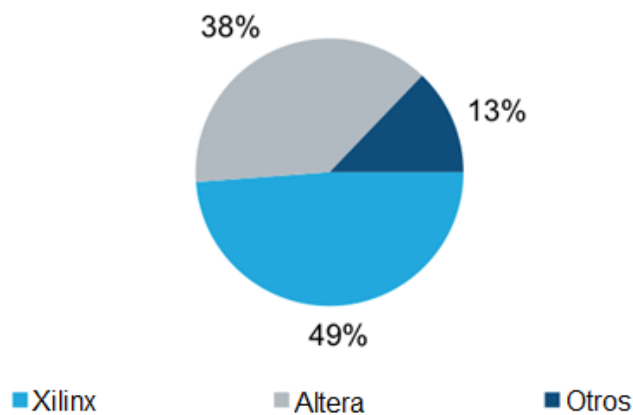


Figura 1: Market Share Fabricantes FPGAs final 2015 (Fuente: IHS)

Por tanto, el mercado de las FPGAs está controlado principalmente por 5 fabricantes. Entre estos 5, la mayoría cuentan como mínimo con un sistema microprocesador tipo software, compatible con sus placas de desarrollo principales. Los softcores más actuales se enumeran a continuación según quién se ha encargado de su desarrollo. Además, vale la pena destacar que Quick Logic no cuenta con softcore, a diferencia de sus competidores.

	Altera	Lattice Semi	Microsemi	Quick Logic	Xilinx
¿Softcore disponible?	SI	SI	SI	NO	SI
Softcore	NIOS II	Mico32	RV32IM	-	Microblaze

Tabla 1: Softcores según desarrollador

A continuación, se analizan las capacidades de los procesadores software de cada uno de los fabricantes anteriormente mencionados.

1.1.1 NIOS II

El procesador NIOS II, creado por Altera, es capaz de manejar instrucciones de hasta 32-bits, maneja instrucciones RISC, y está disponible en tres variantes, según el propósito del diseño.

Tal y como muestra la Figura 2 el microprocesador NIOS II, cuenta con una serie de elementos y periféricos tales como la unidad aritmético lógica, módulo de debug, etc.

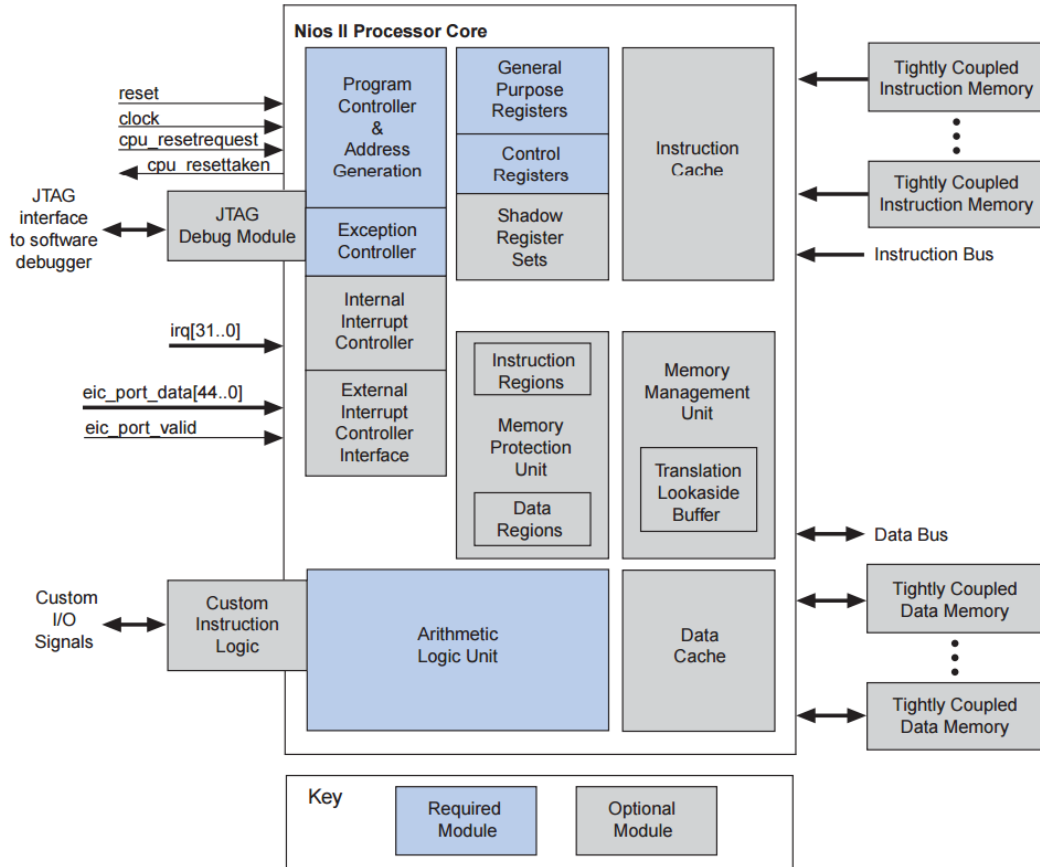


Figura 2: Diagrama de bloques del softcore NIOS II

La siguiente tabla permite resumir las especificaciones generales de cada una de las variantes, de forma que se posteriormente se pueda comparar con el resto de softcores.

NIOS II	Fast	Standard	Economy
Caché Instrucciones	SI	SI	NO
Caché Datos	SI	NO	NO
MMU	Opcional	NO	NO
Memoria externa	SI	SI	SI
Pipeline	6	5	1
Barrel Shifter	1 ciclo	Opcional	NO
Multiplicador HW	1 ciclo	Opcional	NO
Divisor HW	Opcional	Opcional	NO
Módulo Debug JTAG	SI	Opcional	NO
Soporte multi Timer	SI	NO	NO

Tabla 2: Variantes del NIOS II

Tal y como muestra la Tabla 2, de las tres variantes disponibles el único procesador con un pipeline de 6 y con capacidad para incluir una unidad de gestión de memoria, así como caché de datos e instrucciones es la variante Fast. El resto de variantes disponen de unas características más limitadas y un número menor de opciones a escoger.

Una vez analizadas las peculiaridades del NIOS II se pasa al análisis del microprocesador de Lattice Semiconductor.

1.1.2 MICO32

Por su parte, Lattice Semiconductor cuenta con un softcore al que han denominado Mico32, que tal y como indica su nombre trabaja con hasta 32 bits. Por otra parte, la Figura 3 se encarga de mostrar el “interior” del microprocesador, donde podemos encontrar la unidad de barrido lógico, multiplicador y divisor, etc.

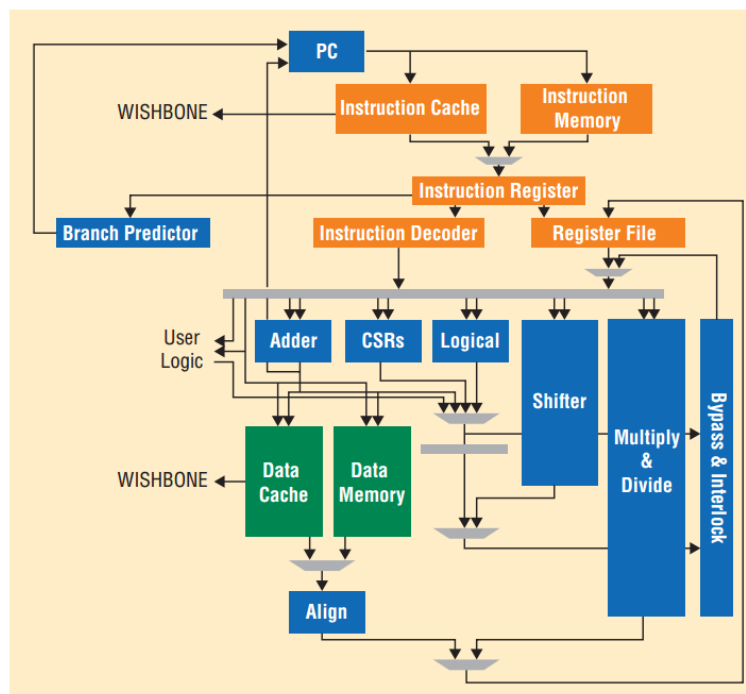


Figura 3: Diagrama de bloques del softcore Mico32

Igual que softcore de Altera, el Mico32 está disponible en tres variantes, la variante Full es la más completa, mientras que las variantes Standard y Basic ven reducidas sus capacidades.

Mico32	Full	Standard	Basic
Caché Instrucciones	SI	SI	NO
Caché Datos	SI	NO	NO
MMU	NO	NO	NO
Memoria externa	SI	SI	SI
Pipeline	6	5	1
Barrel Shifter	Opcional	NO	NO
Multiplicador HW	Opcional	NO	NO

Divisor HW	Opcional	NO	NO
Módulo Debug JTAG	NO	NO	NO
Soporte multi Timer	SI	NO	NO

Tabla 3: Variantes del Mico32

El Mico32 es capaz de igualar las características del NIOSII, sin embargo, éste no cuenta con la posibilidad de incluir una unidad de gestión de memoria.

Tras analizar las opciones que ofrece Altera y Lattice Semi, se procede al análisis del RV32IM, el softcore de la empresa Microsemi.

1.1.3 RV32IM

Por su parte, la empresa Microsemi cuenta con un microprocesador soft, de 32 bits, igual que sus competidores, el RV32IM es también capaz de trabajar con instrucciones RISC y cuenta con los elementos mostrados en la siguiente figura, la Figura 4.

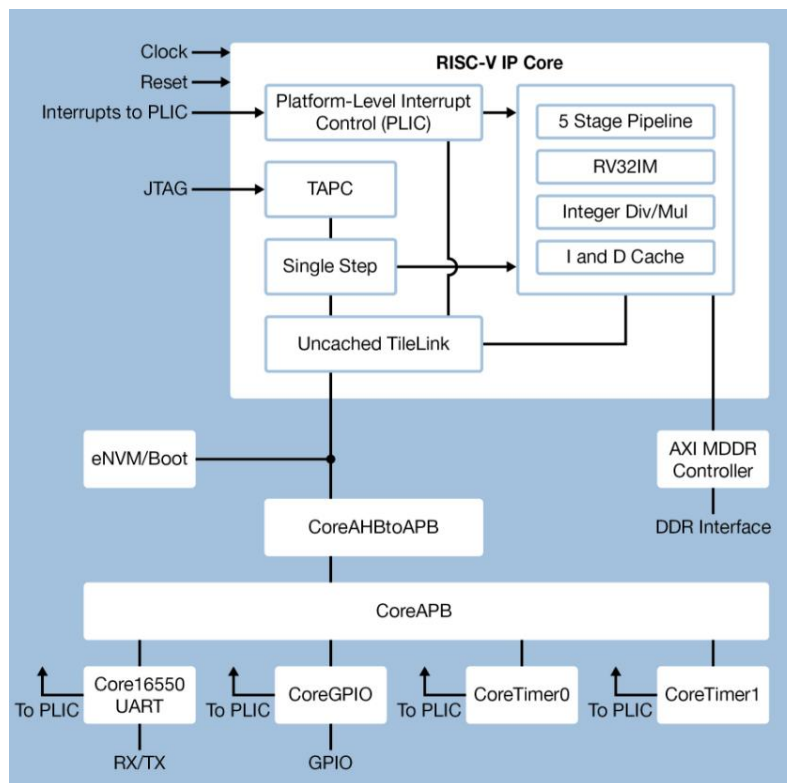


Figura 4: Diagrama de bloques de software RV32IM

Analizando el diagrama de bloques que proporciona la empresa Microsemi, se pueden extraer las siguientes características.

RV32IM	
Caché Instrucciones	SI
Caché Datos	SI
MMU	NO
Memoria externa	SI
Pipeline	5

Barrel Shifter	NO
Multiplicador HW	SI
Divisor HW	SI
Módulo Debug JTAG	SI
Soporte multi Timer	SI

Tabla 4: Características del RV32IM

En este caso, el microprocesador de Microsemi, es similar en características al de Lattice Semi, ya que no cuenta con la posibilidad de emplear una unidad de gestión de memoria, aunque si incluye multiplicador y divisor, así como caché de datos e instrucciones.

Por último, queda el microprocesador de Xilinx, que se analiza a continuación.

1.1.4 Microblaze

En este punto, solamente queda por analizar el softcore de Xilinx, cuyo nombre es Microblaze. Microblaze es un microprocesador de 32 bits con capacidad para interpretar instrucciones RISC y con un mayor número de opciones que sus rivales. La XX muestra su diagrama de bloques, donde se pueden observar las posibles configuraciones de éste.

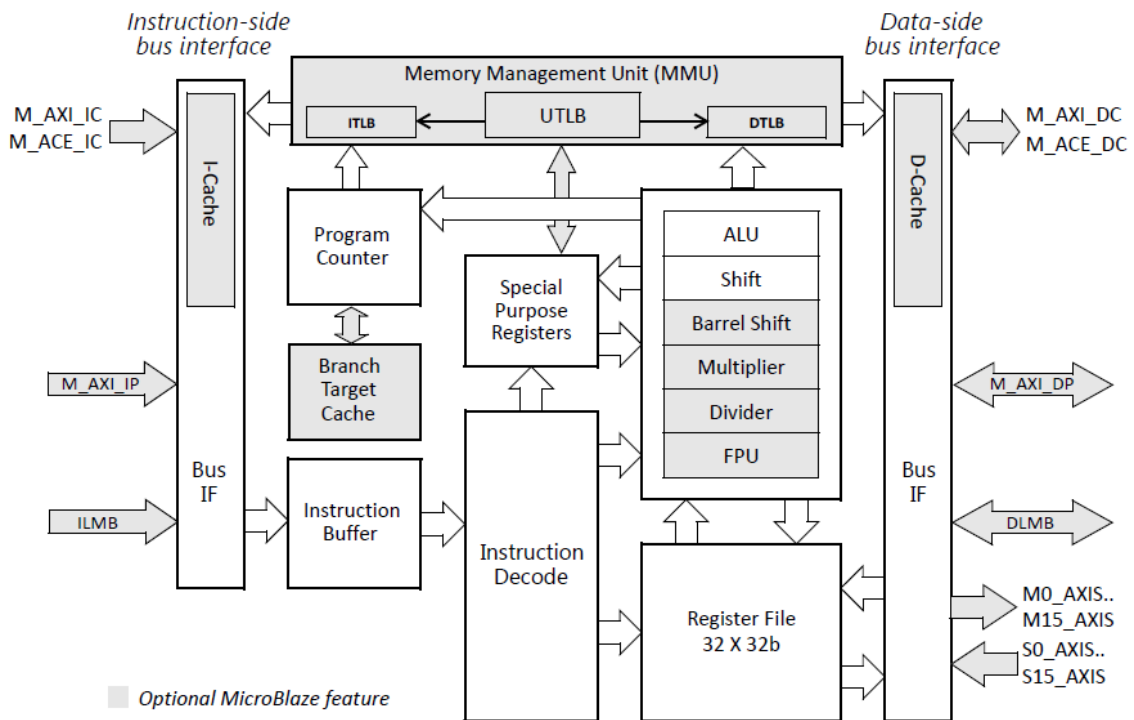


Figura 5: Diagrama de bloques de Microblaze

En el anterior diagrama se pueden observar las posibilidades de configuración de Microblaze, teniendo en cuenta sus elementos opcionales, el resumen es el siguiente.

Microblaze	
Caché Instrucciones	Opcional
Caché Datos	Opcional

MMU	Opcional
Memoria externa	SI
Pipeline	5/3
Barrel Shifter	Opcional
Multiplicador HW	Opcional
Divisor HW	Opcional
Módulo Debug JTAG	SI
Soporte multi Timer	SI

Tabla 5: Características de Microblaze

Microblaze, a diferencia de sus competidores permite seleccionar los elementos como multiplicador, divisor, pipeline, etc. durante el diseño, por lo que el usuario es capaz de decidir si quiere un mayor rendimiento, con una mayor área de la FPGA ocupada o un rendimiento más reducido incluyendo multitud de opciones como la FPU y las unidades de cálculo, además, junto al NIOS II es el único que permite incluir la unidad de gestión de memoria. Por tanto, es uno de los más recomendables a la hora de realizar el diseño.

1.1.5 ¿Qué softcores soportan Linux?

Anteriormente se han analizado diferentes soluciones disponibles, pero no se ha entrado en profundidad sobre si son o no capaces de ejecutar un sistema operativo, como si de PC se tratasen.

De acuerdo con la documentación de Linux, cualquier kernel cuya versión sea superior a la 2.6 requiere obligatoriamente del uso de la unidad de gestión de memoria (MMU), caché de instrucciones, barrel shifter y al menos dos timers.

Por tanto, los únicos softcores compatibles son en este caso, NIOSII y Microblaze.

1.2 SISTEMA PROPUESTO

Se propone un sistema basado en una FPGA de la familia Virtex-5, sobre este chip se implementará un diseño hardware cuyo SoC estará formado por Microblaze. En Microblaze se integrarán los periféricos tales como GPIO, ethernet, controlador de memoria RAM y dos UART. Dicho microprocesador softcore se encargará de ejecutar un kernel Linux personalizado que a su vez ejecutará una aplicación.

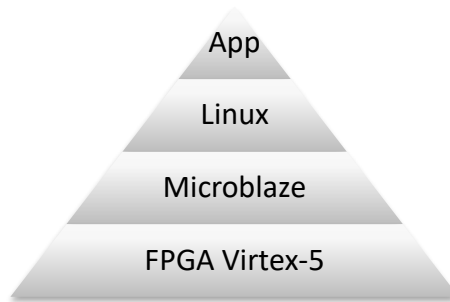


Figura 6: Capas del sistema

Para el diseño del sistema existen básicamente dos líneas de desarrollo, la primera el diseño del hardware, es decir del propio sistema microprocesador, en este caso Microblaze.

La segunda el diseño software, que se centra en personalizar el kernel Linux para que pueda ser ejecutado usando Microblaze en la FPGA. Dentro del diseño software se incluye el diseño de la aplicación que se ejecuta sobre el sistema operativo Linux, haciendo uso del procesador Microblaze y los periféricos asociados. El sistema completo se implementará sobre la placa de desarrollo *Avnet LX-50 Evaluation Board*.

1.2.1 Diseño Hardware

Tal y como se ha comentado con anterioridad, el sistema propuesto cuenta con un microprocesador soft de Xilinx, en concreto Microblaze. El softcore contará con unidad de gestión de memoria, barrel shifter, FPU, divisor, multiplicador, dos timers, módulo de debug y memoria caché tanto para instrucciones como para datos.

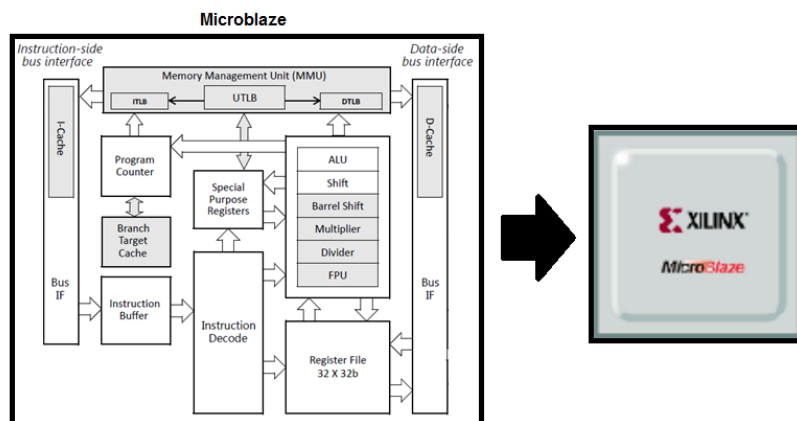


Figura 7: Softcore Microblaze

Dicho procesador incluirá dos controladores para comunicación serie, múltiples GPIO, controlador de memoria RAM y controlador para comunicación vía ethernet.

El esquema simplificado del sistema se muestra en la siguiente figura, donde se pueden apreciar los diferentes periféricos a incluir.

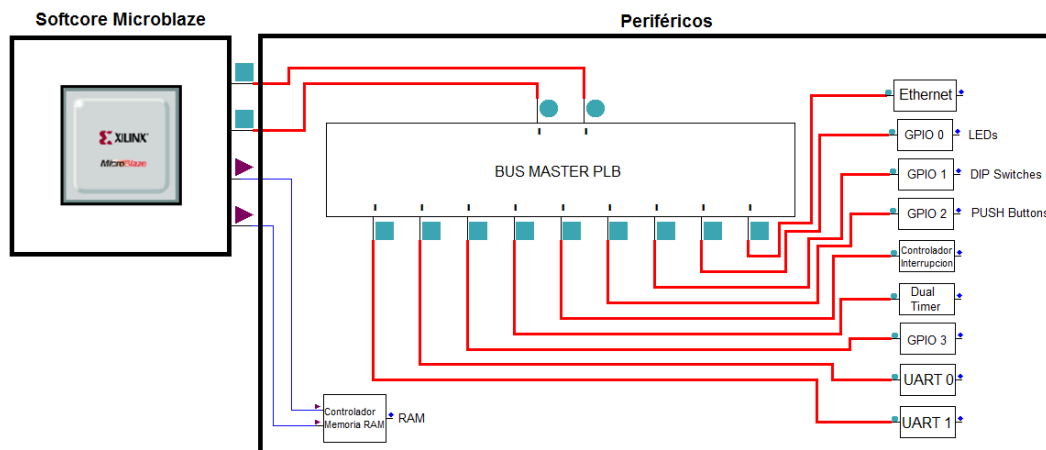


Figura 8: Diagrama hardware del sistema propuesto

1.2.2 Diseño Software

1.2.2.1 Personalización del kernel Linux

Tras obtener el diseño del hardware, se debe configurar el kernel Linux deseado, en este caso la versión 3.8-r1, para que éste sea capaz de ejecutarse haciendo uso del microprocesador Microblaze. Por tanto, es necesario obtener una descripción del hardware para poder ajustar la arquitectura sobre la que se va a ejecutar el sistema operativo, drivers de los periféricos asociados, parámetros de arranque, etc.

1.2.2.2 Diseño de la aplicación

Una vez comprobado el correcto funcionamiento del sistema operativo sobre Microblaze, se comprobarán los periféricos, una vez todos funcionen correctamente se diseñará una aplicación Linux, será un script de Linux, capaz de utilizar los periféricos que se han asociado a Microblaze.

1.3 OBJETIVOS

Una vez descrito el sistema propuesto, se plantean una serie de objetivos que se pretenden alcanzar durante el desarrollo del proyecto. A continuación, se enumeran en función las diferentes capas anteriormente mencionadas.

1.- Diseño del softcore

Mediante el uso de las herramientas proporcionadas por Xilinx se diseñará un softcore cuyo núcleo será Microblaze. Dicho softcore deberá contar con los siguientes elementos:

- Unidad de coma flotante (FPU)
- Unidad de gestión de memoria (MMU)
- Unidad de multiplicación y División
- Memoria Caché de datos e instrucciones

En cuanto a los periféricos, se enumeran a continuación:

- Controlador de memoria RAM asociado a la memoria de 64MB de la placa
- Controlador Ethernet
- Cuatro módulos de control para los diferentes GPIOs
- Dos Timers
- Dos UART
- Módulo de Debug JTAG

2.- Compilación del kernel Linux

Tras obtener el diseño del sistema microprocesador Microblaze, se obtendrá la descripción hardware asociada al mismo, lo que permite diseñar un kernel Linux personalizado.

El kernel Linux deberá cumplir las siguientes premisas:

- Deberá ocupar menos de 64MB, memoria disponible en la placa de desarrollo
- Permitir interacción con el usuario haciendo uso de la UART
- Emplear drivers compatibles con los periféricos asociados a Microblaze
- Transferir información haciendo uso del puerto Ethernet
- Controlar los diferentes periféricos correctamente

3.- Diseño de la aplicación para Linux

Una vez obtenidas las partes de Hardware y Software críticas para ejecutar el sistema operativo en la placa de desarrollo, quedará diseñar una aplicación que demuestre el correcto funcionamiento de la placa de desarrollo, así como las diferentes capas del diseño, esto es, Microblaze, sus periféricos y Linux.

Para ello se diseñará una aplicación cuyos objetivos principales son los siguientes:

- Hacer uso de los periféricos
- Mostrar información obtenida desde la UART y/o GPIOs
- Crear un entorno web con información obtenida desde los periféricos
- Permitir la interacción desde la web con los propios periféricos

En cuanto a los objetivos principales del proyecto, son los siguientes:

- Diseñar e implementar sobre una FPGA Virtex-5 el sistema microprocesador Microblaze, incluyendo los elementos anteriormente mencionados.
- Compilar e implementar sobre la memoria RAM de la placa de desarrollo un kernel Linux personalizado para trabajar con los elementos que forman parte de Microblaze, así como los periféricos añadidos.
- Diseñar y ejecutar sobre Linux una aplicación a modo de servidor web desde la FPGA, con la finalidad de interactuar con los sensores y actuadores asociados a los periféricos de Microblaze.

CAPÍTULO 2

DISEÑO DEL SISTEMA MICROPROCESADOR (SOFTCORE MICROBLAZE)

En este capítulo se describen los pasos del diseño del sistema procesador, para ello hace una breve descripción de la herramienta XPS proporcionada por Xilinx. Una vez descrita la herramienta de desarrollo del sistema microprocesador, se pasa a la descripción del proceso de diseño de Microblaze, dividiéndolo en tres pasos, configuración del proyecto, configuración de Microblaze y compilación del diseño final.

2.1 Software empleado para el desarrollo del hardware

Para el desarrollo del sistema microprocesador la herramienta XPS de Xilinx, ésta y otras herramientas necesarias para el desarrollo del proyecto se encuentran dentro de la ISE Design Suite 14.7 de Xilinx.

XPS son las siglas de *Xilinx Platform Studio*, ésta herramienta permitirá crear y configurar el sistema microprocesador softcore, así como añadir los periféricos deseados, de forma que el diseño sea completamente compatible con la FPGA empleada, en este caso la Virtex-5.

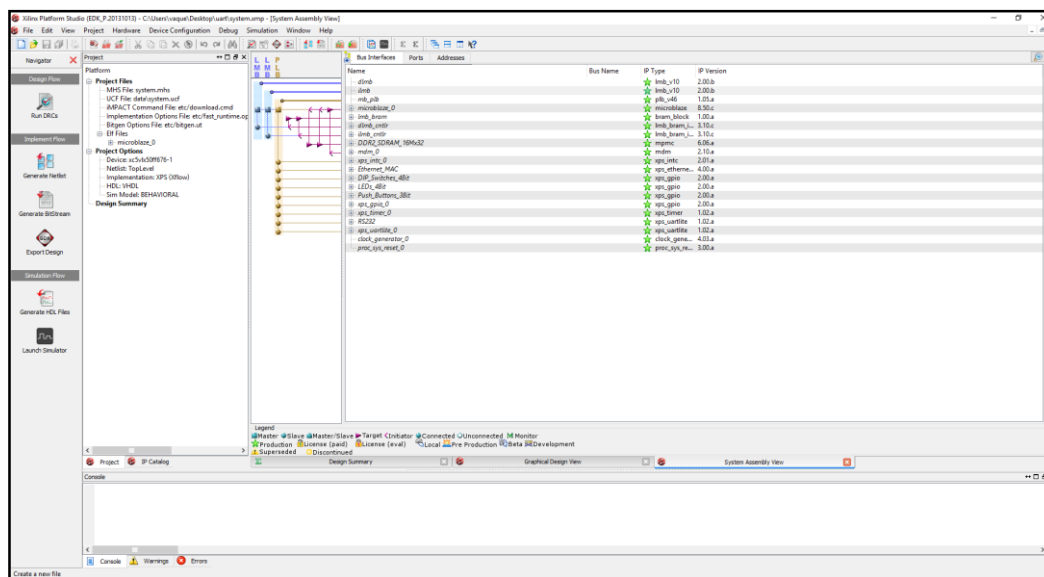


Figura 9: Herramienta Xilinx XPS

2.2 Diseño de la parte hardware

El diseño del softcore consta básicamente de tres pasos, el primero, la creación del proyecto en base a la FPGA a emplear, el segundo, configurar el softcore Microblaze para que cumpla con las especificaciones deseadas y el tercero, exportar el diseño de forma que se obtenga el fichero a cargar sobre la FPGA.

2.2.1 Creación del proyecto para la placa de desarrollo empleada

El primer paso necesario para poder diseñar el sistema basado en Microblaze es configurar el proyecto en base a la placa de desarrollo a emplear, en este caso la Avnet Virtex-5 LX-50 Evaluation Board. El proceso se muestra a continuación.

Para poder implementar el softcore, es necesario configurar el tipo de bus deseado, en este caso, la placa de desarrollo empleada, que cuenta con una FPGA de la familia Virtex-5 solamente es compatible con PLB (Processor Local Bus), existe otra variante de bus más moderna, el AXI (Advanced eXtensible Interface), sin embargo, la FPGA empleada solamente es compatible con PLB, por lo que se selecciona esa opción.

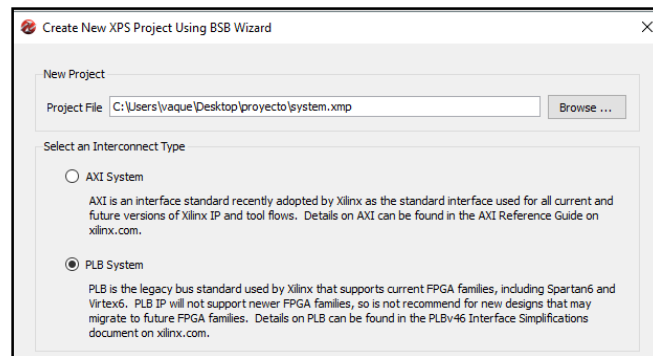


Figura 10: XPS - Selección del tipo de BUS del sistema

Tras configurar el tipo de bus, es necesario escoger entre las diferentes placas de desarrollo compatibles con la versión ISE 14.7. Por tanto se debe escoger la Avnet LX-50, de forma que el entorno seleccione qué chip de las diferentes variantes de Virtex-5 es el que está integrado, tal y como muestra la Figura 11.

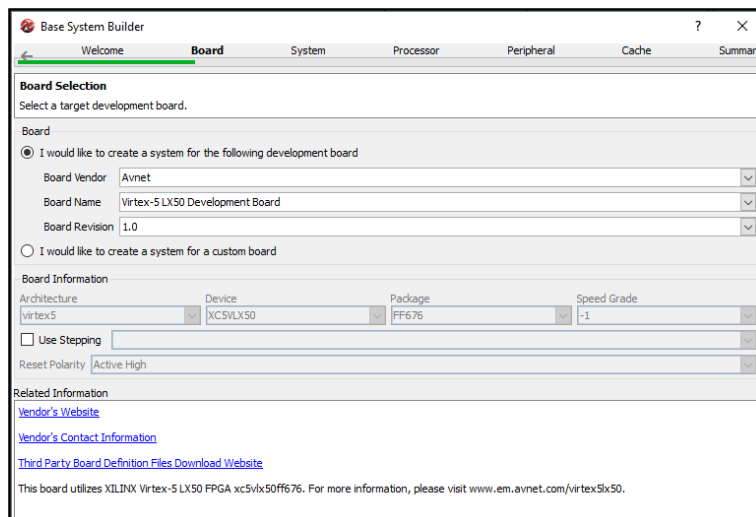


Figura 11: XPS - Selección de la placa de desarrollo

Tras configurar la placa de desarrollo, XPS permite escoger entre la creación de un sistema mono núcleo o multi núcleo, por lo que en este paso se escoge sistema mono núcleo y en el posterior paso se escoge el reloj de referencia de Microblaze a la velocidad más elevada que permita la placa, en este caso 100MHz.

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

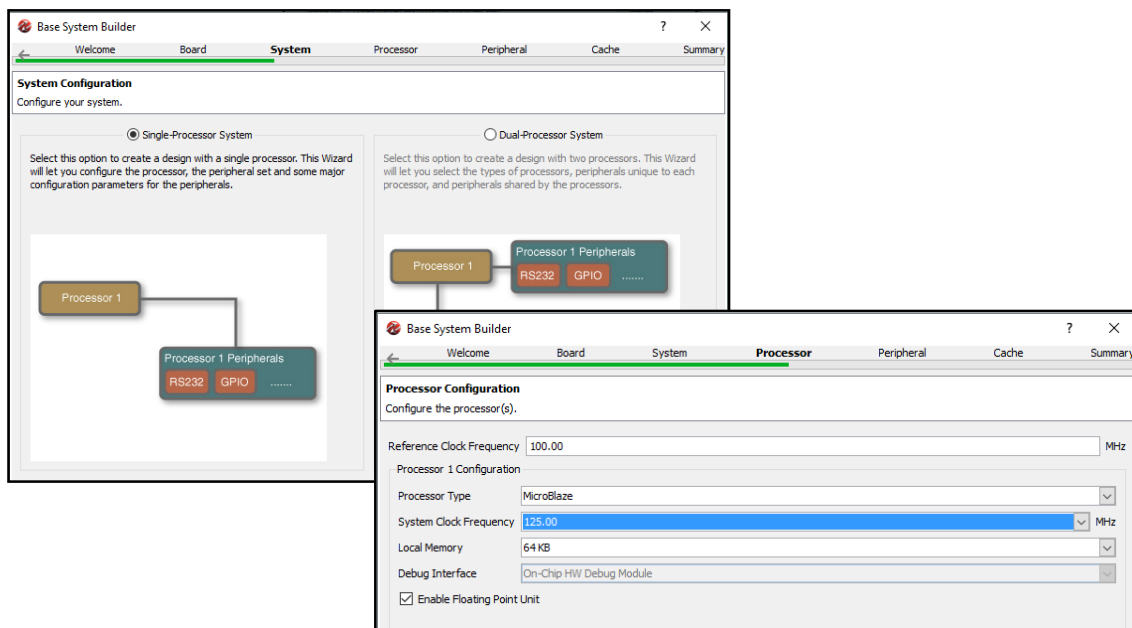


Figura 12: XPS - Configuración inicial del sistema procesador

Una vez escogido el procesador y la velocidad de procesado, el entorno permite seleccionar algunos de los periféricos compatibles con la placa de desarrollo, en este paso se añaden algunos como la UART para comunicación, el controlador Ethernet, los diferentes controladores GPIO para los elementos soldados en la placa, el timer doble y el controlador de memoria RAM. Por lo que, a posteriori será necesario añadir algunos periféricos extra.

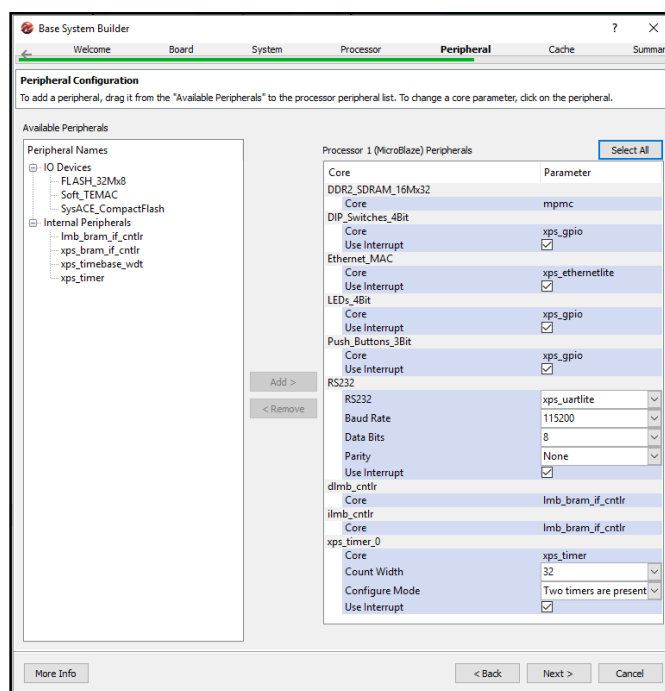


Figura 13: XPS - Selección inicial de periféricos

En este paso, es importante destacar que es necesario activar las interrupciones para cada uno de los periféricos, de otra forma no funcionarán correctamente bajo Linux.

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA
Daniel Vaquerizo Cid

Tras configurar los periféricos iniciales, solamente queda configurar la memoria caché, que posteriormente puede modificarse dentro de la configuración de Microblaze.

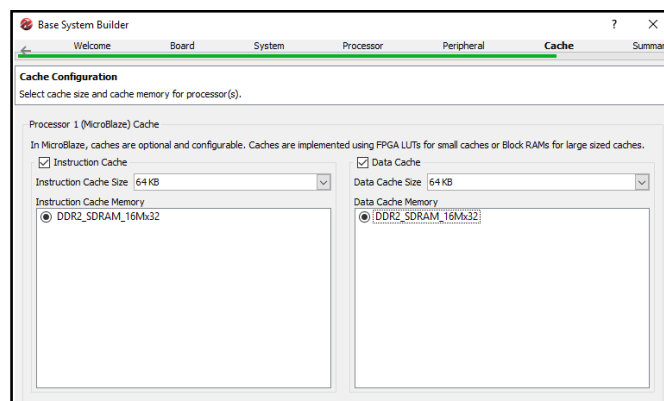


Figura 14: XPS - Configuración memoria Caché

Una vez configurada la memoria cache se muestra un pequeño resumen del sistema creado y se puede pasar a la configuración del softcore.

2.2.2 Configuración de Microblaze y los periféricos

2.2.2.1 Configuración de Microblaze

Tras crear el proyecto se debe configurar correctamente Microblaze para que sea capaz de ejecutar Linux a posteriori. Para ello, el primer paso es seleccionar la instancia "microblaze_0" para poder acceder a su configuración.

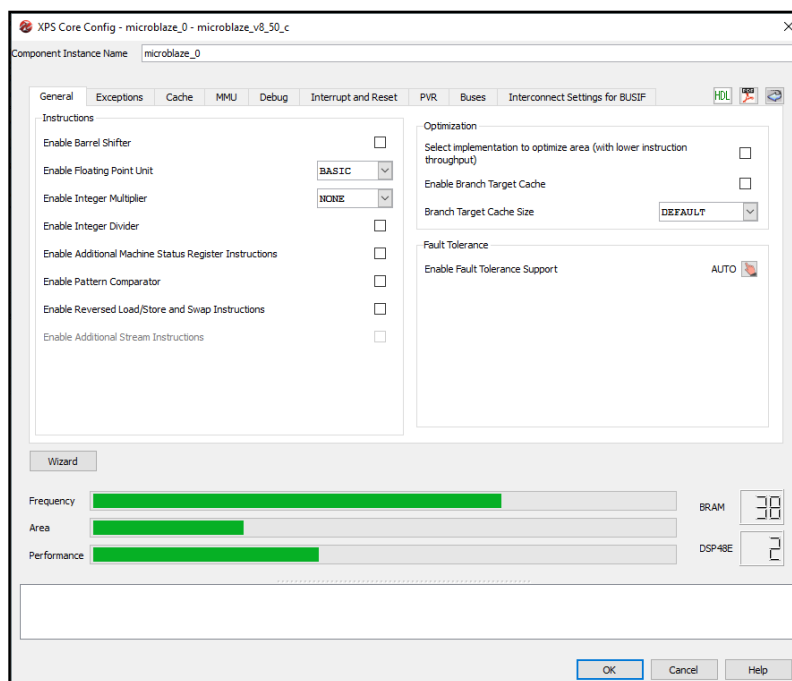


Figura 15: XPS - Panel de configuración de Microblaze

Dentro del panel de configuración de Microblaze se pueden configurar los elementos

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

necesarios como la unidad de gestión de memoria, tamaño de la memoria caché, divisor, multiplicador, etc.

En la configuración general, se activan los diferentes parámetros para incluir en el diseño el hardware de multiplicación, división y barrel shifter, el resto de parámetros son opcionales.

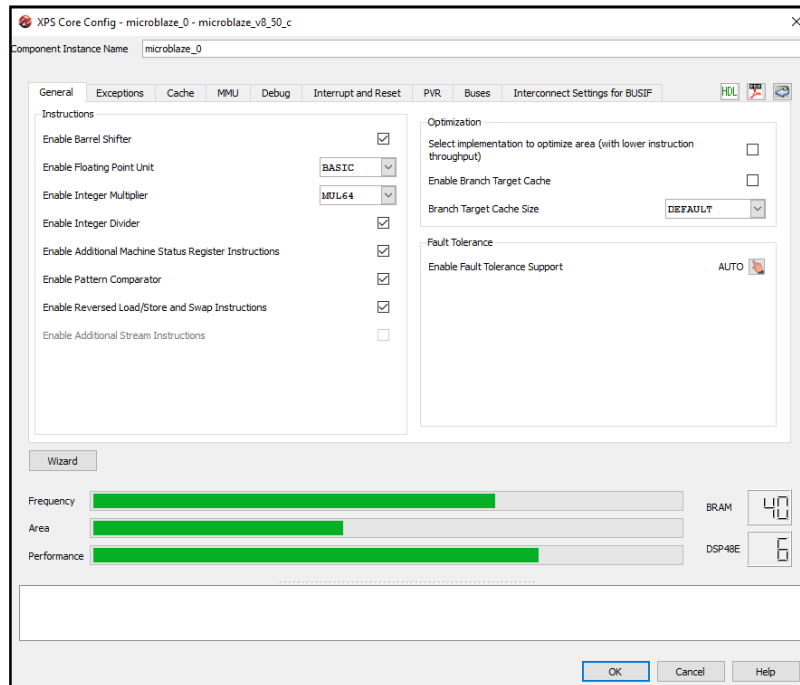


Figura 16: XPS - Microblaze configuración general

A continuación, en la pestaña de Excepciones, se deshabilitan para ahorrar algo de área de la FPGA.

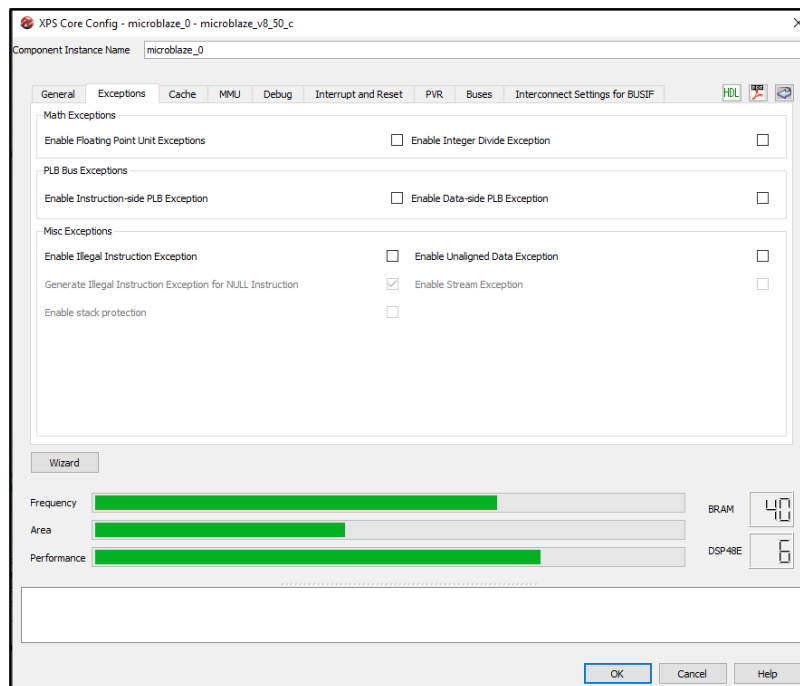


Figura 17: XPS - Microblaze configuración de excepciones

Tras desactivar las excepciones se debe configurar la MMU dado que es un elemento

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

indispensable para poder ejecutar Linux con Microblaze.

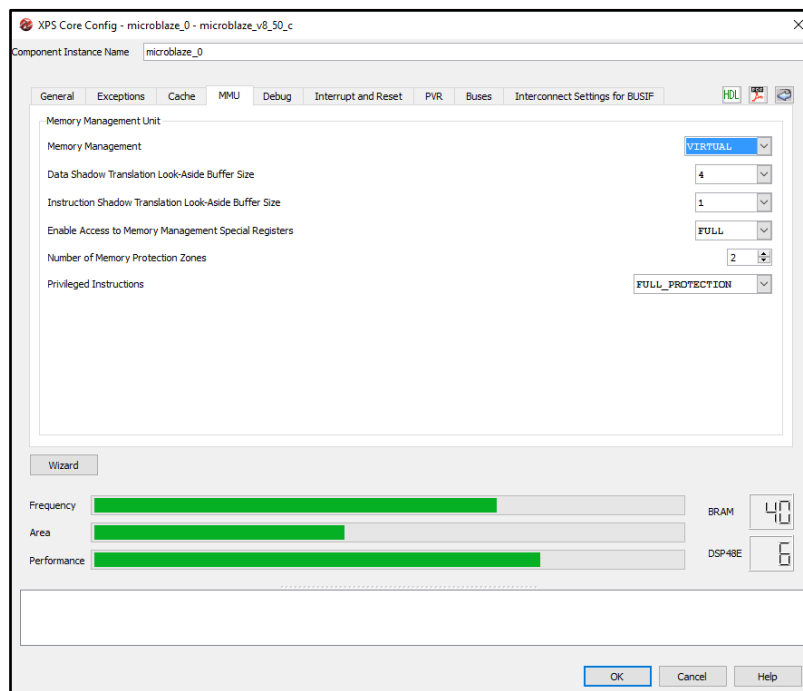


Figura 18: XPS - Microblaze configuración de la MMU

Con el resto de elementos configurados, se desactiva el PVR para ahorrar área en el diseño.

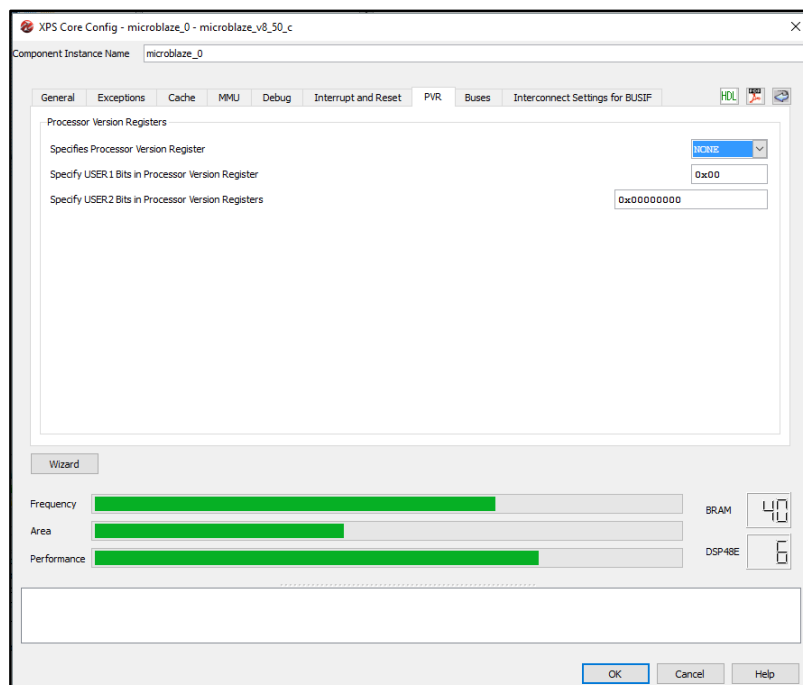
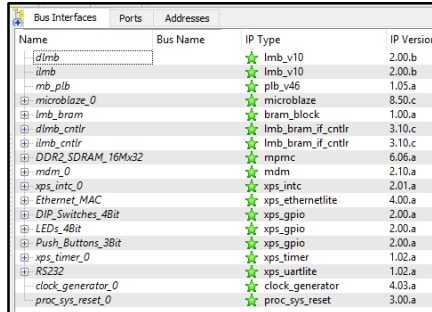


Figura 19: XPS - Microblaze configuración del PVR

2.2.2.2 Configuración de los periféricos

Tras configurar los elementos “internos” de Microblaze es necesario añadir algunos periféricos para ajustar el diseño al sistema propuesto anteriormente.

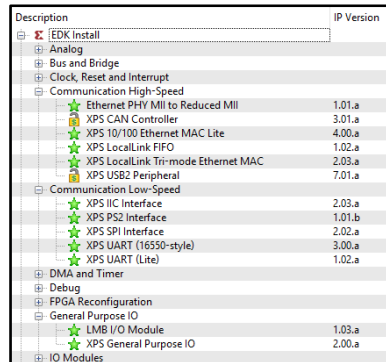


Name	Bus Name	IP Type	IP Version
dlmb		lmb_v10	2.00.b
ilmb		lmb_v10	2.00.b
mb_plb		plb_v46	1.05.a
microblaze_0		microblaze	8.50.c
lmb_bram		bram_block	1.00.a
dlmb_cntlr		lmb_bram_if_cntlr	3.10.c
ilmb_cntlr		lmb_bram_if_cntlr	3.10.c
DDR2_SDRAM_16Mx32		mpmc	6.06.a
mdm_0		mdm	2.10.a
xps_intc_0		xps_intc	2.01.a
Ethernet_MAC		xps_ethernetlite	4.00.a
DIP_Switches_48bit		xps_gpio	2.00.a
LEDs_48bit		xps_gpio	2.00.a
Push_Buttons_38bit		xps_gpio	2.00.a
xps_timer_0		xps_timer	1.02.a
RS232		xps_uartlite	1.02.a
clock_generator_0		clock_generator	4.03.a
proc_sys_reset_0		proc_sys_reset	3.00.a

Figura 20: XPS - Periféricos iniciales Microblaze

Para cumplir con las especificaciones anteriormente indicadas, es necesario añadir una segunda unidad UART y un controlador GPIO extra.

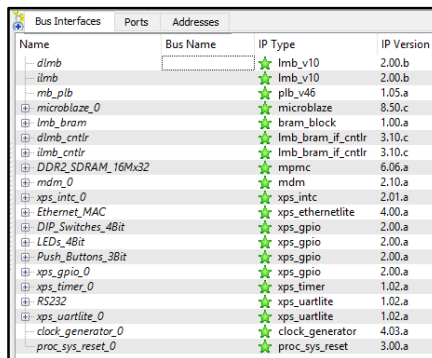
Para añadir periféricos se hace uso de IPs (Intellectual Properties) de Xilinx, que son instancias extra, compatibles con Microblaze, con multitud de funcionalidades, como lo son la UART o el controlador de los GPIO, por ejemplo. En la siguiente figura se muestran algunas de las IPs disponibles.



Description	IP Version
EDK Install	
Analog	
Bus and Bridge	
Clock, Reset and Interrupt	
Communication High-Speed	
Ethernet PHY MII to Reduced MII	1.01.a
XPS CAN Controller	3.01.a
XPS 10/100 Ethernet MAC Lite	4.00.a
XPS LocalLink FIFO	1.02.a
XPS LocalLink Tri-mode Ethernet MAC	2.03.a
XPS USB2 Peripheral	7.01.a
Communication Low-Speed	
XPS IC Interface	2.03.a
XPS PS2 Interface	1.01.b
XPS SPI Interface	2.02.a
XPS UART (16550-style)	3.00.a
XPS UART (Lite)	1.02.a
DMA and Timer	
Debug	
FPGA Reconfiguration	
General Purpose IO	
LMB I/O Module	1.03.a
XPS General Purpose IO	2.00.a
IO Modules	

Figura 21: XPS - IPs disponibles para Microblaze

Tras añadir los periféricos deseados, la lista de elementos asociados a Microblaze queda de la siguiente forma.



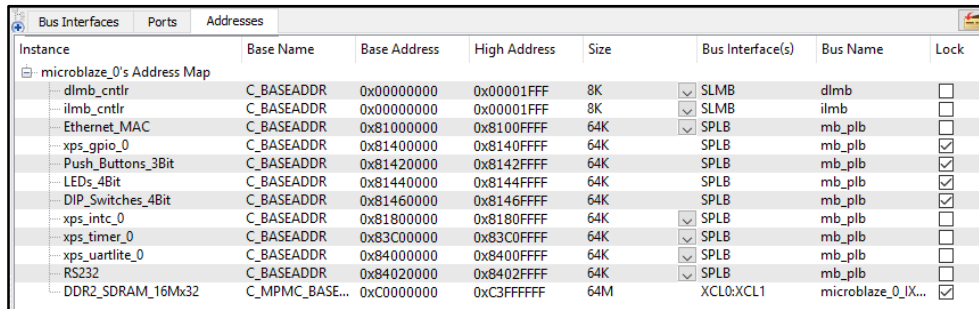
Name	Bus Name	IP Type	IP Version
dlmb		lmb_v10	2.00.b
ilmb		lmb_v10	2.00.b
mb_plb		plb_v46	1.05.a
microblaze_0		microblaze	8.50.c
lmb_bram		bram_block	1.00.a
dlmb_cntlr		lmb_bram_if_cntlr	3.10.c
ilmb_cntlr		lmb_bram_if_cntlr	3.10.c
DDR2_SDRAM_16Mx32		mpmc	6.06.a
mdm_0		mdm	2.10.a
xps_intc_0		xps_intc	2.01.a
Ethernet_MAC		xps_ethernetlite	4.00.a
DIP_Switches_48bit		xps_gpio	2.00.a
LEDs_48bit		xps_gpio	2.00.a
Push_Buttons_38bit		xps_gpio	2.00.a
xps_gpio_0		xps_gpio	2.00.a
xps_timer_0		xps_timer	1.02.a
RS232		xps_uartlite	1.02.a
xps_uartlite_0		xps_uartlite	1.02.a
clock_generator_0		clock_generator	4.03.a
proc_sys_reset_0		proc_sys_reset	3.00.a

Figura 22: XPS - Periféricos finales Microblaze

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

Una vez el diseño cumple con lo establecido en el sistema propuesto y los objetivos, se debe ajustar la dirección de los diferentes periféricos, ya que es necesario fijar la dirección inicial de la memoria RAM en 0xC0000000 para poder ejecutar Linux posteriormente.

Para ello, se modifica manualmente la dirección de la memoria RAM y se recalculan el resto de direcciones.



Instance	Base Name	Base Address	High Address	Size	Bus Interface(s)	Bus Name	Lock
microblaze_0's Address Map							
dlimb_cntlr	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB	dlimb	<input type="checkbox"/>
ilmb_cntlr	C_BASEADDR	0x00000000	0x00001FFF	8K	SLMB	ilmb	<input type="checkbox"/>
Ethernet_MAC	C_BASEADDR	0x81000000	0x8100FFFF	64K	SPLB	mb_plb	<input type="checkbox"/>
xps_gpio_0	C_BASEADDR	0x81400000	0x8140FFFF	64K	SPLB	mb_plb	<input checked="" type="checkbox"/>
Push_Buttons_3Bit	C_BASEADDR	0x81420000	0x8142FFFF	64K	SPLB	mb_plb	<input checked="" type="checkbox"/>
LEDs_4Bit	C_BASEADDR	0x81440000	0x8144FFFF	64K	SPLB	mb_plb	<input checked="" type="checkbox"/>
DIP_Switches_4Bit	C_BASEADDR	0x81460000	0x8146FFFF	64K	SPLB	mb_plb	<input checked="" type="checkbox"/>
xps_intc_0	C_BASEADDR	0x81800000	0x8180FFFF	64K	SPLB	mb_plb	<input type="checkbox"/>
xps_timer_0	C_BASEADDR	0x83C00000	0x83C0FFFF	64K	SPLB	mb_plb	<input type="checkbox"/>
xps_uartlite_0	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB	mb_plb	<input type="checkbox"/>
RS232	C_BASEADDR	0x84020000	0x8402FFFF	64K	SPLB	mb_plb	<input type="checkbox"/>
DDR2_SDRAM_16Mx32	C_MPMC_BASE...	0xC0000000	0xC3FFFFFF	64M	XCL0:XCL1	microblaze_0_JX...	<input checked="" type="checkbox"/>

Figura 23: XPS - Dirección de memoria de los elementos del softcore

Posteriormente se deben configurar las conexiones de cada una de las instancias para decidir cuáles van a ir hacia el exterior de la FPGA y si serán entradas, salidas, o ambos. Una vez realizadas todas las modificaciones sobre el hardware, se pueden ver los cambios realizados en el fichero de descripción del sistema (.mhs), cabe destacar que los ajustes realizados de forma gráfica también se pueden realizar directamente sobre el fichero de texto.

A continuación, se muestran los diferentes fragmentos del fichero.

```
PARAMETER VERSION = 2.1.0
PORT fpga_0_LEDs_4Bit_GPIO_IO_O_pin = fpga_0_LEDs_4Bit_GPIO_IO_O_pin, DIR = O, VEC = [0:3]
PORT fpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin = fpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin, DIR = I, VEC = [0:3]
PORT fpga_0_Push_Buttons_3Bit_GPIO_IO_I_pin = fpga_0_Push_Buttons_3Bit_GPIO_IO_I_pin, DIR = I, VEC = [0:2]
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_Clk_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_Clk_pin, DIR = O, VEC = [1:0]
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_Clk_n_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_Clk_n_pin, DIR = O, VEC = [1:0]
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_CE_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_CE_pin, DIR = O
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_CS_n_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_CS_n_pin, DIR = O
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_ODT_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_ODT_pin, DIR = O
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_RAS_n_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_RAS_n_pin, DIR = O
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_CAS_n_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_CAS_n_pin, DIR = O
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_WE_n_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_WE_n_pin, DIR = O
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_BankAddr_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_BankAddr_pin, DIR = O, VEC = [1:0]
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin, DIR = O, VEC = [12:0]
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin, DIR = IO, VEC = [31:0]
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_DM_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_DM_pin, DIR = O, VEC = [3:0]
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_pin, DIR = IO, VEC = [3:0]
PORT fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_n_pin = fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_n_pin, DIR = IO, VEC = [3:0]
PORT fpga_0_Ethernet_MAC_PHY_tx_clk_pin = fpga_0_Ethernet_MAC_PHY_tx_clk_pin, DIR = I
PORT fpga_0_Ethernet_MAC_PHY_rx_clk_pin = fpga_0_Ethernet_MAC_PHY_rx_clk_pin, DIR = I
PORT fpga_0_Ethernet_MAC_PHY_crs_pin = fpga_0_Ethernet_MAC_PHY_crs_pin, DIR = I
PORT fpga_0_Ethernet_MAC_PHY_dv_pin = fpga_0_Ethernet_MAC_PHY_dv_pin, DIR = I
PORT fpga_0_Ethernet_MAC_PHY_rx_data_pin = fpga_0_Ethernet_MAC_PHY_rx_data_pin, DIR = I, VEC = [3:0]
PORT fpga_0_Ethernet_MAC_PHY_col_pin = fpga_0_Ethernet_MAC_PHY_col_pin, DIR = I
PORT fpga_0_Ethernet_MAC_PHY_rx_er_pin = fpga_0_Ethernet_MAC_PHY_rx_er_pin, DIR = I
PORT fpga_0_Ethernet_MAC_PHY_rst_n_pin = fpga_0_Ethernet_MAC_PHY_rst_n_pin, DIR = O
PORT fpga_0_Ethernet_MAC_PHY_tx_en_pin = fpga_0_Ethernet_MAC_PHY_tx_en_pin, DIR = O
PORT fpga_0_Ethernet_MAC_PHY_tx_data_pin = fpga_0_Ethernet_MAC_PHY_tx_data_pin, DIR = O, VEC = [3:0]
PORT fpga_0_Ethernet_MAC_PHY_MDC_pin = fpga_0_Ethernet_MAC_PHY_MDC_pin, DIR = O
PORT fpga_0_Ethernet_MAC_PHY_MDIO_pin = fpga_0_Ethernet_MAC_PHY_MDIO_pin, DIR = IO
PORT fpga_0_Ethernet_MAC_TXER_pin = net_gnd, DIR = O
PORT fpga_0_RS232_RX_pin = fpga_0_RS232_RX_pin, DIR = I
PORT fpga_0_RS232_TX_pin = fpga_0_RS232_TX_pin, DIR = O
PORT fpga_0_clk_1_sys_clk_pin = CLK_S, DIR = I, SIGIS = CLK, CLK_FREQ = 100000000
PORT fpga_0_rst_1_sys_rst_pin = sys_rst_s, DIR = I, SIGIS = RST, RST_POLARITY = 1
PORT xps_gpio_0_GPIO_IO_O_pin = xps_gpio_0_GPIO_IO_O, DIR = O, VEC = [0:3]
PORT xps_uartlite_0_RX_pin = xps_uartlite_0_RX, DIR = I
PORT xps_uartlite_0_TX_pin = xps_uartlite_0_TX, DIR = O
```

Figura 24: Fichero de descripción .MHS - Puertos

En este primer fragmento se muestran las descripciones de los puertos de cada una de las instancias, así como su tipo (entrada / salida), tamaño (1 bit o varios), etc.

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

```
BEGIN microblaze
PARAMETER INSTANCE = microblaze_0
PARAMETER C_USE_BARREL = 1
PARAMETER C_USE_FPU = 1
PARAMETER C_DEBUG_ENABLED = 1
PARAMETER C_ICACHE_BASEADDR = 0xc0000000
PARAMETER C_ICACHE_HIGHADDR = 0xc3ffff
PARAMETER C_CACHE_BYTE_SIZE = 16384
PARAMETER C_ICACHE_ALWAYS_USED = 1
PARAMETER C_DCACHE_BASEADDR = 0xc0000000
PARAMETER C_DCACHE_HIGHADDR = 0xc3ffff
PARAMETER C_DCACHE_BYTE_SIZE = 16384
PARAMETER C_DCACHE_ALWAYS_USED = 1
PARAMETER HW_VER = 8.50.c
PARAMETER C_USE_ICACHE = 1
PARAMETER C_USE_DCACHE = 1
PARAMETER C_USE_MMU = 3
PARAMETER C_MMU_ITLB_SIZE = 1
PARAMETER C_MMU_ZONES = 2
PARAMETER C_OPCODE_0x0_ILLEGAL = 1
PARAMETER C_USE_HW_MUL = 2
PARAMETER C_USE_DIV = 1
BUS_INTERFACE DPLB = mb_plb
BUS_INTERFACE IPLB = mb_plb
BUS_INTERFACE DXCL = microblaze_0_DXCL
BUS_INTERFACE IXCL = microblaze_0_IXCL
BUS_INTERFACE DEBUG = microblaze_0_mdm_bus
BUS_INTERFACE DLMB = dlmb
BUS_INTERFACE ILMB = ilmb
PORT MB_RESET = mb_reset
PORT INTERRUPT = microblaze_0_Interrupt
END

BEGIN plb_v46
PARAMETER INSTANCE = mb_plb
PARAMETER HW_VER = 1.05.a
PORT PLB_Clk = clk_100_0000MHzPLL0
PORT SYS_Rst = sys_bus_reset
END

BEGIN lmb_v10
PARAMETER INSTANCE = ilmb
PARAMETER HW_VER = 2.00.b
PORT LMB_Clk = clk_100_0000MHzPLL0
PORT SYS_Rst = sys_bus_reset
END

BEGIN lmb_v10
PARAMETER INSTANCE = dlmb
PARAMETER HW_VER = 2.00.b
PORT LMB_Clk = clk_100_0000MHzPLL0
PORT SYS_Rst = sys_bus_reset
END

BEGIN lmb_bram_if_cntlr
PARAMETER INSTANCE = dlmb_cntlr
PARAMETER HW_VER = 3.10.c
PARAMETER C_BASEADDR = 0x00000000
PARAMETER C_HIGHADDR = 0x00001fff
BUS_INTERFACE SLMB = dlmb
BUS_INTERFACE BRAM_PORT = dlmb_port
END

BEGIN lmb_bram_if_cntlr
PARAMETER INSTANCE = ilmb_cntlr
PARAMETER HW_VER = 3.10.c
PARAMETER C_BASEADDR = 0x00000000
PARAMETER C_HIGHADDR = 0x00001fff
BUS_INTERFACE SLMB = ilmb
BUS_INTERFACE BRAM_PORT = ilmb_port
END

BEGIN bram_block
PARAMETER INSTANCE = lmb_bram
PARAMETER HW_VER = 1.00.a
BUS_INTERFACE PORTA = ilmb_port
BUS_INTERFACE PORTB = dlmb_port
END
```

Figura 25: Fichero de descripción .MHS – Microblaze y Buses

En este segundo fragmento se muestran las instancias referentes a Microblaze, se pueden ver las configuraciones del softcore, memoria bram, el bus de memoria local y el bus master PLB. A continuación, se muestra el siguiente fragmento donde se pueden ver los diferentes periféricos que se habían añadido al sistema.

```
BEGIN xps_gpio
PARAMETER INSTANCE = LEDs_4Bit
PARAMETER C_ALL_INPUTS = 0
PARAMETER C_GPIO_WIDTH = 4
PARAMETER C_INTERRUPT_PRESENT = 1
PARAMETER C_IS_DUAL = 0
PARAMETER HW_VER = 2.00.a
PARAMETER C_BASEADDR = 0x81440000
PARAMETER C_HIGHADDR = 0x8144ffff
BUS_INTERFACE SPLB = mb_plb
PORT IP2INTC_Irpt = LEDs_4Bit_IP2INTC_Irpt
PORT GPIO_IO_O = fpga_0_LEDs_4Bit_GPIO_IO_O_pin
END

BEGIN xps_gpio
PARAMETER INSTANCE = DIP_Switches_4Bit
PARAMETER C_ALL_INPUTS = 1
PARAMETER C_GPIO_WIDTH = 4
PARAMETER C_INTERRUPT_PRESENT = 1
PARAMETER C_IS_DUAL = 0
PARAMETER HW_VER = 2.00.a
PARAMETER C_BASEADDR = 0x81460000
PARAMETER C_HIGHADDR = 0x8146ffff
BUS_INTERFACE SPLB = mb_plb
PORT IP2INTC_Irpt = DIP_Switches_4Bit_IP2INTC_Irpt
PORT GPIO_IO_I = fpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin
END

BEGIN xps_gpio
PARAMETER INSTANCE = Push_Butons_3Bit
PARAMETER C_ALL_INPUTS = 1
PARAMETER C_GPIO_WIDTH = 3
PARAMETER C_INTERRUPT_PRESENT = 1
PARAMETER C_IS_DUAL = 0
PARAMETER HW_VER = 2.00.a
PARAMETER C_BASEADDR = 0x81420000
PARAMETER C_HIGHADDR = 0x8142ffff
BUS_INTERFACE SPLB = mb_plb
PORT IP2INTC_Irpt = Push_Butons_3Bit_IP2INTC_Irpt
PORT GPIO_IO_I = fpga_0_Push_Butons_3Bit_GPIO_IO_I_pin
END

BEGIN xps_gpio
PARAMETER INSTANCE = xps_gpio_0
PARAMETER HW_VER = 2.00.a
PARAMETER C_INTERRUPT_PRESENT = 1
PARAMETER C_GPIO_WIDTH = 4
PARAMETER C_BASEADDR = 0x81400000
PARAMETER C_HIGHADDR = 0x8140ffff
BUS_INTERFACE SPLB = mb_plb
PORT GPIO_IO_O = xps_gpio_0_GPIO_IO_O
PORT IP2INTC_Irpt = xps_gpio_0_IP2INTC_Irpt
END

BEGIN xps_uartlite
PARAMETER INSTANCE = xps_uartlite_0
PARAMETER C_BAUDRATE = 9600
PARAMETER C_DATA_BITS = 8
PARAMETER C_USE_PARITY = 0
PARAMETER C_ODD_PARITY = 0
PARAMETER HW_VER = 1.02.a
PARAMETER C_BASEADDR = 0x84000000
PARAMETER C_HIGHADDR = 0x8400ffff
BUS_INTERFACE SPLB = mb_plb
PORT Interrupt = xps_uartlite_0_Interrupt
PORT RX = xps_uartlite_0_RX
PORT TX = xps_uartlite_0_TX
END

BEGIN xps_uartlite
PARAMETER INSTANCE = RS232
PARAMETER C_BAUDRATE = 115200
PARAMETER C_DATA_BITS = 8
PARAMETER C_USE_PARITY = 0
PARAMETER C_ODD_PARITY = 0
PARAMETER HW_VER = 1.02.a
PARAMETER C_BASEADDR = 0x84020000
PARAMETER C_HIGHADDR = 0x8402ffff
BUS_INTERFACE SPLB = mb_plb
PORT RX = fpga_0_RS232_RX_pin
PORT TX = fpga_0_RS232_TX_pin
PORT Interrupt = RS232_Interrupt
END
```

Figura 26: Fichero de descripción .MHS – Periféricos asociados a Microblaze

En el tercer fragmento se pueden ver algunos de los periféricos asociados a Microblaze, así como las diferentes opciones de configuración asociadas a cada uno de los elementos. Se puede ver, por ejemplo, los diferentes GPIOs, según si son solamente entradas, salidas o ambos, la velocidad de la instancia RS232 que es 115200bps y la de la xps_uart_lite0 es 9600bps, las direcciones de cada periférico, entre otros...

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

```
BEGIN mpmc
PARAMETER INSTANCE = DDR2_SDRAM_16Mx32
PARAMETER C_NUM_PORTS = 2
PARAMETER C_NUM_IDELAYCTRL = 2
PARAMETER C_IDELAYCTRL_LOC = IDELAYCTRL_X0Y4-IDELAYCTRL_X2Y4
PARAMETER C_MEM_PARTNO = mt47h16m16-5e
PARAMETER C_MEM_CLK_WIDTH = 2
PARAMETER C_MEM_ODT_WIDTH = 1
PARAMETER C_MEM_CE_WIDTH = 1
PARAMETER C_MEM_CS_N_WIDTH = 1
PARAMETER C_MEM_DATA_WIDTH = 32
PARAMETER C_DDR2_DQSN_ENABLE = 1
PARAMETER C_PIM0_BASETYPE = 1
PARAMETER C_PIM1_BASETYPE = 1
PARAMETER HW_VER = 6.06.a
PARAMETER C_MPMC_BASEADDR = 0xc0000000
PARAMETER C_MPMC_HIGHADDR = 0xc3ffff
BUS_INTERFACE XCL0 = microblaze_0_IJCL
BUS_INTERFACE XCL1 = microblaze_0_DXCL
PORT MPMC_Clk0 = clk_200_0000MHzPLL0
PORT MPMC_Clk0_DIV2 = clk_100_0000MHzPLL0
PORT MPMC_Clk90 = clk_200_0000MHz90PLL0
PORT MPMC_Clk_200MHz = clk_200_0000MHzPLL0
PORT MPMC_Rst = sys_periph_reset
PORT DDR2_Clk = fpga_0_DDR2_SDRAM_16Mx32_DDR2_Clk_pin
PORT DDR2_Clk_n = fpga_0_DDR2_SDRAM_16Mx32_DDR2_Clk_n_pin
PORT DDR2_CE = fpga_0_DDR2_SDRAM_16Mx32_DDR2_CE_pin
PORT DDR2_CS_n = fpga_0_DDR2_SDRAM_16Mx32_DDR2_CS_n_pin
PORT DDR2_ODT = fpga_0_DDR2_SDRAM_16Mx32_DDR2_ODT_pin
PORT DDR2_RAS_n = fpga_0_DDR2_SDRAM_16Mx32_DDR2_RAS_n_pin
PORT DDR2_CAS_n = fpga_0_DDR2_SDRAM_16Mx32_DDR2_CAS_n_pin
PORT DDR2_WE_n = fpga_0_DDR2_SDRAM_16Mx32_DDR2_WE_n_pin
PORT DDR2_BankAddr = fpga_0_DDR2_SDRAM_16Mx32_DDR2_BankAddr_pin
PORT DDR2_Addr = fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin
PORT DDR2_DQ = fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin
PORT DDR2_DM = fpga_0_DDR2_SDRAM_16Mx32_DDR2_DM_pin
PORT DDR2_DQS = fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_pin
PORT DDR2_DQS_n = fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_n_pin
END

BEGIN xps_ethernetlite
PARAMETER INSTANCE = Ethernet_MAC
PARAMETER HW_VER = 4.00.a
PARAMETER C_BASEADDR = 0x81000000
PARAMETER C_HIGHADDR = 0x8100ffff
BUS_INTERFACE SPLB = mb_plb
PORT PHY_tx_clk = fpga_0_Ethernet_MAC_PHY_tx_clk_pin
PORT PHY_rx_clk = fpga_0_Ethernet_MAC_PHY_rx_clk_pin
PORT PHY_crs = fpga_0_Ethernet_MAC_PHY_crs_pin
PORT PHY_dv = fpga_0_Ethernet_MAC_PHY_dv_pin
PORT PHY_rx_data = fpga_0_Ethernet_MAC_PHY_rx_data_pin
PORT PHY_col = fpga_0_Ethernet_MAC_PHY_col_pin
PORT PHY_rx_er = fpga_0_Ethernet_MAC_PHY_rx_er_pin
PORT PHY_rst_n = fpga_0_Ethernet_MAC_PHY_rst_n_pin
PORT PHY_tx_en = fpga_0_Ethernet_MAC_PHY_tx_en_pin
PORT PHY_tx_data = fpga_0_Ethernet_MAC_PHY_tx_data_pin
PORT PHY_MDC = fpga_0_Ethernet_MAC_PHY_MDC_pin
PORT IP2INTC_lrp = Ethernet_MAC_IP2INTC_lrp
PORT PHY_MDIO = fpga_0_Ethernet_MAC_PHY_MDIO_pin
END
```

Figura 27: Fichero de descripción .MHS – Periféricos asociados a Microblaze II

En este cuarto fragmento de descripción, se pueden ver los periféricos más voluminosos en cuanto a área de FPGA, son el controlador de memoria RAM y Ethernet.

En sus instancias se pueden ver las direcciones de memoria que tienen configuradas (que en caso de la RAM es la anteriormente indicada 0xC0000000), puertos asociados a ellas, etc.


```
BEGIN clock_generator
PARAMETER INSTANCE = clock_generator_0
PARAMETER C_CLKIN_FREQ = 100000000
PARAMETER C_CLKOUT0_FREQ = 100000000
PARAMETER C_CLKOUT0_PHASE = 0
PARAMETER C_CLKOUT0_GROUP = PLL0
PARAMETER C_CLKOUT0_BUF = TRUE
PARAMETER C_CLKOUT1_FREQ = 200000000
PARAMETER C_CLKOUT1_PHASE = 90
PARAMETER C_CLKOUT1_GROUP = PLL0
PARAMETER C_CLKOUT1_BUF = TRUE
PARAMETER C_CLKOUT2_FREQ = 200000000
PARAMETER C_CLKOUT2_PHASE = 0
PARAMETER C_CLKOUT2_GROUP = PLL0
PARAMETER C_CLKOUT2_BUF = TRUE
PARAMETER C_EXT_RESET_HIGH = 1
PARAMETER HW_VER = 4.03.a
PORT CLKIN = CLK_S
PORT CLKOUT0 = clk_100_0000MHzPLL0
PORT CLKOUT1 = clk_200_0000MHz90PLL0
PORT CLKOUT2 = clk_200_0000MHzPLL0
PORT RST = sys_rst_s
PORT LOCKED = Dcm_all_locked
END

BEGIN mdm
PARAMETER INSTANCE = mdm_0
PARAMETER C_MB_DBG_PORTS = 1
PARAMETER C_USE_UART = 0
PARAMETER HW_VER = 2.10.a
BUS_INTERFACE MBDEBUG_0 = microblaze_0_mdm_bus
PORT Debug_SYS_Rst = Debug_SYS_Rst
END

BEGIN proc_sys_reset
PARAMETER INSTANCE = proc_sys_reset_0
PARAMETER C_EXT_RESET_HIGH = 1
PARAMETER HW_VER = 3.00.a
PORT Slowest_sync_clk = clk_100_0000MHzPLL0
PORT Ext_Reset_In = sys_rst_s
PORT MB_Debug_Sys_Rst = Debug_SYS_Rst
PORT Dcm_locked = Dcm_all_locked
PORT MB_Reset = mb_reset
PORT Bus_Struct_Reset = sys_bus_reset
PORT Peripheral_Reset = sys_periph_reset
END

BEGIN xps_intc
PARAMETER INSTANCE = xps_intc_0
PARAMETER HW_VER = 2.01.a
PARAMETER C_BASEADDR = 0x81800000
PARAMETER C_HIGHADDR = 0x8180ffff
BUS_INTERFACE SPLB = mb_plb
PORT Intr = LEDs_4Bit_IP2INTC_Irpt & DIP_Switches_4Bit_IP2INTC_Irpt & Push_Buttons_3Bit_IP2INTC_Irpt & Ethernet_MAC_IP2INTC_Irpt & RS232_Interrupt & xps_timer_0_Interrupt & xps_gpio_0_IP2INTC_Irpt & xps_uartlite_0_Interrupt
PORT Irq = microblaze_0_Interrupt
END
```

Figura 28: Fichero de descripción .MHS – Elementos de Microblaze

En el fragmento final del fichero se pueden encontrar las instancias correspondientes al generador de la señal de reloj, la gestión del reset, módulo de debug y el controlador de interrupciones, donde se pueden ver las interrupciones de cada uno de los periféricos anteriormente comentados.

2.2.2.3 Configuración de las restricciones del diseño

Tras analizar el fichero de descripción del hardware, se pasa a analizar el fichero de restricciones de diseño, que permiten escoger en que pines “físicos” se conectan las diferentes instancias del diseño de Microblaze. Para indicar a XPS éstas restricciones se debe modificar el fichero .ucf, que contiene las restricciones del diseño que se va a implementar en la FPGA Virtex-5.

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA
Daniel Vaquerizo Cid

Para poder modificar el fichero de restricciones es necesario consultar la documentación de la placa [1], donde se puede ver a que corresponde cada pin, y por tanto configurar correctamente las restricciones.

Los GPIOs para los Leds, Switchs y Botones, son elementos soldados en la placa, tal y como muestra la Figura 29, por lo que su localización y conexionado viene fijada por el fabricante.

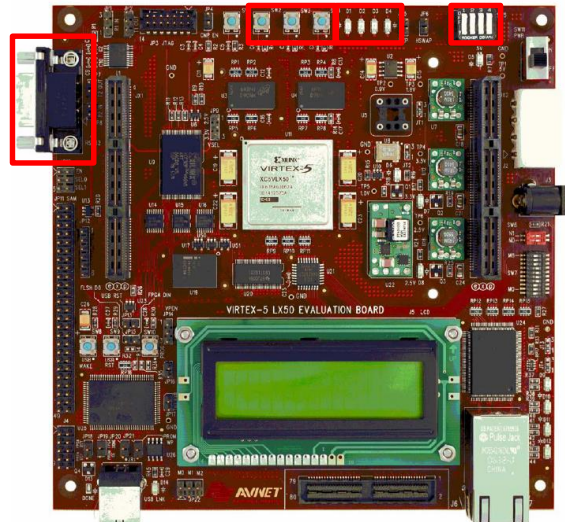


Figura 29: Placa de desarrollo Avnet Virtex-5 LX50

La Tabla 6 muestra la correspondencia entre los elementos mencionados y los pines físicos de la FPGA.

Nodo	Virtex-5 Pin
LED0	E11
LED1	E17
LED2	F10
LED3	F19
SWITCH0	B26
SWITCH1	C26
SWITCH2	D26
SWITCH3	D25
SWITCH_PB2	B2
SWITCH_PB3	E8
SWITCH_PB4	F17

Tabla 6: Localización de los elementos

Una vez conocida la localización de los elementos físicos de la placa, se puede añadir la restricción en el fichero .ucf, teniendo en cuenta que deben corresponder con los puertos de salida de cada una de las instancias GPIO que se habían añadido anteriormente durante el diseño de Microblaze. La primera parte del fichero queda de la siguiente forma:

```
Net fpga_0_LEDs_4Bit_GPIO_IO_O_pin<0> LOC=E11 | IOSTANDARD = LVCMOS25;
Net fpga_0_LEDs_4Bit_GPIO_IO_O_pin<1> LOC=E17 | IOSTANDARD = LVCMOS25;
Net fpga_0_LEDs_4Bit_GPIO_IO_O_pin<2> LOC=F10 | IOSTANDARD = LVCMOS25;
Net fpga_0_LEDs_4Bit_GPIO_IO_O_pin<3> LOC=F19 | IOSTANDARD = LVCMOS25;
Net fpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin<0> LOC=B26 | IOSTANDARD = LVCMOS18;
Net fpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin<1> LOC=C26 | IOSTANDARD = LVCMOS18;
Net fpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin<2> LOC=D26 | IOSTANDARD = LVCMOS18;
Net fpga_0_DIP_Switches_4Bit_GPIO_IO_I_pin<3> LOC=D25 | IOSTANDARD = LVCMOS18;
Net fpga_0_Push_Buttons_3Bit_GPIO_IO_I_pin<0> LOC=B2 | PULLUP | IOSTANDARD = LVCMOS18;
Net fpga_0_Push_Buttons_3Bit_GPIO_IO_I_pin<1> LOC=E8 | PULLUP | IOSTANDARD = LVCMOS25;
Net fpga_0_Push_Buttons_3Bit_GPIO_IO_I_pin<2> LOC=F17 | PULLUP | IOSTANDARD = LVCMOS25;
```

Figura 30: Fichero de restricciones .ucf – GPIOs de elementos soldados en la placa

Posteriormente, es necesario añadir las restricciones referentes a las UART y el último controlador para GPIOs.

Para la primera UART, que irá conectada al puerto RS232 de la placa de desarrollo, es necesario consultar de nuevo la documentación de la placa [1] para encontrar a qué pines corresponde.

Nodo	Virtex-5 Pin
RS232_RXD	AB7
RS232_TXD	AC6

Tabla 7: Localización del puerto RS232

A continuación, es necesario decidir a qué pines van a ir conectadas las últimas instancias de GPIO y UART. La conexión de estos elementos es de libre elección, siempre y cuando se encuentren dentro del conector JP11 de la placa de desarrollo.

Las ubicaciones escogidas son las siguientes:

Nodo	Virtex-5 Pin
Uart_lite_0_TX	Y3
Uart_lite_0_RX	U6
GPIO_0	V7
GPIO_1	W6
GPIO_2	T5
GPIO_3	T7

Tabla 8: Localización de elementos del usuario

Tras escoger la ubicación, el siguiente fragmento tiene el siguiente aspecto. Tal y como se puede observar, los pines escogidos van asociados a cada uno de los puertos de las instancias de gpio_0 y uartlite_0.

```
Net fpga_0_RS232_RX_pin LOC=AB7 | IOSTANDARD = LVCMOS33;
Net fpga_0_RS232_TX_pin LOC=AC6 | IOSTANDARD = LVCMOS33;
Net xps_uartlite_0_RX_pin LOC = U6 | IOSTANDARD = LVCMOS33;
Net xps_uartlite_0_TX_pin LOC = Y3 | IOSTANDARD = LVCMOS33;
Net xps_gpio_0_GPIO_IO_O_pin<0> LOC=V7 | IOSTANDARD = LVCMOS33;
Net xps_gpio_0_GPIO_IO_O_pin<1> LOC=W6 | IOSTANDARD = LVCMOS33;
Net xps_gpio_0_GPIO_IO_O_pin<2> LOC=T5 | IOSTANDARD = LVCMOS33;
Net xps_gpio_0_GPIO_IO_O_pin<3> LOC=T7 | IOSTANDARD = LVCMOS33;
```

Figura 31: Fichero de restricciones .ucf – GPIO y UART del usuario

Por último y para finalizar con la configuración de las restricciones, se muestra la

configuración correspondiente al controlador de memoria RAM y la comunicación vía Ethernet. Cabe destacar que las asignaciones de pines correspondientes se realizan automáticamente al añadir sus IPs, así como las del clock y reset.

```
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Clk_pin<0> LOC=A18 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Clk_pin<1> LOC=B9 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Clk_n_pin<0> LOC=A19 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Clk_n_pin<1> LOC=B10 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_CE_pin LOC=A17 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_CS_n_pin LOC=D10 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_ODT_pin LOC=C9 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_RAS_n_pin LOC=A10 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_CAS_n_pin LOC=D11 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_WE_n_pin LOC=B17 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_BankAddr_pin<0> LOC=C13 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_BankAddr_pin<1> LOC=D16 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<0> LOC=B11 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<1> LOC=A14 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<2> LOC=C11 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<3> LOC=A12 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<4> LOC=C12 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<5> LOC=B12 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<6> LOC=B16 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<7> LOC=C14 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<8> LOC=B15 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<9> LOC=B14 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<10> LOC=A13 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<11> LOC=A15 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_Addr_pin<12> LOC=C16 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<0> LOC=B24 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<1> LOC=D24 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<2> LOC=B25 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<3> LOC=C24 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<4> LOC=C23 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<5> LOC=A25 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<6> LOC=D23 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<7> LOC=A23 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<8> LOC=C21 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<9> LOC=B19 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<10> LOC=D21 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<11> LOC=C18 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<12> LOC=D18 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<13> LOC=C22 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<14> LOC=D20 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<15> LOC=B21 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<16> LOC=B5 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<17> LOC=D5 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<18> LOC=A5 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<19> LOC=C6 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<20> LOC=C7 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<21> LOC=B6 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<22> LOC=D6 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<23> LOC=A4 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<24> LOC=A3 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<25> LOC=C2 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<26> LOC=B4 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<27> LOC=D1 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<28> LOC=C1 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQ_pin<29> LOC=C4 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DM_pin<0> LOC=B22 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DM_pin<1> LOC=A22 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DM_pin<2> LOC=A8 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DM_pin<3> LOC=A9 | IOSTANDARD = SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_pin<0> LOC=A20 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_pin<1> LOC=C19 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_pin<2> LOC=D8 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_pin<3> LOC=B7 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_n_pin<0> LOC=B20 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_n_pin<1> LOC=D19 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_n_pin<2> LOC=C8 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_DDR2_SDRAM_16Mx32_DDR2_DQS_n_pin<3> LOC=A7 | IOSTANDARD = DIFF_SSTL18_II;
Net fpga_0_Ethernet_MAC_PHY_tx_clk_pin LOC=AC17 | IOSTANDARD = LVCMOS25 | PERIOD=40000 ps;
Net fpga_0_Ethernet_MAC_PHY_rx_clk_pin LOC=AC8 | IOSTANDARD = LVCMOS25 | PERIOD=40000 ps;
Net fpga_0_Ethernet_MAC_PHY_crs_pin LOC=AE13 | IOSTANDARD = LVCMOS25 | IOBDELAY=NONE;
Net fpga_0_Ethernet_MAC_PHY_dv_pin LOC=AE12 | IOSTANDARD = LVCMOS25 | IOBDELAY=NONE;
Net fpga_0_Ethernet_MAC_PHY_rx_data_pin<0> LOC=AE8 | IOSTANDARD = LVCMOS25 | IOBDELAY=NONE;
Net fpga_0_Ethernet_MAC_PHY_rx_data_pin<1> LOC=AF9 | IOSTANDARD = LVCMOS25 | IOBDELAY=NONE;
Net fpga_0_Ethernet_MAC_PHY_rx_data_pin<2> LOC=AD9 | IOSTANDARD = LVCMOS25 | IOBDELAY=NONE;
Net fpga_0_Ethernet_MAC_PHY_rx_data_pin<3> LOC=AF10 | IOSTANDARD = LVCMOS25 | IOBDELAY=NONE;
Net fpga_0_Ethernet_MAC_PHY_col_pin LOC=AC13 | IOSTANDARD = LVCMOS25 | IOBDELAY=NONE;
Net fpga_0_Ethernet_MAC_PHY_rx_er_pin LOC=AB12 | IOSTANDARD = LVCMOS25 | IOBDELAY=NONE;
Net fpga_0_Ethernet_MAC_PHY_rst_n_pin LOC=AC14 | IOSTANDARD = LVCMOS25;
Net fpga_0_Ethernet_MAC_PHY_tx_en_pin LOC=AF17 | IOSTANDARD = LVCMOS25;
Net fpga_0_Ethernet_MAC_PHY_tx_data_pin<0> LOC=AD16 | IOSTANDARD = LVCMOS25;
Net fpga_0_Ethernet_MAC_PHY_tx_data_pin<1> LOC=AE16 | IOSTANDARD = LVCMOS25;
Net fpga_0_Ethernet_MAC_PHY_tx_data_pin<2> LOC=AE15 | IOSTANDARD = LVCMOS25;
Net fpga_0_Ethernet_MAC_PHY_tx_data_pin<3> LOC=AF15 | IOSTANDARD = LVCMOS25;
Net fpga_0_Ethernet_MAC_PHY_MDC_pin LOC=AE7 | IOSTANDARD = LVCMOS25;
Net fpga_0_Ethernet_MAC_PHY_MDIO_pin LOC=AF7 | IOSTANDARD = LVCMOS25;
Net fpga_0_Ethernet_MAC_TXER_pin LOC=AE17 | IOSTANDARD = LVCMOS25;
Net fpga_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin;
TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100000 kHz;
Net fpga_0_clk_1_sys_clk_pin TNM_NET = sys_clk_pin | LOC=E18 | IOSTANDARD = LVCMOS25;
Net fpga_0_rst_1_sys_rst_pin TIG;
Net fpga_0_rst_1_sys_rst_pin LOC=B1 | IOSTANDARD = LVCMOS18;
```

Figura 32: Fichero de restricciones .ucf – Ethernet, RAM, reset y clock

2.2.3 Compilación del diseño hardware

Una vez completada la configuración de Microblaze, sus periféricos y las restricciones del diseño, se puede proceder a compilar el diseño, con la finalidad de obtener el diseño implementable sobre la FPGA.

Para ello es necesario emplear la opción “Export Design to SDK”, dentro del software XPS, dicha función se encargará de generar los diferentes ficheros necesarios para implementar el diseño sobre la FPGA y exportar el diseño al software SDK (Software Development Kit) de Xilinx, que se empleará posteriormente.

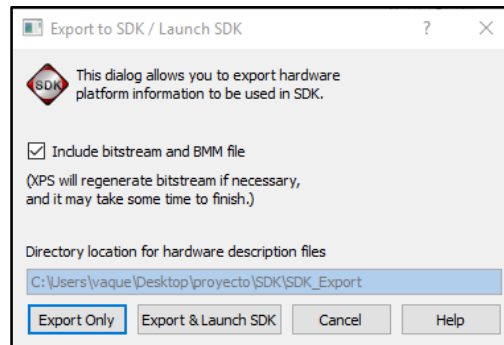


Figura 33: XPS - Export del diseño a SDK

Tras finalizar la compilación del diseño, no deberían existir errores y por tanto se debería abrir el entorno de desarrollo SDK, sin embargo, en este caso, tras la configuración de los diferentes elementos, el diseño no es implementable sobre la FPGA escogida, debido a un uso excesivo de BRAMs.

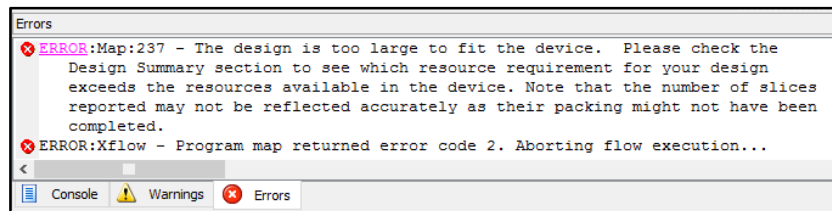


Figura 34: XPS - Errores en el Export del diseño a SDK

Tras analizar el problema, se llega a la conclusión de que el elemento que más área está ocupando es la memoria caché escogida, por lo que reduciendo su tamaño de 64kB a 32kB o un valor inferior, el diseño sí es implementable y se puede exportar sin problemas a SDK.

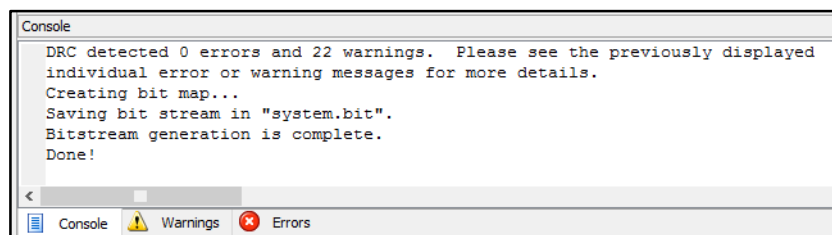


Figura 35: XPS - Final de compilación del diseño

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

Tras completar la compilación, se muestra el resumen de recursos de la FPGA empleados para la implementación de Microblaze y todos sus periféricos.

Device Utilization Summary:		
Number of BSCANs	1 out of 4	25%
Number of BUFPGs	4 out of 32	12%
Number of BUFIOs	4 out of 56	7%
Number of DSP48Es	6 out of 48	12%
Number of IDELAYCTRLs	2 out of 16	12%
Number of LOCed IDELAYCTRLs	2 out of 2	100%
Number of ILOGICs	47 out of 560	8%
Number of LOCed ILOGICs	4 out of 47	8%
Number of External IOBs	108 out of 440	24%
Number of LOCed IOBs	108 out of 108	100%
Number of IODELAYS	40 out of 560	7%
Number of LOCed IODELAYS	4 out of 40	10%
Number of OLOGICs	86 out of 560	15%
Number of PLL_ADVs	1 out of 6	16%
Number of RAMB36_EXPs	31 out of 48	64%
Number of Slices	3865 out of 7200	53%
Number of Slice Registers	7233 out of 28800	25%
Number used as Flip Flops	7216	
Number used as Latches	0	
Number used as LatchThrus	17	
Number of Slice LUTs	7214 out of 28800	25%
Number of Slice LUT-Flip Flop pairs	10278 out of 28800	35%

Figura 36: XPS - Resumen de recursos empleados de la FPGA

Finalmente, una vez compilado y tras exportar el diseño a SDK, se muestra un resumen del hardware a implementar sobre la FPGA, en él se pueden ver los valores de las direcciones de cada uno de los periféricos, así como las numerosas IPs empleadas para crear el sistema.

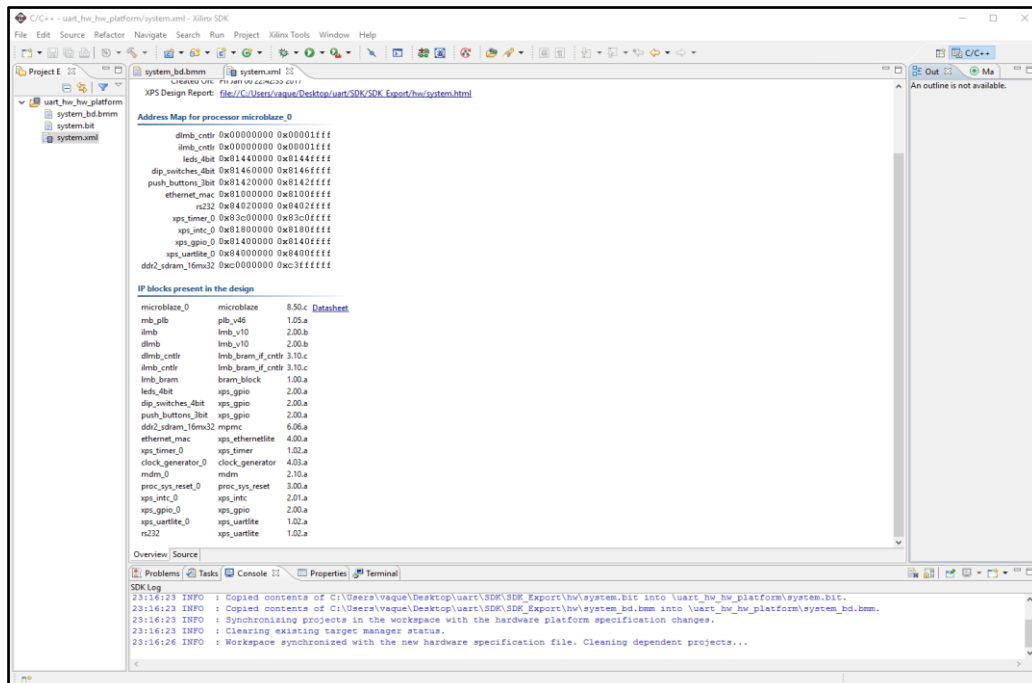


Figura 37: SDK - Resumen del hardware a implementar

Una vez finalizado el diseño del hardware y obtenidos los ficheros de implementación del sistema software sobre la FPGA, se pasa a la parte de diseño software.

CAPÍTULO 3

DISEÑO DE LA PARTE SOFTWARE

Este tercer capítulo se centra en el desarrollo de la parte software, esto es, el diseño y compilación del sistema operativo Linux capaz de ser ejecutado empleando Microblaze anteriormente desarrollado.

Durante el capítulo se hace una breve descripción de la herramienta SDK, proporcionada por Xilinx, así como la herramienta de compilación del kernel Linux 3.8-r1 necesario para obtener el sistema listo para ejecutar con el softcore.

Posteriormente se divide el proceso de diseño en cuatro pasos, que son, la obtención de los elementos necesarios para poder compilar el sistema operativo Linux, la obtención del sistema de ficheros, la configuración del kernel y la compilación del mismo.

3.1 Software empleado para el desarrollo del kernel

Para la compilación del kernel son necesarias principalmente dos herramientas software, la primera es el software SDK, incluido en la suite ISE 14.7, que permitirá obtener la descripción del hardware diseñado, en este caso, Microblaze y los periféricos.

La segunda es el kernel base 3.8-r1, que permite compilar el sistema operativo Linux para diferentes plataformas, en este caso Microblaze. Vale la pena destacar que el kernel Linux solamente puede compilarse bajo un entorno Linux, por lo que para tal finalidad se va a emplear Ubuntu.

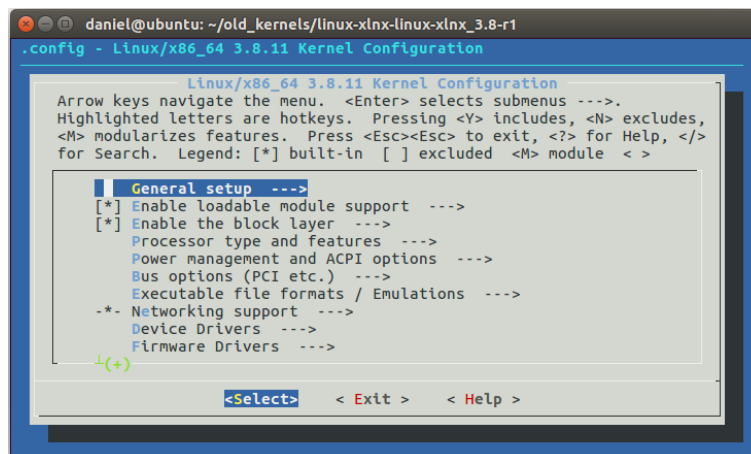


Figura 38: Kernel Linux 3.8-r1 - Menú de configuración

3.2 Creación del kernel Linux para Microblaze

La creación del kernel Linux se divide en 4 pasos principales, el primero de ellos consiste en obtener los ficheros necesarios para poder compilar el kernel, una vez obtenidos se procede a la obtención del sistema de ficheros, imprescindible para poder crear, modificar y guardar información tras el arranque del sistema Linux.

A continuación, con todos los elementos obtenidos se puede proceder a la configuración del kernel, y posteriormente a su compilación, para obtener el ejecutable que se cargará en la memoria RAM de la placa de desarrollo LX-50 de Avnet.

3.2.1 Obtención de los ficheros necesarios

El primer paso del proceso de obtención del kernel es la obtención de los ficheros necesarios para poder compilar el sistema operativo, para ello, el primer fichero imprescindible es la descripción del hardware, el llamado *device-tree*, que permitirá a la herramienta de compilación del sistema operativo Linux conocer las direcciones de los diferentes periféricos, memoria disponible, velocidad el procesador, etc.

Para obtener el fichero de descripción de hardware es necesario emplear la herramienta anteriormente mencionada SDK ya que permitirá crear el fichero *device-tree* con la información sobre Microblaze.

Antes de poder crearlo es necesario añadir al repositorio del proyecto de SDK un nuevo BSP (Board Support Package), dicho BSP se encarga de “traducir” las especificaciones de cada uno de los periféricos y de Microblaze a formato texto, dándoles el formato necesario para que la herramienta de compilación del kernel sea capaz de entenderlo, generando por tanto un fichero .dts.

El BSP generador del *device-tree* se puede obtener desde el repositorio GitHub de Xilinx [2]. Una vez obtenido debe añadirse al repositorio de forma que los BSP disponibles pasen de 2 a 3, tal y como se muestra a continuación.

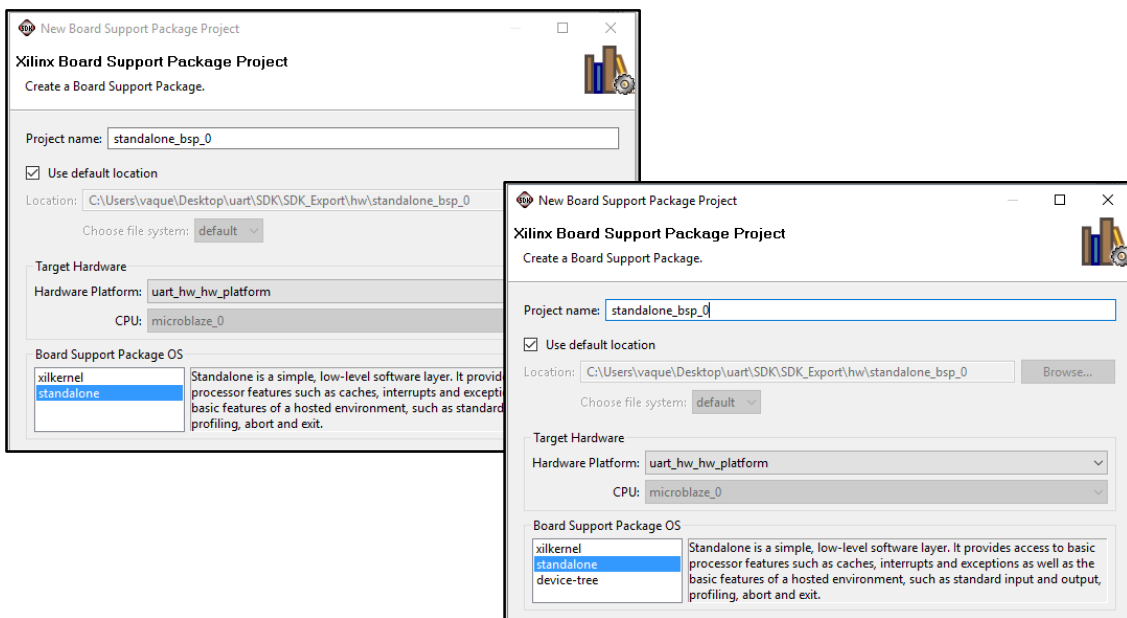


Figura 39: SDK - BSPs Antes/Después añadir el generador de device-tree

Tras seleccionar el BSP para generar el *device-tree*, es necesario configurar algunas opciones como el dispositivo usado para comunicar con el sistema operativo y los argumentos de arranque.

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

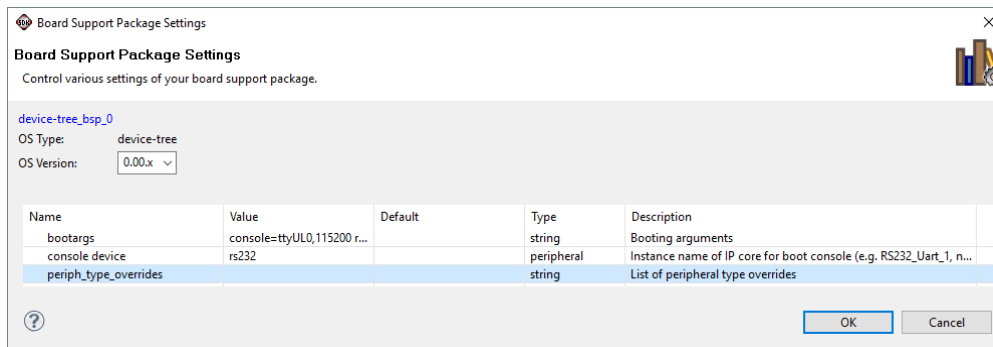


Figura 40: SDK - Configuración del BSP generador de device-tree

En este caso, tal y como muestra la Figura 40, la consola será el RS232 y los argumentos de arranque serán "console=ttyUL0,115200 root=/dev/ram rw ip=on earlyprintk".

Dichos argumentos indican al kernel que la consola es el dispositivo ttyUL0, que funciona a 115200bps, que el sistema de ficheros debe cargarse en memoria RAM permitiendo lectura y escritura, que existe un dispositivo de red y por último que muestre a través de la consola los pasos del arranque.

Tras configurar el BSP correctamente, se comienza la compilación, tras finalizar se muestra el siguiente mensaje y se genera el fichero xilinx.dts, que contiene la descripción del hardware a implementar.

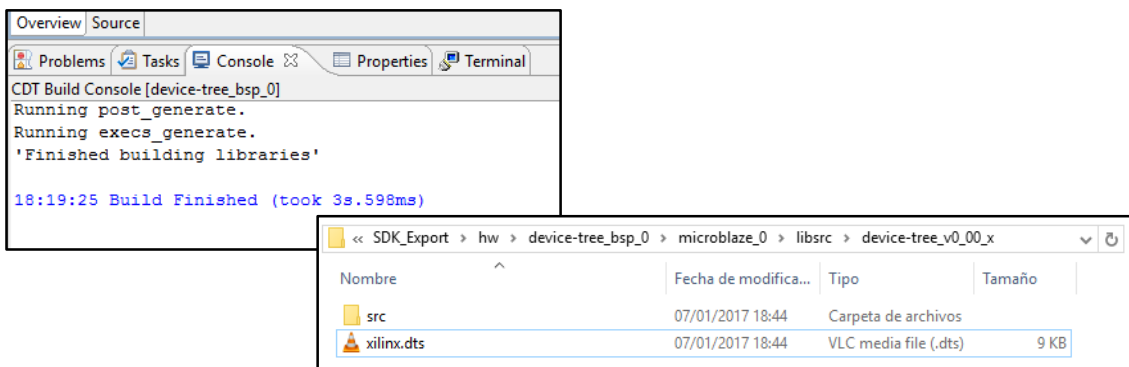


Figura 41: SDK - Obtención del fichero de descripción .dts

A continuación, se muestra el contenido del fichero de descripción del hardware, en él, se pueden encontrar los datos relevantes de configuración de Microblaze y los periféricos.

```
/dts-v1/;
/ {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "xlnx,microblaze";
    model = "Xilinx MicroBlaze";
    aliases {
        ethernet0 = &ethernet_mac;
        serial0 = &rs232;
        serial1 = &xps_uartlite_0;
    };
    chosen {
        bootargs = "console=ttyUL0,115200 root=/dev/ram rw ip=on earlyprintk";
        linux,stdout-path = "/plb@0/serial@84020000";
    };
    cpus {
        #address-cells = <1>;
        #cpus = <0x1>;
        #size-cells = <0>;
        microblaze_0: cpu@0 {
            bus-handle = <&mb_plb>;clock-frequency = <100000000>;
            compatible = "xlnx,microblaze-8.50.c";d-cache-baseaddr = <0xc0000000>;
            d-cache-highaddr = <0xc3ffffff>;d-cache-line-size = <0x10>;
            d-cache-size = <0x8000>;device_type = "cpu";
            i-cache-baseaddr = <0xc0000000>;i-cache-highaddr = <0xc3ffffff>;i-cache-line-size = <0x10>;
            i-cache-size = <0x8000>;interrupt-handle = <&xps_intc_0>;
            model = "microblaze,8.50.c";reg = <0>;
            timebase-frequency = <100000000>;xlnx,addr-tag-bits = <0xb>;
            xlnx,allow-dcache-wr = <0x1>;xlnx,allow-icache-wr = <0x1>;
            xlnx,area-optimized = <0x0>;xlnx,avoid-primitives = <0x0>;
            xlnx,base-vectors = <0x0>;xlnx,branch-target-cache-size = <0x0>;
            xlnx,cache-byte-size = <0x8000>;xlnx,d-axi = <0x0>;
            xlnx,d-lmb = <0x1>;xlnx,d-plb = <0x1>;
            xlnx,data-size = <0x20>;xlnx,dcache-addr-tag = <0xb>;
            xlnx,dcache-always-used = <0x1>;xlnx,dcache-byte-size = <0x8000>;
            xlnx,dcache-data-width = <0x0>;xlnx,dcache-force-tag-lutram = <0x0>;
            xlnx,dcache-interface = <0x0>;xlnx,dcache-line-len = <0x4>;
            xlnx,dcache-use-fsl = <0x1>;xlnx,dcache-use-writeback = <0x0>;
            xlnx,dcache-victims = <0x0>;xlnx,debug-enabled = <0x1>;
            xlnx,div-zero-exception = <0x0>;xlnx,dynamic-bus-sizing = <0x1>;
            xlnx,ecc-use-ce-exception = <0x0>;xlnx,edge-is-positive = <0x1>;
            xlnx,endianness = <0x0>;xlnx,fault-tolerant = <0x0>;
            xlnx,fpu-exception = <0x0>;xlnx,freq = <0x5f5e100>;
            xlnx,fsl-data-size = <0x20>;xlnx,fsl-exception = <0x0>;
            xlnx,fsl-links = <0x0>;xlnx,i-axi = <0x0>;
            xlnx,i-lmb = <0x1>;xlnx,i-plb = <0x1>;xlnx,icache-always-used = <0x1>;
            xlnx,icache-data-width = <0x0>;xlnx,icache-force-tag-lutram = <0x0>;
            xlnx,icache-interface = <0x0>;xlnx,icache-line-len = <0x4>;
            xlnx,icache-streams = <0x0>;xlnx,icache-use-fsl = <0x1>;
            xlnx,icache-victims = <0x0>;xlnx,ill-opcode-exception = <0x0>;
            xlnx,instance = "microblaze_0";xlnx,interconnect = <0x1>;
            xlnx,interrupt-is-edge = <0x0>;xlnx,lockstep-slave = <0x0>;
            xlnx,mmu-dtlb-size = <0x4>;xlnx,mmu-itlb-size = <0x1>;
            xlnx,mmu-privileged-instr = <0x0>;xlnx,mmu-tlb-access = <0x3>;
            xlnx,mmu-zones = <0x2>;xlnx,number-of-pc-brk = <0x1>;
            xlnx,number-of-rd-addr-brk = <0x0>;xlnx,number-of-wr-addr-brk = <0x0>;
            xlnx,opcode-0x0-illegal = <0x1>;xlnx,optimization = <0x0>;
            xlnx,pc-width = <0x20>;xlnx,pvr = <0x0>;
            xlnx,pvr-user1 = <0x0>;xlnx,pvr-user2 = <0x0>;
            xlnx,reset-msr = <0x0>;xlnx,sco = <0x0>;
            xlnx,stream-interconnect = <0x0>;xlnx,unaligned-exceptions = <0x0>;
            xlnx,use-barrel = <0x1>;xlnx,use-branch-target-cache = <0x0>;
            xlnx,use-dcache = <0x1>;xlnx,use-div = <0x1>;
            xlnx,use-ext-brk = <0x1>;xlnx,use-ext-nm-brk = <0x1>;
            xlnx,use-extended-fsl-instr = <0x0>;xlnx,use-fpu = <0x1>;
            xlnx,use-hw-mul = <0x2>;xlnx,use-icache = <0x1>;
            xlnx,use-interrupt = <0x1>;xlnx,use-mmu = <0x3>;
            xlnx,use-msr-instr = <0x1>;xlnx,use-pcmp-instr = <0x1>;
            xlnx,use-reorder-instr = <0x1>;xlnx,use-stack-protection = <0x0>;
        };
    };
};
```

Figura 42: Fichero de descripción HW .dts - Descripción de Microblaze

En este primer fragmento del fichero se puede encontrar la descripción de los argumentos de arranque, la dirección de la consola, en este caso la UART y la configuración de Microblaze.

Realizando un pequeño análisis del fragmento se puede ver como se especifica el uso del bus PLB, que cuenta con multiplicador, divisor, unidad de coma flotante e incluso el barrel shifter. Además, se pueden ver las direcciones de memoria caché, frecuencia del reloj, etc.

Después de la definición de la configuración de Microblaze aparecen los periféricos, que se muestran a continuación.

```
};
ddr2_sdram_16mx32: memory@c0000000 {
    device_type = "memory";
    reg = <0xc0000000 0x4000000>;
};
mb_plb: plb@0 {
    #address-cells = <1>;
    #size-cells = <1>;
    compatible = "xlnx,plb-v46-1.05.a", "xlnx,plb-v46-1.00.a", "simple-bus";
    ranges ;
    dip_switches_4bit: gpio@81460000 {
        #gpio-cells = <2>;
        compatible = "xlnx,xps-gpio-2.00.a", "xlnx,xps-gpio-1.00.a";
        gpio-controller ;
        interrupt-parent = <&xps_intc_0>;
        interrupts = <6 2>;
        reg = <0x81460000 0x10000>;
        xlnx,all-inputs = <0x1>;
        xlnx,all-inputs-2 = <0x0>;
        xlnx,dout-default = <0x0>;
        xlnx,dout-default-2 = <0x0>;
        xlnx,gpio-width = <0x4>;
        xlnx,gpio2-width = <0x20>;
        xlnx,interrupt-present = <0x1>;
        xlnx,is-dual = <0x0>;
        xlnx,tri-default = <0xffffffff>;
        xlnx,tri-default-2 = <0xffffffff>;
    };
    ethernet_mac: ethernet@81000000 {
        compatible = "xlnx,xps-ethernetlite-4.00.a", "xlnx,xps-ethernetlite-1.00.a";
        device_type = "network";
        interrupt-parent = <&xps_intc_0>;
        interrupts = <4 0>;
        local-mac-address = [00 0a 35 00 00 00];
        phy-handle = <&phy0>;
        reg = <0x81000000 0x10000>;
        xlnx,duplex = <0x1>;
        xlnx,include-global-buffers = <0x0>;
        xlnx,include-internal-loopback = <0x0>;
        xlnx,include-mdio = <0x1>;
        xlnx,rx-ping-pong = <0x0>;
        xlnx,tx-ping-pong = <0x0>;
        mdio {
            #address-cells = <1>;
            #size-cells = <0>;
            phy0: phy@7 {
                compatible = "marvell,88e1111";
                device_type = "ethernet-phy";
                reg = <7>;
            };
        };
    };
    leds_4bit: gpio@81440000 {
        #gpio-cells = <2>;
        compatible = "xlnx,xps-gpio-2.00.a", "xlnx,xps-gpio-1.00.a";
        gpio-controller ;
        interrupt-parent = <&xps_intc_0>;
        interrupts = <7 2>;
        reg = <0x81440000 0x10000>;
        xlnx,all-inputs = <0x0>;
        xlnx,all-inputs-2 = <0x0>;
        xlnx,dout-default = <0x0>;
        xlnx,dout-default-2 = <0x0>;
        xlnx,gpio-width = <0x4>;
        xlnx,gpio2-width = <0x20>;
        xlnx,interrupt-present = <0x1>;
        xlnx,is-dual = <0x0>;
        xlnx,tri-default = <0xffffffff>;
        xlnx,tri-default-2 = <0xffffffff>;
    };
};
```

Figura 43: Fichero de descripción HW .dts - Descripción de periféricos I

En esta segunda parte del fichero se pueden encontrar las definiciones de los periféricos tales como la memoria RAM y sus direcciones de memoria inicial y final, los GPIO para los switches y los Leds, con información sobre el número de entradas/salidas, la configuración de éstas por defecto y sus direcciones de memoria, y, por último, la definición del controlador ethernet, donde se puede ver la dirección MAC del dispositivo, direcciones y la configuración del mismo.

```
push_buttons_3bit: gpio@81420000 {
    #gpio-cells = <2>;
    compatible = "xlnx,xps-gpio-2.00.a", "xlnx,xps-gpio-1.00.a";
    gpio-controller ;
    interrupt-parent = <&xps_intc_0>;
    interrupts = <5 2>;
    reg = <0x81420000 0x10000>;
    xlnx,all-inputs = <0x1>;
    xlnx,all-inputs-2 = <0x0>;
    xlnx,dout-default = <0x0>;
    xlnx,dout-default-2 = <0x0>;
    xlnx,gpio-width = <0x3>;
    xlnx,gpio2-width = <0x20>;
    xlnx,interrupt-present = <0x1>;
    xlnx,is-dual = <0x0>;
    xlnx,tri-default = <0xffffffff>;
    xlnx,tri-default-2 = <0xffffffff>;
};

rs232: serial@84020000 {
    clock-frequency = <100000000>;
    compatible = "xlnx,xps-uartlite-1.02.a", "xlnx,xps-uartlite-1.00.a";
    current-speed = <115200>;
    device_type = "serial";
    interrupt-parent = <&xps_intc_0>;
    interrupts = <3 0>;
    port-number = <0>;
    reg = <0x84020000 0x10000>;
    xlnx,baudrate = <0x1c200>;
    xlnx,data-bits = <0x8>;
    xlnx,odd-parity = <0x0>;
    xlnx,use-parity = <0x0>;
};

xps_gpio_0: gpio@81400000 {
    #gpio-cells = <2>;
    compatible = "xlnx,xps-gpio-2.00.a", "xlnx,xps-gpio-1.00.a";
    gpio-controller ;
    interrupt-parent = <&xps_intc_0>;
    interrupts = <1 2>;
    reg = <0x81400000 0x10000>;
    xlnx,all-inputs = <0x0>;
    xlnx,all-inputs-2 = <0x0>;
    xlnx,dout-default = <0x0>;
    xlnx,dout-default-2 = <0x0>;
    xlnx,gpio-width = <0x4>;
    xlnx,gpio2-width = <0x20>;
    xlnx,interrupt-present = <0x1>;
    xlnx,is-dual = <0x0>;
    xlnx,tri-default = <0xffffffff>;
    xlnx,tri-default-2 = <0xffffffff>;
};

xps_intc_0: interrupt-controller@81800000 {
    #interrupt-cells = <0x2>;
    compatible = "xlnx,xps-intc-2.01.a", "xlnx,xps-intc-1.00.a";
    interrupt-controller ;
    reg = <0x81800000 0x10000>;
    xlnx,kind-of-intr = <0x1d>;
    xlnx,num-intr-inputs = <0x8>;
};

xps_timer_0: timer@83c00000 {
    compatible = "xlnx,xps-timer-1.02.a", "xlnx,xps-timer-1.00.a";
    interrupt-parent = <&xps_intc_0>;
    interrupts = <2 0>;
    reg = <0x83c00000 0x10000>;
    xlnx,count-width = <0x20>;
    xlnx,gen0-assert = <0x1>;
    xlnx,gen1-assert = <0x1>;
    xlnx,one-timer-only = <0x0>;
    xlnx,trig0-assert = <0x1>;
    xlnx,trig1-assert = <0x1>;
};

xps_uartlite_0: serial@84000000 {
    clock-frequency = <100000000>;
    compatible = "xlnx,xps-uartlite-1.02.a", "xlnx,xps-uartlite-1.00.a";
    current-speed = <9600>;
    device_type = "serial";
    interrupt-parent = <&xps_intc_0>;
    interrupts = <0 0>;
    port-number = <1>;
    reg = <0x84000000 0x10000>;
    xlnx,baudrate = <0x2580>;
    xlnx,data-bits = <0x8>;
    xlnx,odd-parity = <0x0>;
    xlnx,use-parity = <0x0>;
};
```

Figura 44: Fichero de descripción HW .dts - Descripción de periféricos II

La Figura 44 muestra la parte final de la descripción del hardware presente en el diseño, en el fragmento se pueden encontrar el resto de GPIOs, ambas UART, con sus respectivas velocidades y configuración, el controlador de interrupciones y el timer, en todas ellas se puede ver la dirección a la que van asociadas y su configuración, información imprescindible para poder compilar el kernel.

Por último, es necesario obtener el kernel base 3.8-r1 del repositorio GitHub de Xilinx [2], que permitirá, bajo un equipo Linux, obtener un kernel personalizado totalmente compatible con Microblaze. A continuación, se trata sobre el sistema de ficheros, elemento imprescindible para tener un sistema operativo Linux funcional.

3.2.2 Obtención del sistema de ficheros

Para poder ejecutar el kernel y poder modificar, crear y guardar elementos es necesario disponer de un sistema de ficheros, existen dos posibilidades para obtenerlo.

La primera, descargarlo de la wiki de Xilinx [3], donde existen dos variantes disponibles, la más interesante es la variante completa ya que incluye BusyBox, de forma que empleando este sistema de ficheros se dispone de multitud de herramientas de utilidad. La segunda, compilar el sistema de ficheros desde 0 empleando la herramienta Buildroot. Esta segunda opción permite escoger qué elementos se desean incluir en el sistema de ficheros y ajustar así el tamaño del mismo, sin embargo, es un proceso tedioso y se deben configurar cada una de las opciones de las que dispone la herramienta de compilación para poder obtener un sistema de ficheros funcional.

Dado que no es objetivo de este proyecto la creación del sistema de ficheros y sus aplicaciones, se escoge la primera opción, un sistema de ficheros ya compilado y listo para su uso, que Xilinx pone a disposición de los usuarios que emplean Microblaze como elemento procesador.

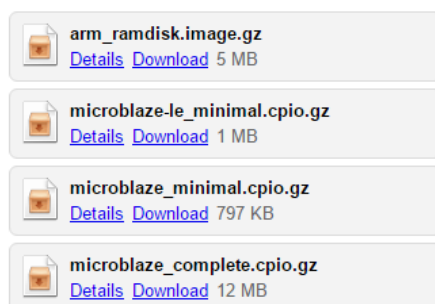


Figura 45: Sistema de ficheros para Linux (Xilinx Wiki)

En la wiki de Xilinx, se encuentra las 4 imágenes comprimidas que contienen el sistema de ficheros completo para poder compilar el kernel Linux, sin embargo, cada una sirve para un sistema concreto.

La primera, solamente es compatible con sistemas cuyo procesador sea de tipo hardware y basado en la arquitectura ARM.

La segunda es específica para Microblaze Little Endian, que son sistemas softcore basados en Microblaze con bus AXI.

Y las dos últimas, son para Microblaze Big Endian, que son los sistemas softcore Microblaze que emplean el bus PLB, como es este caso, y por tanto, cualquiera de las dos es perfectamente compatible para compilar el kernel Linux del sistema propuesto, aunque tal y como se ha comentado, se va a emplear la imagen “microblaze_complete.gpio.gz”.

3.2.3 Configuración del kernel

El primer paso para la configuración del kernel es copiar el fichero de descripción de hardware (xilinx.dts), obtenido anteriormente, a la ruta específica para Microblaze, que es la siguiente:

```
/linux-xlnx-linux-xlnx_3.8-r1/arch/microblaze/boot
```

El segundo paso es copiar el sistema de ficheros a la carpeta principal del kernel.

```
/linux-xlnx-linux-xlnx_3.8-r1/
```

Tras mover ambos ficheros, cabe destacar que el compilador depende de multitud de paquetes de Linux, como, por ejemplo: make, gcc, etc. y teniendo en cuenta que tanto los paquetes de los que depende como la suite ISE 14.7 deben estar instalados, se debe abrir un terminal en la carpeta raíz del kernel.

Tras abrirlo, se puede ejecutar los siguientes comandos.

```
source /opt/Xilinx/14.7/ISE_DS/settings64.sh
export CROSS_COMPILE=microblaze-xilinx-linux-gnu-
make ARCH=microblaze mmu_defconfig
```

El primero indica al terminal donde se encuentran las herramientas de Xilinx, el segundo se encarga de configurar el cross compiler para Microblaze Big Endian, y el tercero, se encarga de crear una configuración por defecto para Microblaze con MMU.

Tras ejecutar los tres comandos se debe crear un fichero de configuración .config, que contiene la configuración de compilación por defecto para Microblaze.

El siguiente paso consiste en abrir el menú de configuración del kernel y ajustar el diseño al hardware diseñado. Para ello, es necesario ejecutar el comando:

```
make ARCH=microblaze menuconfig
```

Lo cual abrirá la siguiente ventana, donde se muestra un entorno gráfico que permite variar la configuración del kernel para ajustarla a el diseño actual.

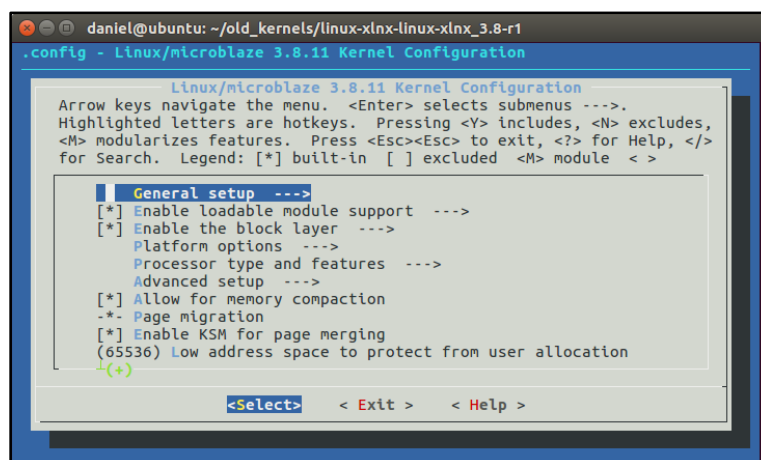


Figura 46: Kernel 3.8-r1 - Menú de configuración

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

Tras abrir el menú de configuración, el primer paso es navegar hasta las opciones generales e indicar al compilador que debe emplear un sistema de ficheros RAM inicial y que éste se llama “microblaze_complete.cpio.gz”, además hay que indicarle que añada soporte para sistemas de ficheros comprimidos usando gzip, dado que el empleado está en formato comprimido .gz, tal y como muestra la siguiente figura.

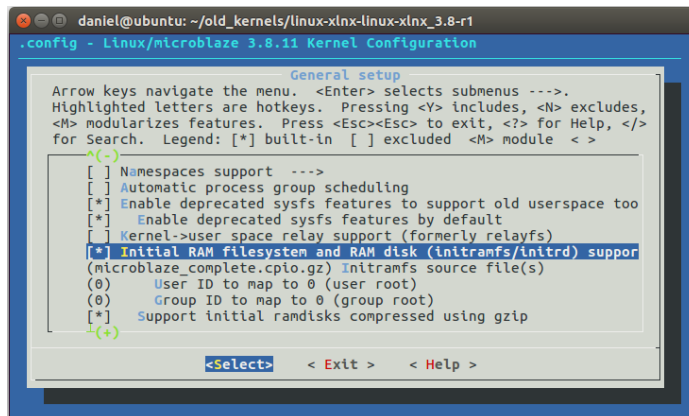


Figura 47: Kernel 3.8-r1 - Ajuste del sistema de ficheros

A continuación, se debe ajustar correctamente la plataforma, para indicar que Microblaze tiene multiplicador, divisor y FPU, además se indica en este paso la dirección de arranque del kernel, que tal y como se ha comentado anteriormente debe ser 0xC0000000, valor que debe coincidir con la dirección de inicio de la memoria RAM.

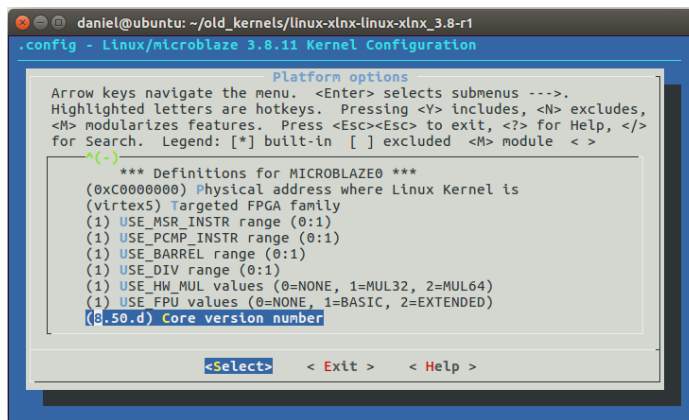


Figura 48: Kernel 3.8-r1 - Ajustes de la plataforma empleada

Tras ajustar las opciones de plataforma, se debe desactivar el BUS PCI, ya que la placa empleada no dispone de él y activar el soporte para red, de otra forma, el driver de ethernet y la funcionalidad ethernet no estarían disponibles al arrancar el sistema operativo.

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

Para configurar correctamente el soporte ethernet, se debe activar la opción "Soporte de Red". Navegando por el submenú se puede escoger si debe tener soporte para IPv4, IPv6, TCP, etc.

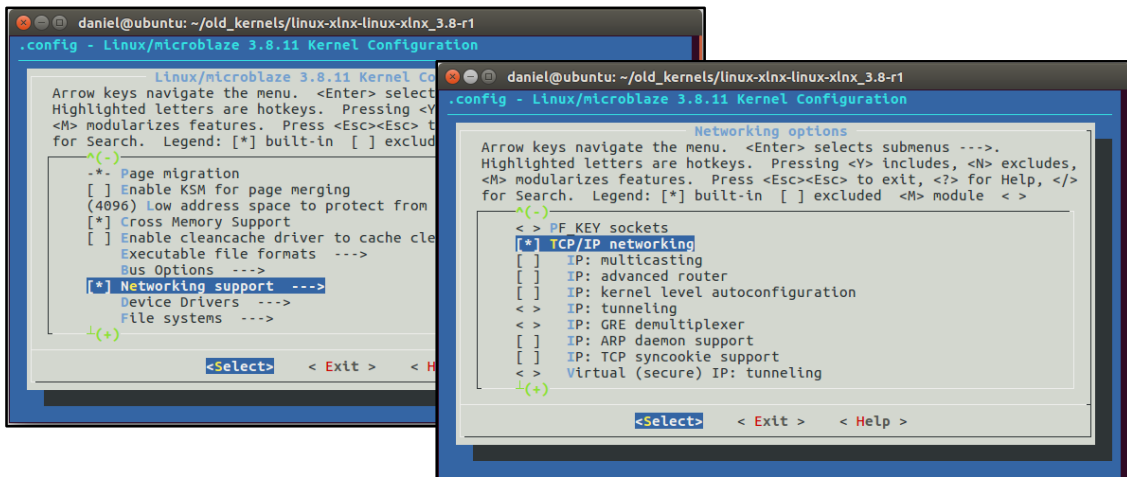


Figura 49: Kernel 3.8-r1 - Ajuste del soporte para Red y ethernet

Tras ajustar el kernel para que disponga de soporte para red, se deben ajustar los drivers del dispositivo de red, en este caso, debe ser compatible con la IP ethernet MAC lite. Para ello, navegando hasta las opciones de drivers ethernet, es necesario activar las dos opciones referentes a Xilinx ethernet MAC lite (no AXI).

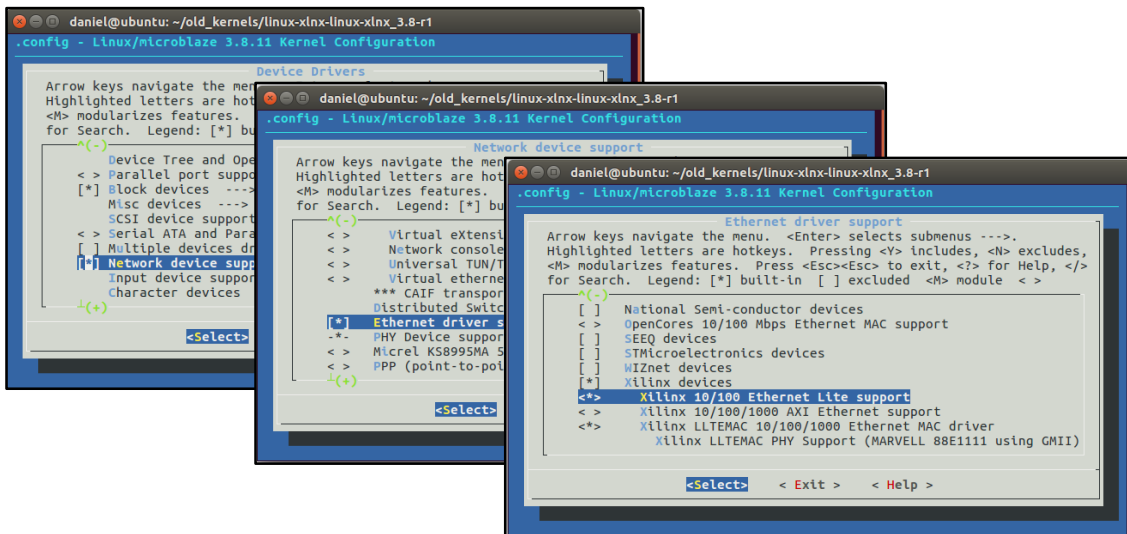


Figura 50: Kernel 3.8-r1 – Ajuste de los drivers ethernet

Tras ajustar las opciones de red y los drivers del chip ethernet, queda habilitar el soporte para GPIO, e indicar al compilador que active el acceso a los diferentes GPIOs desde la consola del usuario.

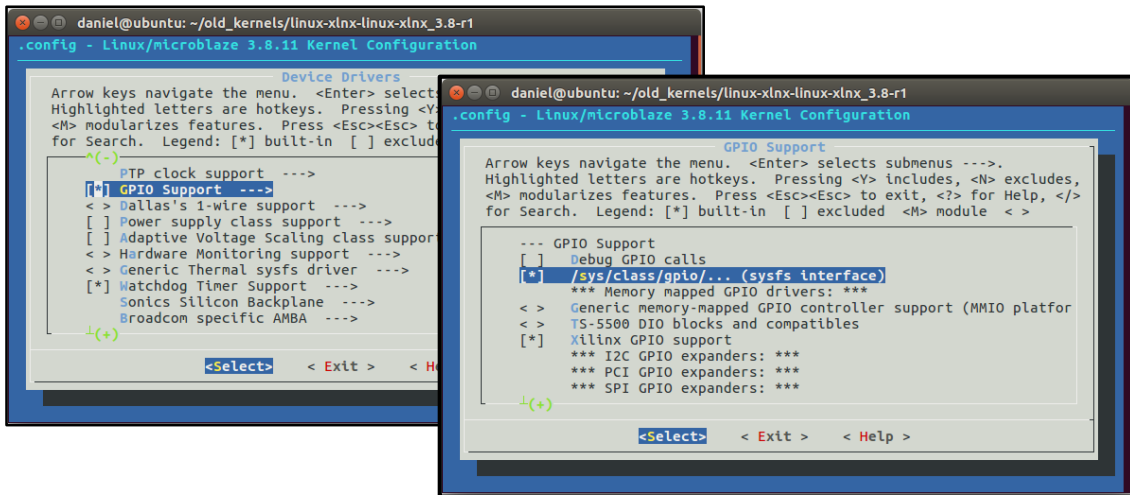


Figura 51: Kernel 3.8-r1 – Ajuste del soporte para GPIOs

Tras realizar todos los ajustes, se debe guardar el fichero de configuración y se puede pasar al siguiente paso, la compilación del kernel.

3.2.4 Compilación del kernel

Una vez guardada la configuración ajustada al diseño de Microblaze, el siguiente paso es compilar el kernel para obtener la imagen que debe cargarse en memoria RAM para arrancar el sistema operativo. Para ello, se deje ejecutar el siguiente comando:

```
make ARCH=microblaze simpleImage.xilinx
```

Donde “xilinx” es el nombre del fichero de descripción de hardware que se había movido anteriormente. Tras finalizar la compilación se muestra la siguiente información y se genera la imagen del kernel Linux.

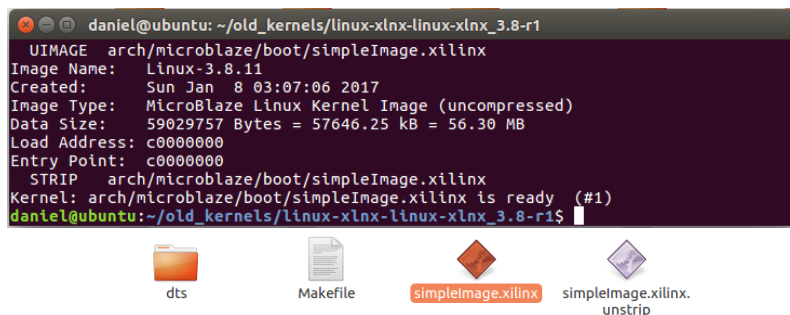


Figura 52: Kernel 3.8-r1 – Final de compilación

En el mensaje final se puede ver información interesante como el tamaño del sistema operativo una vez descomprimido, en este caso 56.3MB, valor suficientemente pequeño para poder cargarlo en la RAM de la placa de desarrollo, que tiene un tamaño de 64MB.

CAPÍTULO 4

IMPLEMENTACIÓN Y TEST DEL DISEÑO REALIZADO

En este cuarto capítulo se muestran los resultados de implementar el hardware Microblaze y cargar el sistema operativo Linux.

Tras implementar el diseño hardware y cargar el sistema operativo se realizan una serie de pruebas para determinar el correcto funcionamiento de los diferentes periféricos y la funcionalidad del sistema de ficheros, así como de las herramientas de busybox.

Los pasos de implementación y test se dividen principalmente en dos apartados, que se muestran a continuación.

4.1 Implementación del diseño

El primer paso consiste en implementar el diseño hardware realizado, en este caso Micorblaze y los contoladores para los diferentes periféricos, para poder implementar el softcore es necesario emplear la herramienta SDK.

Tras implementar el diseño del hardware, se debe descargar el kernel sobre la memoria de la placa de desarrollo, para tal finalidad se emplea también el software SDK, aunque puede emplearse el terminal XMD sin necesidad de abrir la aplicación SDK.

4.1.1 Implementación del hardware

Para realizar la implementación del hardware y la carga del kernel Linux se va a emplear el programador JTAG Xilinx Platform Cable II, cuyo driver no es compatible con Windows 10, por lo que se emplea una máquina virtual con Windows 7 instalado. Una vez se ha configurado la herramienta, tan solo queda cargar el bitstream en la FPGA.

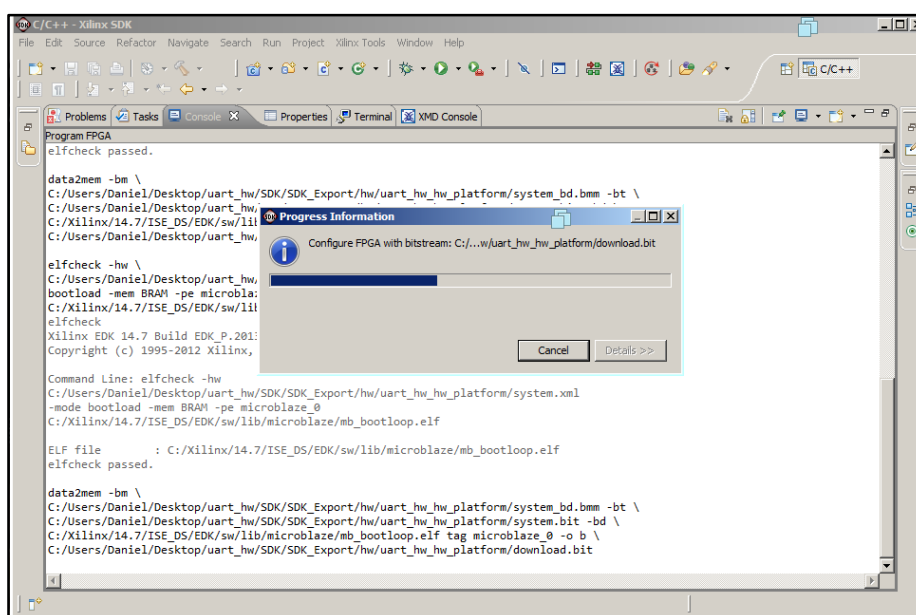


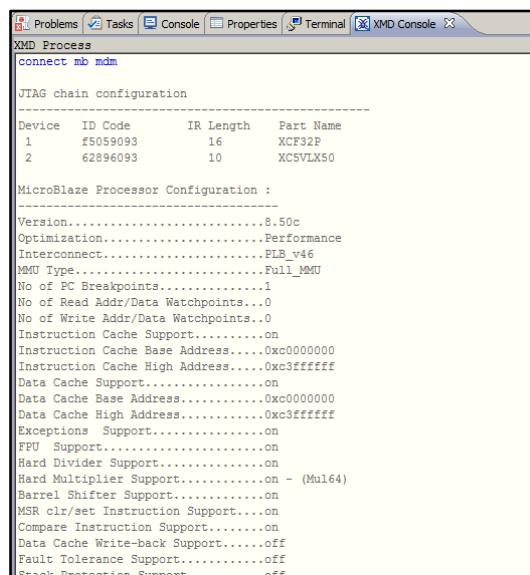
Figura 53: SDK - Programación de la FPGA

Una vez implementado el bitstream (código VHDL de programación de la FPGA), se tiene el microprocesador Microblaze implementado sobre la placa de desarrollo. A continuación, es necesario descargar el sistema operativo Linux en la memoria RAM, para poder ejecutarlo.

4.1.2 Implementación del software

Para descargar el sistema operativo en la placa se emplea el terminal XMD, que permite conectarse al módulo de debug que se había implementado durante el diseño del softcore.

Con el terminal XMD abierto, se ejecuta el comando “connect mb mdm” para conectar con el módulo de debug de Microblaze. Tras conectarse muestra la configuración actual del softcore.



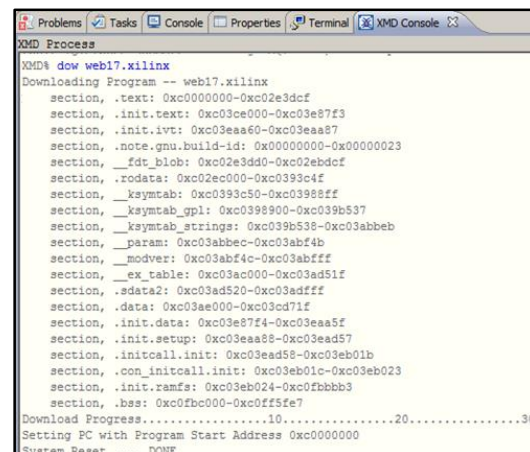
```
XMD Process
connect mb mdm

JTAG chain configuration
-----
Device  ID Code      IR Length  Part Name
-----  -
1       f5059093      16        XCF32P
2       62896093      10        XC5VLX50

MicroBlaze Processor Configuration :
-----
Version.....8.50c
Optimization.....Performance
Interconnect.....PLB_v46
MMU Type.....Full_MMU
No of PC Breakpoints.....1
No of Read Addr/Data Watchpoints...0
No of Write Addr/Data Watchpoints..0
Instruction Cache Support.....on
Instruction Cache Base Address.....0xc0000000
Instruction Cache High Address.....0xc3ffffff
Data Cache Support.....on
Data Cache Base Address.....0xc0000000
Data Cache High Address.....0xc3ffffff
Exceptions Support.....on
FPU Support.....on
Hard Divider Support.....on
Hard Multiplier Support.....on - (Mul64)
Barrel Shifter Support.....on
MSR clr/set Instruction Support.....on
Compare Instruction Support.....on
Data Cache Write-back Support.....off
Fault Tolerance Support.....off
Stack Protection Support.....off
```

Figura 54: XMD - Conexión al módulo de debug

Tras conectar con Microblaze y habiendo copiado el kernel Linux a la máquina con Windows 7, se puede descargar la imagen del sistema operativo sobre la memoria de la placa, para ello se emplea el comando “dow web17.xilinx” donde “web17.xilinx” es el nombre que se le ha dado al kernel tras compilarlo.



```
XMD Process
XMD% dow web17.xilinx
Downloading Program -- web17.xilinx
section, .text: 0xc0000000-0xc02e3dcf
section, .init.text: 0xc03ce000-0xc03e87f3
section, .init.ivt: 0xc03eaa60-0xc03eaa87
section, .note.gnu.build-id: 0x00000000-0x00000023
section, __fdt_blob: 0xc02e3dd0-0xc02ebdcf
section, .rodata: 0xc02ec000-0xc0393c4f
section, __ksymtab: 0xc0393c50-0xc03988ff
section, __ksymtab_gpl: 0xc0398900-0xc039b537
section, __ksymtab_strings: 0xc039b538-0xc03abbef
section, __param: 0xc03abbec-0xc03abf4b
section, __modver: 0xc03abf4c-0xc03abfff
section, __ex_table: 0xc03ac000-0xc03ad51f
section, .sdata2: 0xc03ad520-0xc03adfff
section, .data: 0xc03ae000-0xc03cd71f
section, .init.data: 0xc03e87f4-0xc03eaa5f
section, .init.setup: 0xc03eaa88-0xc03ead57
section, .initcall.init: 0xc03ead58-0xc03eb01b
section, .con_initcall.init: 0xc03eb01c-0xc03eb023
section, .init.ramfs: 0xc03eb024-0xc03fbbb3
section, .bss: 0xc03fb000-0xc03ff5e7
Download Progress.....10.....20.....30
Setting PC with Program Start Address 0xc0000000
System Reset .... DONE
```

Figura 55: XMD - Descarga del kernel sobre la RAM

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

Finalmente, tras cargar el kernel sobre la memoria, ejecutando el comando “con” comienza la ejecución del sistema operativo desde la dirección de memoria 0xC0000000, de ahí que fuese tan importante fijar la dirección de inicio de la memoria RAM durante el diseño del softcore.

Por otra parte, mientras arranca el sistema operativo se puede ver como la consola (en este caso el terminal que muestra los datos del cable RS232) comienza a mostrar los diferentes elementos cargados, versión del kernel, periféricos asignados, etc.



```
COM10 - Tera Term VT
File Edit Setup Control Window Help

Early console on uarilite at 0x84020000
bootconsole [earlyser0] enabled
Ramdisk addr 0x00000000,
Compiled-in FDT at 0xc02e3dd0
Linux version 3.8.11 (daniel@ubuntu) (gcc version 4.6.4 20120924 (Xilinx 14.1 Build EDR_P.13 28 Sep 2013) (crosstool-N6 1.18.0) ) #1 Sun Jan 15 11:45:44 PST 2017
setup_cpuinfo: initialising
setup_cpuinfo: No PVR support. Using static CPU info from FDT
ERROR: Microblaze AH_HUL-different for kernel and DTS
ht_nsr
setup_memory: max_mapnr: 0x4000
setup_memory: min_low_pfn: 0xc0000
setup_memory: max_low_pfn: 0xc4000
setup_memory: max_pfn: 0xc4000
Zone ranges:
  DMA      mem 0xc0000000-0xc3fffffff
  Normal   empty
  HighMem  empty
Movable zone start for each node
Early memory node ranges
  node 0: mem 0xc0000000-0xc3fffffff
On node 0 totalpages: 16384
free_area_init_node: node 0, pgdat c03cc8cc, node_mem_map c0fff000
  DMA zone: 128 pages used for memmap
  DMA zone: 0 pages reserved
  DMA zone: 16256 pages, LIFO batch:3
early_printk console remapping from 0x84020000 to 0xff7ff000
pcpu-alloc: s0 r0 d32768 u32768 alloc=1*32768
pcpu-alloc: [0] 0
Built 1 zonelists in Zone order, mobility grouping on. Total pages: 16256
Kernel command line: console=ttyUL0,115200 root=/dev/ram rw ip=on earlyprintk
PID hash table entries: 256 (order: -2, 1024 bytes)
Dentry cache hash table entries: 8192 (order: 3, 32768 bytes)
Inode-cache hash table entries: 4096 (order: 2, 16384 bytes)
High memory: 0k
Memory: 48476k/65536k available (3768k kernel code, 17060k reserved, 125k data, 231k bss, 12239k init)
Kernel virtual memory layout:
 * 0xffffea000..0xfffff0000 : fixmap
 * 0xffff80000..0xffffc0000 : highmem PTEs
 * 0xffff7ff00..0xffff80000 : early ioremap
 * 0xffff00000..0xffff7ff00 : vmlalloc & ioremap
NR_IRQS:33
interrupt-controller #0 at 0xf0000000, num_irqs=0, edge=0x1d
No chosen timer found, using default
timer #0 at 0xf0002000, irq=2
microblaze_timer_set_mode: shutdown
microblaze_timer_set_mode: periodic
Calibrating delay loop... 49.35 BogoMIPS (lpj=246784)
pid_max: default: 4096 minimum: 301
Mount-cache hash table entries: 512
NET: Registered protocol family 16
bio: create slab <bio-0> at 0
Switching to clocksource microblaze_clocksource
NET: Registered protocol family 2
TCP established hash table entries: 512 (order: 0, 4096 bytes)
TCP bind hash table entries: 512 (order: 1, 10240 bytes)
TCP: Hash tables configured (established 512 bind 512)
TCP: veno registered
UDP hash table entries: 128 (order: 0, 6144 bytes)
UDP-Lite hash table entries: 128 (order: 0, 6144 bytes)
NET: Registered protocol family 1
RPC: Registered named UNIX socket transport module.
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
RPC: Registered tcp NFSv4.1 backchannel transport module.
Skipping unavailable RESET gpio -2 (reset)
audit: initializing netlink socket (disabled)
type=2000 audit(19.260:1): initialized
ROMFS RTD (C) 2007 Red Hat, Inc.
msgmni has been set to 94
io scheduler noop registered
io scheduler deadline registered
io scheduler cfs registered (default)
Serial: 8250/16550 driver, 4 ports, IRQ sharing disabled
84020000.serial: ttyUL0 at MMIO 0x84020000 (irq = 3) is a uarilite
console [ttyUL0] enabled, bootconsole disabled
console [ttyUL0] enabled, bootconsole disabled
84000000.serial: ttyUL1 at MMIO 0x84000000 (irq = 8) is a uarilite
brd: module loaded
xilinx_emaclite 81000000.ethernet: Device Tree Probing
```

Figura 56: Terminal Linux – Consola de arranque

4.2 Test de la parte hardware y software

Con el sistema operativo en ejecución, será necesario realizar algunas pruebas para verificar que tanto el kernel como los diferentes periféricos, en este caso los diferentes GPIOs, las UARTs y Ethernet, funcionan de la forma deseada.

Para empezar las pruebas, se comienza por los diferentes GPIOs, el primer paso es navegar hasta la localización que da acceso a ellos a través del espacio de usuario, que es, “/sys/class/gpio”.

Dentro de esta localización deberían mostrarse un total de cuatro “*gpiochipXXX*”, que corresponden a las cuatro instancias conectadas a Microblaze.

```
# ls
bin dev etc init lib mnt proc sbin sys tmp usr var
# cd /sys/class/gpio
# ls
export gpiochip241 gpiochip245 gpiochip248 gpiochip252 unexport
```

Figura 57: Terminal Linux - GPIOs user space

Viendo que aparecen los cuatro *gpiochip* se procede a realizar algunas pruebas sobre cada una de las instancias, el primer paso es determinar a qué elemento de Microblaze corresponde cada uno.

Para determinar la identidad de cada uno se puede hacer uso de las instrucciones “*cat gpiochipXXX/base*” y “*cat gpiochipXXX/label*” que sirven para determinar a qué dirección del bus PLB está asignado cada chip y cuál es el primer elemento GPIO asignado en Linux.

```
# ls
export gpiochip241 gpiochip245 gpiochip248 gpiochip252 unexport
# cat gpiochip252/base
252
# cat gpiochip252/label
/plb@0/gpio@81460000
```

Figura 58: Terminal Linux - Obtención de información del *gpiochip252*

Ejecutando ambos comandos se determina la siguiente información:

- El *gpiochip252* corresponde a la dirección 81460000
- El primer elemento del chip es el 252

Con esta información y comparando con la definición del hardware anteriormente generada (Figura 44), se puede determinar la siguiente tabla:

gpiochip252	GPIO Linux	Elemento FPGA
plb@81460000	252	SW5-4
	253	SW5-3
DIP SWITCH	254	SW5-2
	255	SW5-1

Tabla 9: Linux - Correspondencia GPIO (dip switch)

Conociendo a que elemento se corresponde, el siguiente paso es realizar un “export” del elemento deseado, de forma que se pueda manipular su estado y configuración.

Dado que los switch son elementos de entrada, se configura el primero de ellos como entrada, en este caso el 252 y posteriormente se lee el valor.

```
# echo 252 > export
# ls
export      gpiochip241  gpiochip248  unexport
gpio252     gpiochip245  gpiochip252
# cd gpio252
# ls
active_low  direction    subsystem    uevent       value
# echo in > direction
# cat value
1
# cat value
1
# cat value
0
# cat value
1
```

Figura 59: Terminal Linux - Configuración y test de los switch buttons

Tras exportar el gpio 252, se muestran los elementos dentro de “/sys/class/gpio”, como se puede ver, ha aparecido un nuevo elemento, que permite controlar el elemento GPIO 252, para ello se debe configurar como entrada, cambiando a su directorio específico y ejecutando “echo in > direction”, tras configurarlo como entrada es posible leer su valor empleado “cat value”, lo cual devuelve 0 si está a nivel bajo y 1 si está a nivel alto. Variando algunas veces el interruptor número 4 y leyendo el estado se puede comprobar que la instancia de los dip switch funciona correctamente en Linux.

Una vez comprobada la primera instancia de las cuatro, se procede a testear el segundo elemento, el gpiochip245.

```
# ls
export      gpiochip241  gpiochip248  unexport
gpio252     gpiochip245  gpiochip252
# cat gpiochip245/base
245
# cat gpiochip245/label
/plb00/gpio081420000
```

Figura 60: Terminal Linux - Obtención de información del gpiochip245

De igual forma que en el caso anterior, se pueden obtener los datos consultando la etiqueta y dirección base del gpiochip245. Con esta información se puede determinar la siguiente tabla:

gpiochip245	GPIO Linux	Elemento FPGA
plb@81420000	245	SW4
	246	SW3
PUSH BUTTONS	247	SW2
	-	-

Tabla 10: Linux - Correspondencia GPIO (push buttons)

Para probar el funcionamiento de los pulsadores se emplea el mismo metodo que antes, configurando un par de ellos como entrada y posteriormente pulsándolos para leer si cambian de estado o no, a continuación se muestra la configuración del gpio 245 y 246, así como la lectura de sus valores.


```
# ls
export      gpiochip241  gpiochip248  unexport
gpio252    gpiochip245  gpiochip252
# cat gpiochip245/base
245
# cat gpiochip245/label
/plb00/gpio@81440000
# echo 245 > export
# ls
export      gpio252    gpiochip245  gpiochip248  gpiochip252
gpio245    gpiochip241  gpiochip248  unexport
# echo in > gpio245/direction
# cat gpio245/value
0
# cat gpio245/value
1
# cat gpio245/value
0
# echo 246 > export
# ls
export      gpio246    gpiochip241  gpiochip248  unexport
gpio245    gpio252    gpiochip245  gpiochip252
# echo in > gpio246/direction
# cat gpio246/value
0
# cat gpio246/value
1
# cat gpio246/value
1
#
```

Figura 61: Terminal Linux - Configuración y test de los push buttons

Dado que ambos botones responden de la forma esperada se pasa a realizar el test del tercer elemento GPIO, el gpiochip248.

El proceso es el mismo que para los dos casos anteriores, el primer paso es la obtención de los valores de correspondencia entre la descripción de hardware y el chip en Linux, y posteriormente se realiza la configuración y test del elemento.

```
# ls
export      gpio246    gpiochip241  gpiochip248  unexport
gpio245    gpio252    gpiochip245  gpiochip252
# cat gpiochip248/label
/plb00/gpio@81440000
# cat gpiochip248/base
248
```

Figura 62: Terminal Linux - Obtención de información del gpiochip248

Con información obtenida del terminal se crea la tabla para este chip:

gpiochip248	GPIO Linux	Elemento FPGA
plb@81440000	248	D4
	249	D3
LEDs	250	D2
	251	D1

Tabla 11: Linux - Correspondencia GPIO (LEDs)

Con la información de correspondencia de cada uno de los LEDs se pueden configurar como salidas y posteriormente encender cada uno de ellos, para testarlos se han encendido primeramente el LED D1 y D3, y a continuación el D4.

```
# ls
export      gpio246    gpiochip241  gpiochip248  unexport
gpio245    gpio252    gpiochip245  gpiochip252
# cat gpiochip248/label
/plb00/gpio081440000
# cat gpiochip248/base
248
# echo 248 > export
# echo 249 > export
# echo 250 > export
# echo 251 > export
# echo out > gpio248/direction
# echo out > gpio249/direction
# echo out > gpio250/direction
# echo out > gpio251/direction
# echo 1 > gpio251/value
# echo 1 > gpio249/value
# echo 1 > gpio248/value
```

Figura 63: Configuración y test de los LEDs

El funcionamiento de los LEDs se muestra en la siguiente figura, lo cual determina que funcionan correctamente y se puede pasar a analizar el último de los GPIOs.



Figura 64: Placa de desarrollo - Test del GPIO de los LEDs

Para acabar con el test de GPIOs es necesario comprobar el funcionamiento del gpiochip241, para ello se sigue la misma metodología que en los casos anteriormente expuestos.

Primero se debe determinar la correspondencia del chip, tal y como muestra la siguiente figura.

```
# ls
export      gpio248    gpio251    gpiochip245  unexport
gpio245    gpio249    gpio252    gpiochip248
gpio246    gpio250    gpiochip241  gpiochip252
# cat gpiochip241/label
/plb00/gpio081440000
# cat gpiochip241/base
241
```

Figura 65: Obtención de información del gpiochip241

Tras obtener los valores se puede obtener la tabla, empleando para ello la descripción de hardware y el fichero de restricciones del diseño.

gpiochip241	GPIO Linux	Elemento FPGA	PIN JP11
plb@81400000	241	T7	17
	242	T5	15
GPIOs	243	W6	39
	244	V7	37

Tabla 12: Correspondencia GPIO (GPIOs)

Configurando el primero de los GPIO como salida se puede determinar si funciona correctamente, variando para ello su salida entre 0 y 3.3V.

```
# ls
export      gpio248    gpio251    gpiochip245 unexport
gpio245     gpio249    gpio252    gpiochip248
gpio246     gpio250    gpiochip241 gpiochip252
# cat gpiochip241/label
/p1b00/gpio081400000
# cat gpiochip241/base
241
# echo 241 > export
# echo out > gpio241/direction
# echo 0 > gpio241/value
# echo 1 > gpio241/value
```

Figura 66: Configuración y test de los GPIOs

Conectando un multímetro al pin 17 del conector JP11 se puede ver que el valor de tensión varía entre 0V y 3.3V, por lo tanto, este último chip funciona correctamente.

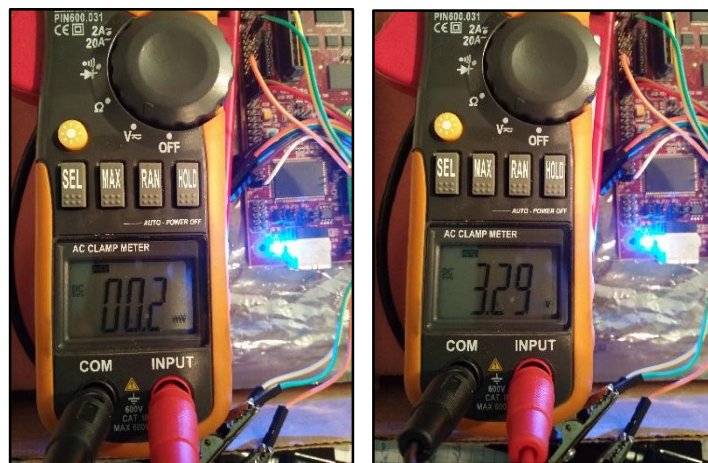


Figura 67: Placa de desarrollo - Test del GPIO asignados a JP11

Tras verificar el correcto funcionamiento de los diferentes chips GPIO en Linux, queda por comprobar el funcionamiento de la segunda UART (ttyUL1) y las herramientas de BusyBox incluidas en el sistema de ficheros.

Cabe destacar que el funcionamiento de la primera UART (ttyUL0) ha quedado demostrado ya que se ha podido comunicar sin problemas empleado el terminal de Linux, vinculado al puerto RS232 y desde donde se están enviando todos los comandos que se han mostrado hasta ahora.

Para realizar el test de BusyBox se comprueban primero que herramientas están instaladas, para ello es suficiente con ejecutar el comando “busybox”.

```
# busybox
BusyBox v1.13.2 (2009-10-01 13:02:35 MDT) multi-call binary
Copyright (C) 1998-2008 Erik Andersen, Rob Landley, Denys Vlasenko
and others. Licensed under GPLv2.
See source distribution for full notice.

Usage: busybox [function] [arguments]...
or: function [arguments]...

BusyBox is a multi-call binary that combines many common Unix
utilities into a single executable. Most people will create a
link to busybox for each function they wish to use and BusyBox
will act like whatever it was invoked as!

Currently defined functions:
[, ll, addgroup, adduser, adjtimex, ar, arp, arping, ash,
auk, basename, blkid, brctl, bunzip2, bzip2, cal,
cat, catv, chat, chatr, chgrp, chmod, chown, chpasswd,
chpst, chroot, chrt, chvt, cksum, clear, cmp, comm, cp,
cpio, crond, crontab, cryptpw, ctyhack, cut, date, dc,
dd, dealloctv, delgroup, deluser, depmod, devmem, df, dhcprelay,
diff, dirname, dmesg, dnsd, dos2unix, du, dumpkmap, dumpleases,
echo, ed, egrep, eject, env, envdir, envuidgid, ether-uake,
expand, expr, fakeidentd, false, fbset, fbsplash, fdflush,
fdformat, fdisk, fgrep, find, findfs, fold, free, freerandisk,
fsck, fsck.ninux, ftpget, ftpput, fuser, getopt, getty,
grep, gunzip, gzip, halt, hd, hdparm, head, hexdump, hostid,
hostname, httpd, hush, huclock, id, ifconfig, ifdown, ifenslave,
ifup, inetd, init, insmod, install, ip, ipaddr, ipcalc,
ipcrm, ipcs, iplink, iproute, iprule, iptunnel, kbd mode,
kill, killall, killall5, klogd, last, length, less, linux32,
linux64, linuxrc, ln, loadfont, loadkmap, logger, login,
logname, logread, losetup, lpd, lpq, lpr, ls, lsattr, lsmod,
lznacat, nakedevs, makerime, nan, nd5sun, ndev, msg, microcom,
mkdir, mkfifo, mkfs.ninux, mknod, mkswap, mktemp, modprobe,
more, mount, mountpoint, msh, ni, nv, nameif, nc, netstat,
nice, nmeter, nohup, nslookup, od, openvt, passwd, patch,
pgrep, pidof, ping, ping6, pipe_progress, pivot_root, pkill,
popmaildir, poweroff, printenv, printf, ps, pscan, pud,
raidautorun, rdate, rdev, readahead, readlink, readprofile,
realpath, reboot, reformime, renice, reset, resize, rn,
rmdir, rmod, route, rtcwake, run-parts, runlevel, runsv,
runsvdir, rx, script, sed, sendmail, seq, setarch, setconsole,
setfont, setkeycodes, setlogcons, setuid, setuidgid, sh,
shalsun, shoukey, slattach, sleep, softlimit, sort, split,
start-stop-daemon, stat, strings, stty, su, sulogin, sun,
sv, svlogd, swapon, switch_root, sync, sysctl,
syslogd, tac, tail, tar, tcpsvd, tee, telnet, telnetd, test,
tftp, tftpd, time, top, touch, tr, traceroute, true, tty,
ttysize, udhcpc, udhcpd, udsvd, umount, uname, uncompress,
unexpand, uniq, unix2dos, unlzma, unzip, uptime, usleep,
uudecode, uuencode, vconfig, vi, vlock, watch, watchdog,
wc, wget, which, who, whoami, xargs, yes, zcat, zcip
```

Figura 68: Terminal Linux - Herramientas de BusyBox

Tal y como muestra la anterior figura, viene incluido el intérprete de *“awk”* y *“shell”*, herramientas muy útiles como *vi*, que es un editor de texto, la más que conocida *“ping”*, otras herramientas como *“arping”*, un pequeño servidor web, que es *“httpd”*, y múltiples funcionalidades para el kernel, como *“mount”* y *“chmod”* o *“chown”*, herramientas de compresión para *zip*, una pequeña aplicación para controlar la comunicación serie, como es el caso de *“microcom”* y un largo *etcétera*.

Tras comprobar que elementos estaban disponibles con la instalación de BusyBox incluida en el sistema de ficheros empleado, se procede a ejecutar algunos comandos que devuelvan información útil sobre el kernel y el hardware del dispositivo sobre el que se ejecuta, en este caso Microblaze.

Para mostrar las características del hardware se puede emplear el comando *“cat proc/cpuinfo”*, lo cual devuelve el tipo de microprocesador, su velocidad y el hardware del que dispone internamente, como por ejemplo el multiplicador, barrel shifter, etc.

```
# cat /proc/cpuinfo
CPU-Family:      MicroBlaze
FPGA-Arch:      virtex5
CPU-Ver:        Unknown, big endian
CPU-MHz:        100.00
BogoMips:       49.35
MMU:
  Shift:        yes
  MSR:          yes
  PCMP:         yes
  DIV:          yes
  MMU:          3
  MUL:          v2
  FPU:          v1
  Exc:
Stream-insns:   privileged
Icache:         16kB   line length: 16B
Dcache:         16kB   line length: 16B
Dcache-Policy:  write-through
MM-Debug:       yes
PVR-USR1:       00
PVR-USR2:       00000000
Page size:      4096
#
```

Figura 69: Terminal Linux – Información del procesador

Por tanto, empleando el anterior comando se puede ver información ya conocida, dado que son los elementos que se han configurado durante el desarrollo de la parte hardware.

Tras ver la información referente al procesador se puede ver la información sobre la memoria, en este caso RAM, ya que es donde está alojado el sistema operativo. Para ver la información sobre la memoria se emplea el comando “cat /proc/meminfo”, tras ejecutarlo se puede ver la memoria total, el espacio libre, etc.

```
# cat /proc/meminfo
MemTotal:       60716 kB
MemFree:        20424 kB
Buffers:         0 kB
Cached:         37580 kB
SwapCached:     0 kB
Active:         860 kB
Inactive:       36860 kB
Active(anon):   140 kB
Inactive(anon): 0 kB
Active(file):   720 kB
Inactive(file): 36860 kB
Unswappable:   0 kB
Mlocked:       0 kB
HighTotal:     0 kB
HighFree:      0 kB
LowTotal:      60716 kB
LowFree:       20424 kB
SwapTotal:     0 kB
SwapFree:      0 kB
Dirty:         0 kB
Writeback:     0 kB
AnonPages:     164 kB
Mapped:        520 kB
Shmem:         0 kB
Slab:          2148 kB
SReclaimable:  768 kB
SUnreclaim:   1380 kB
KernelStack:  200 kB
PageTables:    0 kB
MFS_Unstable:  0 kB
Bounce:        0 kB
WritebackTmp:  0 kB
CommitLimit:   30356 kB
Committed_AS:  1444 kB
VmallocTotal:  253948 kB
VmallocUsed:    420 kB
VmallocChunk:  253240 kB
```

Figura 70: Terminal Linux – Información de la memoria

Tras ver la información referente al hardware, se puede ver la versión del kernel que se está empleando, además del usuario que lo ha compilado e incluso con que versión del compilador se ha hecho. Para ello se ejecuta el comando “cat /proc/version”.

```
# cat /proc/version
Linux version 3.8.11 (daniel@ubuntu)
(gcc version 4.6.4 20120924 (Xilinx 14.1 Build EDK_P.13 28 Sep 2013) (crosstool-MG 1.18.0) )
#1 Sun Jan 8 03:06:58 PST 2017
```

Figura 71: Terminal Linux – Información del kernel

Con el anterior comando se puede ver la versión del kernel, el usuario que lo ha compilado y bajo qué distribución de Linux, la versión del compilador y la fecha de compilación.

Una vez vistas las características de la memoria, la cpu y el propio sistema operativo se pasa al análisis del funcionamiento del controlador del puerto ethernet.

Primero de todo, empleando la instrucción “ifconfig –a” se puede ver que elementos de red están registrados en el sistema.

```
# ls
bin dev etc init lib mnt proc sbin sys tmp usr var
# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:0A:35:00:00:00
          BROADCAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:4 Memory:81000000-8100ffff

lo        Link encap:Local Loopback
          LOOPBACK MTU:65536 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Figura 72: Terminal Linux – Interfaces de red

En la anterior figura se puede ver que aparece la interfaz “eth0”, que está asociada al hardware que se había implementado para controlar la interfaz ethernet, sin embargo, no está configurada puesto que no aparece la dirección IP.

Cabe destacar que la placa de desarrollo está conectada vía ethernet a un repetidor WIFI.

Para configurar la interfaz es suficiente con ejecutar el comando “ifconfig eth0 192.168.1.200 up” donde el valor numérico corresponde a la ip deseada. Tras ejecutarlo se puede ver como la interfaz pasa a estar activa, según indica el sistema operativo, a una velocidad máxima de 100 Mbps, en modo full-dúplex.

```
# ifconfig eth0 192.168.1.200 up
# libphy: 81000000:07 - Link is Up - 100/Full
```

Figura 73: Terminal Linux – Configuración de eth0

Una vez configurada la interfaz se procede a realizar algunas pruebas para determinar si funciona como debería.

La primera prueba consiste en hacer ping con 10 paquetes hacia la dirección IP del router, que es la 192.168.1.1.

```
ping -c 10 192.168.1.1
PING 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=3 ttl=64 time=12.483 ms
64 bytes from 192.168.1.1: seq=4 ttl=64 time=3.150 ms
64 bytes from 192.168.1.1: seq=5 ttl=64 time=9.960 ms
64 bytes from 192.168.1.1: seq=6 ttl=64 time=2.477 ms
64 bytes from 192.168.1.1: seq=7 ttl=64 time=2.316 ms
64 bytes from 192.168.1.1: seq=8 ttl=64 time=6.047 ms
64 bytes from 192.168.1.1: seq=9 ttl=64 time=12.077 ms

--- 192.168.1.1 ping statistics ---
10 packets transmitted, 7 packets received, 30% packet loss
round-trip min/avg/max = 2.316/6.930/12.483 ms
```

Figura 74: Terminal Linux – Ping hacia el router

Algunos de los paquetes se han perdido, por lo que el siguiente paso es realizar ping a alguna de las máquinas conectadas a la red local, en este caso, el ordenador de sobremesa, que está conectado al router mediante WIFI empleando un adaptador USB. El primer paso es determinar la dirección IP del ordenador, para ello se emplea el terminal cmd de Windows y la instrucción “ipconfig/all”.

```
Descripción . . . . . : D-Link DWA-140 Wireless N USB Adapter(rev.D)
Dirección física. . . . . : EC-22-80-0B
DHCP habilitado . . . . . : sí
Configuración automática habilitada . . . : sí
Vínculo: dirección IPv6 local. . . : fe80::8136:2962:a9ea:bbee%21(Preferido)
Dirección IPv4. . . . . : 192.168.1.123(Preferido)
Máscara de subred . . . . . : 255.255.255.0
```

Figura 75: Terminal CMD Windows – Dirección IP

Una vez conocida la dirección de la máquina se pueden enviar 10 paquetes como antes para ver si alguno de ellos se pierde, además la utilidad ping, se encarga de calcular el valor del RTT (Round Trip Time).

```
# ping -c 10 192.168.1.123
PING 192.168.1.123 (192.168.1.123): 56 data bytes
64 bytes from 192.168.1.123: seq=0 ttl=128 time=6.868 ms
64 bytes from 192.168.1.123: seq=1 ttl=128 time=3.105 ms
64 bytes from 192.168.1.123: seq=2 ttl=128 time=1.863 ms
64 bytes from 192.168.1.123: seq=3 ttl=128 time=2.202 ms
64 bytes from 192.168.1.123: seq=4 ttl=128 time=1.943 ms
64 bytes from 192.168.1.123: seq=5 ttl=128 time=2.722 ms
64 bytes from 192.168.1.123: seq=6 ttl=128 time=1.889 ms
64 bytes from 192.168.1.123: seq=7 ttl=128 time=1.870 ms
64 bytes from 192.168.1.123: seq=8 ttl=128 time=1.900 ms
64 bytes from 192.168.1.123: seq=9 ttl=128 time=3.483 ms

--- 192.168.1.123 ping statistics ---
10 packets transmitted, 10 packets received, 0% packet loss
round-trip min/avg/max = 1.863/2.784/6.868 ms
```

Figura 76: Terminal Linux – Ping hacia el ordenador Windows

Haciendo ping al equipo de sobremesa no se pierde ningún paquete, por lo que parece que la conexión ethernet funciona correctamente, para asegurarlo se realizan dos pruebas más. La primera consiste en emplear el protocolo ARP para determinar la dirección física (MAC address) del router, para ello se ejecuta el comando “arping 192.168.1.1”. Lo que resulta en la siguiente figura.

```
# arping 192.168.1.1
ARPING to 192.168.1.1 from 192.168.1.200 via eth0
Unicast reply from 192.168.1.1 [38:d5:7:84:41:0] 11.269ms
Unicast reply from 192.168.1.1 [38:d5:7:84:41:0] 13.840ms
Unicast reply from 192.168.1.1 [38:d5:7:84:41:0] 9.267ms
Unicast reply from 192.168.1.1 [38:d5:7:84:41:0] 7.666ms
Unicast reply from 192.168.1.1 [38:d5:7:84:41:0] 336.438ms
Unicast reply from 192.168.1.1 [38:d5:7:84:41:0] 8.290ms
Unicast reply from 192.168.1.1 [38:d5:7:84:41:0] 3.051ms
```

Figura 77: Terminal Linux – ARP de la IP del router

La comunicación empleando ARP funciona correctamente en la FPGA y la dirección física que devuelve se corresponde MAC del router para el punto de acceso WIFI de 2.4GHz.

Finalmente, para concluir con las pruebas realizadas sobre la conexión ethernet, se hace ping desde el equipo con Windows hacia la FPGA, empleando otra vez el terminal CMD y esta vez, la instrucción “ping -n 10 192.168.1.200”, que enviará un total de 10 paquetes a través de la red local hacia la ip de la placa de desarrollo.

```
C:\Users\vaque>ping -n 10 192.168.1.200

Haciendo ping a 192.168.1.200 con 32 bytes de datos:
Respuesta desde 192.168.1.200: bytes=32 tiempo=3ms TTL=64
Respuesta desde 192.168.1.200: bytes=32 tiempo=6ms TTL=64
Respuesta desde 192.168.1.200: bytes=32 tiempo=3ms TTL=64
Respuesta desde 192.168.1.200: bytes=32 tiempo=2ms TTL=64
Respuesta desde 192.168.1.200: bytes=32 tiempo=2ms TTL=64
Respuesta desde 192.168.1.200: bytes=32 tiempo=3ms TTL=64
Respuesta desde 192.168.1.200: bytes=32 tiempo=2ms TTL=64
Respuesta desde 192.168.1.200: bytes=32 tiempo=2ms TTL=64
Respuesta desde 192.168.1.200: bytes=32 tiempo=4ms TTL=64
Respuesta desde 192.168.1.200: bytes=32 tiempo=3ms TTL=64

Estadísticas de ping para 192.168.1.200:
    Paquetes: enviados = 10, recibidos = 10, perdidos = 0
              (0% perdidos),
    Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 2ms, Máximo = 6ms, Media = 3ms
```

Figura 78: Terminal CMD Windows – Ping hacia la FPGA

Tras enviar los 10 paquetes, ninguno de ellos se pierde por lo que se concluyen las pruebas de la interfaz ethernet con resultados positivos.

A continuación, queda realizar las pruebas sobre la UART restante, primero de todo es necesario configurarla para que sea accesible empleando el dispositivo “ttyUL1” de Linux, ya que el dispositivo “ttyUL0” es la otra UART, que se encarga de transmitir la entrada/salida de Windows por RS232.

Para indicar a Linux que existe una segunda UART es necesario emplear la instrucción “mknod”, sin embargo, primero se deben conocer los parámetros de dicho elemento, para obtener los parámetros se debe navegar hasta el directorio “/dev/tty/ttyUL1” y leer el valor del fichero “dev”.

```
# cd ttyUL1
# ls
close_delay      flags           line           uevent
closing_wait    io_type        port          xmit_fifo_size
custom_divisor  iomen_base     subsystem
dev             iomen_reg_shift type
device         irq           uartclk
# cat dev
204:188
```

Figura 79: Terminal Linux – Parámetros del dispositivo ttyUL1

Con los datos que devuelve el comando, se puede ejecutar el comando “*mknod c 204 188*” que crea el dispositivo dentro de “*/dev*” para que sea accesible desde el entorno de usuario. La siguiente figura muestra la configuración del dispositivo, así como la sintaxis de la instrucción “*mknod*”.

```
# mknod
BusyBox v1.13.2 (2009-10-01 13:02:35 MDT) multi-call binary

Usage: mknod [OPTIONS] NAME TYPE MAJOR MINOR

Create a special file (block, character, or pipe)

Options:
  -m      Create the special file using the specified mode (default a=rw)
TYPES include:
  b:      Make a block device
  c or u: Make a character device
  p:      Make a named pipe (MAJOR and MINOR are ignored)

# mknod /dev/ttyUL1 c 204 188
# ls
close_delay      flags            line            uevent
closing_wait     io_type         port            xmit_fifo_size
custom_divisor   ionen_base      subsystem
dev              ionen_reg_shift type
device           irq             uartclk
# cd
# cd /dev
# ls
console  nen      ptyp0  random  tty2    ttyUL0  ttyp2
i2c      null    ptyp1  tty     tty3    ttyUL1  ssal
log      ptmx   ptyp2  tty0    tty4    tty0    zero
loop     pts    ran    tty1    ttyS0  tty1
```

Figura 80: Terminal Linux - Configuración del dispositivo *ttyUL1*

Tras la configuración, para determinar su funcionamiento se realiza un pequeño script empleando la herramienta de BusyBox “*vi*”, que no es más que un editor de texto. Empleando el comando “*vi read.sh*” se abre en el editor un fichero nuevo con el nombre “*read.sh*”, dentro de él se escribe lo siguiente:

```
microcom -t 500 /dev/ttyUL1 > uart.log &
echo h > /dev/ttyUL1
sleep 0.5
received=$(cat uart.log)
echo $received
```

Figura 81: Terminal Linux – *vi read.sh*

El script se encarga de abrir la comunicación serie del puerto *ttyUL1* y guardar todo lo que sea recibido por la *uart* en un fichero de texto durante los siguientes 500ms, mientras tanto, se envía por el puerto serie el carácter “*h*”.

Trascurridos los 500ms se guarda el texto del fichero en una variable y a continuación se muestra por el terminal.

Conectando a la *uart* un Arduino que se encarga solamente de responder “*Hello*” cada vez que recibe el carácter “*h*”, la ejecución del script se muestra a continuación.

```
# sh read.sh
Hello
#
```

Figura 82: Terminal Linux – Ejecución del script *read.sh*

Tras ejecutar el script se puede ver que la respuesta es la esperada, se han guardado los datos recibidos por la *uart*, procedentes del Arduino, y el resultado es “*Hello*”, por lo

que queda probado el correcto funcionamiento de la uart.

Tras concluir las diferentes pruebas de los periféricos y de las herramientas incluidas con el sistema de ficheros se puede determinar que todos los periféricos asociados a Microblaze funcionan correctamente y el sistema operativo Linux no tiene ningún problema de ejecución sobre el softcore, por lo que el siguiente paso en el desarrollo es la creación de una aplicación capaz de ejecutarse en Linux y verificar que el diseño completo funciona en cuanto a periféricos y control de los mismos se refiere, aportando además alguna utilidad para el usuario final.

CAPÍTULO 5

DISEÑO LA APLICACIÓN LINUX

Este quinto y último capítulo está enfocado a la creación y test de la aplicación Linux que demuestre el correcto funcionamiento del conjunto microprocesador, FPGA y Linux, teniendo en cuenta que debe añadir funcionalidad al sistema.

Para ello, se divide el capítulo en tres puntos principales, el primero se encarga de dar a conocer la aplicación propuesta para el sistema, y los rasgos generales de la funcionalidad que debe desarrollar.

El segundo, se centra en el diseño de la aplicación, creando para ello diferentes diagramas de flujo para cada uno de los elementos, partiendo por tanto de las condiciones propuestas en el punto anterior.

Y finalmente, el tercero, muestra las diferentes capacidades de la web y el correcto funcionamiento de los diferentes elementos que interactúan con la FPGA.

5.1 Aplicación propuesta

El primer punto del desarrollo de la aplicación consiste en definir las características que debe tener, y decidir que hardware es el necesario para la tarea concreta. Por tanto, a continuación, se define en qué consiste la aplicación, cuál es su funcionamiento deseado y qué hardware será necesario para su correcto funcionamiento.

Para aprovechar las características del hardware diseñado, que incluye tanto entradas y salidas, como Ethernet y UART, por tanto, se va a diseñar una aplicación capaz de interactuar con todos ellos.

La aplicación propuesta consiste en ejecutar un servidor web desde la propia FPGA usando Linux, gracias a este servidor web se va a ejecutar un script que permita al usuario interactuar con las entradas/salidas y visualizar información proveniente de algunos sensores.

La aplicación más interesante de cara a crear el servidor web es la de mostrar valores de diferentes sensores para la monitorización de una habitación, mostrando valores como la temperatura, humedad, estado de iluminación exterior y estado de las persianas eléctricas.

Por otro lado, se puede dotar a la web de capacidad de interacción con el usuario, de forma que se añadan una serie de botones para permitir la configuración de algunas de las opciones.

Las tareas a ejecutar por la web son las siguientes:

- Monitorización:
 - Monitorización de la temperatura de la habitación
 - Activación del sistema de calefacción en caso de ser necesario
 - Monitorización del estado de iluminación exterior y persianas
 - Subida o cierre de las persianas en caso de ser necesario
 - Monitorización del movimiento dentro de la habitación

- Interacción con el usuario:
 - Permitir variar el modo de funcionamiento de la calefacción entre manual y automático, fijando para ello una temperatura deseada
 - Permitir variar el modo de funcionamiento de las persianas, entre manual y automático, escogiendo para ello la posición deseada.
 - Permitir activar o desactivar las luces de la habitación

5.2 Diseño de la aplicación

Para poder diseñar la aplicación, primero es necesario decidir cuál debe ser su funcionamiento principal. Empleando un flujograma, se puede entender de forma sencilla la aplicación, la siguiente figura muestra el flujograma que describe el funcionamiento de la web desempeñadas, así como cada una de las acciones de control sobre las persianas, luces y calefacción.

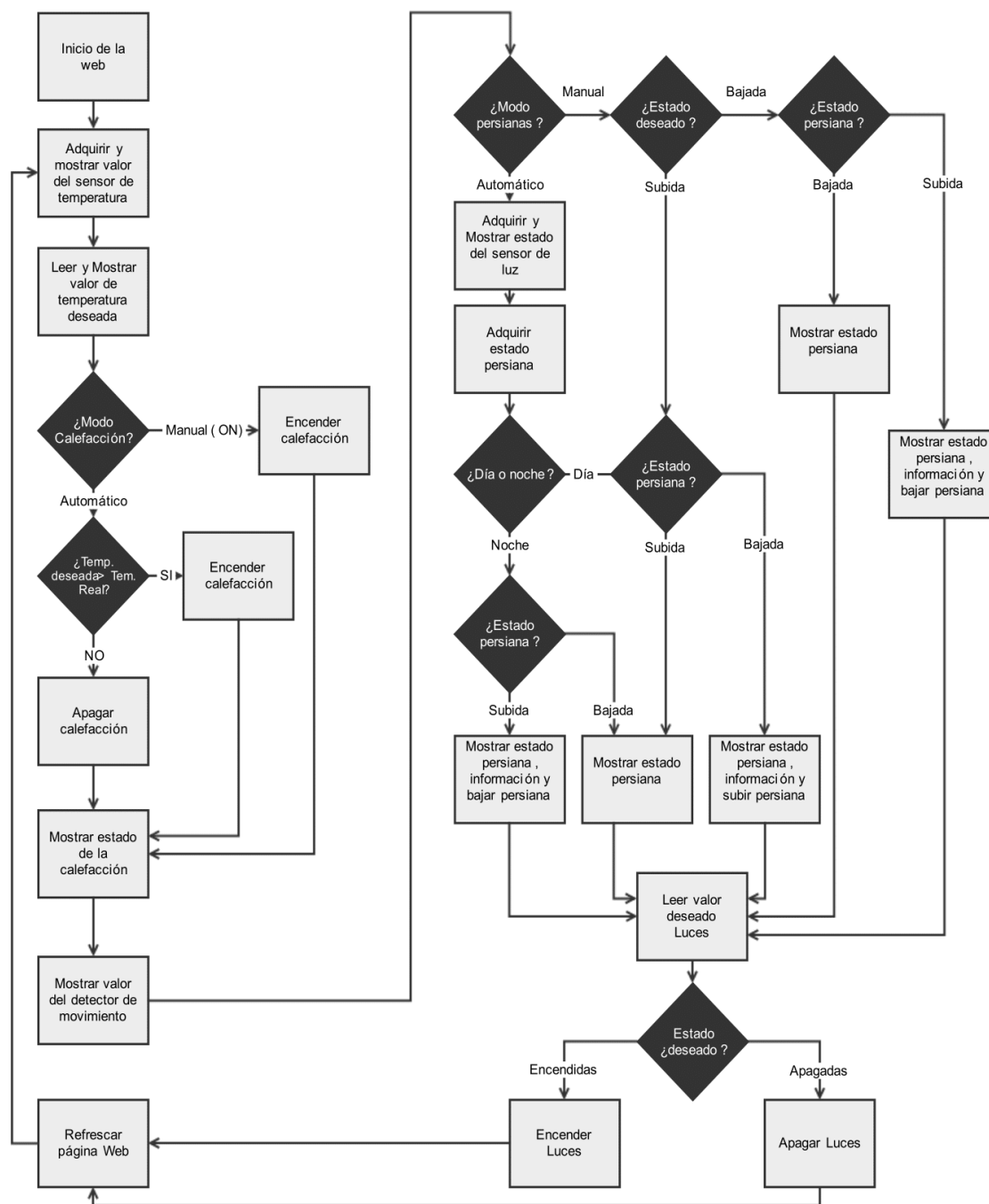


Figura 83: Flujo de funcionamiento de la aplicación Linux

Para realizar la tarea de medida y monitorización, se emplea el hardware mostrado a continuación, se enumeran a los diferentes sensores encargados de la tarea de adquisición de datos y los diferentes elementos encargados de procesar esta información.

5.2.1 Hardware empleado

Para realizar las tareas descritas en el flujo de funcionamiento, se emplea una serie de hardware que permite realizar la adquisición de todos los parámetros y activar las salidas de potencia necesarias.

A continuación, por orden de aparición en el diagrama de flujo se puede ver su utilidad dentro del sistema, así como sus características.

- Sensor de temperatura AM2302: Permite obtener el valor de temperatura y humedad de la sala en cada instante, su resolución es de 0.1°C y 1% de humedad, además se comunica usando un protocolo digital por lo que solamente necesita un cable para comunicación y la alimentación.



Figura 84: Sensores - AM2302

- Detector de movimiento PIR HC-SR501: Su cometido es detectar si existe movimiento dentro de la habitación, se puede configurar para que la salida se actualice entre 1 y 200 segundos, activa una salida digital al detectar movimiento, por lo que igual que en el anterior caso es necesario un pin digital y la alimentación.



Figura 85: Sensores - HC-SR501

- 2 Finales de carrera: Se emplean para saber si la persiana está subida o bajada, cada uno de ellos se situarían en la parte superior de la ventana y en la parte inferior, respectivamente.



Figura 86: Sensores - Final de carrera mecánico

- LDR: Su tarea es medir la cantidad de luz en cada instante, ya que empleando un valor límite se puede decidir si es de día o de noche, para hacer su lectura hace falta realizar un divisor de tensión y disponer de una entrada del ADC.



Figura 87: Sensores - LDR (light dependent resistor)

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA
Daniel Vaquerizo Cid

- Dos módulos de dos relés: Se encargan de cambiar el estado de la salida entre dos estados gracias a la señal digital de la placa FPGA, lo que permite controlar elementos cuya tensión de alimentación sea de hasta 230V a.c.



Figura 88: Actuadores - HL-52 v1.0

- Arduino nano: Dado que Microblaze no cuenta con ningún ADC, se emplea para enviar el valor de los diferentes sensores a través del puerto de comunicación serie, hacia la placa de desarrollo con la FPGA.



Figura 89: Arduino nano

- Placa de desarrollo Virtex 5: Su finalidad es clara, se encarga de recibir los datos deseados en cada instante, además activa y desactiva las salidas físicas del sistema mientras ejecuta la aplicación Linux a modo de servidor web local.

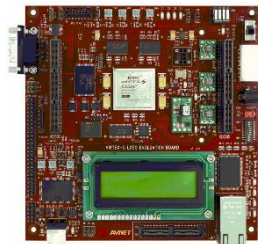


Figura 90: Avnet LX50 Virtex5

- Repetidor WIFI Netgear EX2700: Gracias a su puerto ethernet se encarga de permitir a la placa de desarrollo estar dentro de la red local y conectada al router principal, de forma que sea accesible desde cualquier equipo desde dentro y fuera de la red local.



Figura 91: Repetidor WIFI Netgear ex2700

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA
Daniel Vaquerizo Cid

Tras ver los diferentes elementos, a continuación, se muestra un pequeño esquema donde se puede ver la forma como quedan interconectados los diferentes elementos del conjunto.

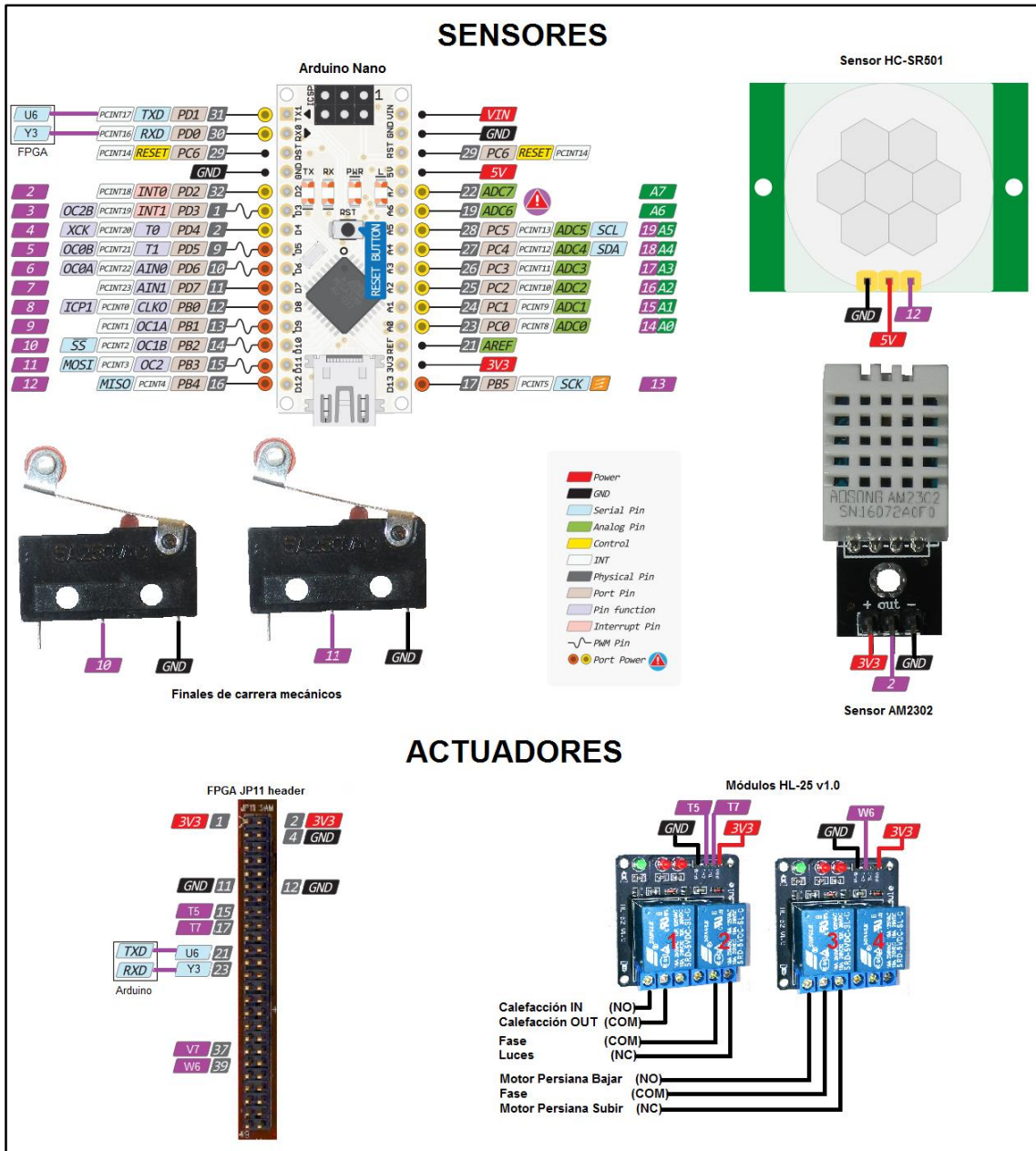


Figura 92: Esquema de sensores y actuadores

Además cabe destacar que la placa de desarrollo queda conectada físicamente mediante un cable ethernet al repetidor wifi anteriormente mencionado.

5.2.2 Funcionamiento del sistema

Tras ver la configuración general del hardware, el siguiente paso es conocer cómo se ha programado tanto el Arduino como la aplicación Linux para que funcione la web.

Por un lado, se encuentra la aplicación de Arduino, su cometido es recibir los datos de los diferentes sensores y enviarlos vía puerto serie a la FPGA, sin embargo, no se hace de forma constante, se recopilan y envían datos a petición de la FPGA, esto es, según el carácter recibido a través del puerto serie, se consultan y envían unos datos u otros. A continuación, se muestra un pequeño fragmento del script donde se puede ver la tarea de lectura y envío de la temperatura real, según si se solicita o no desde la FPGA.

```
if (Serial.available())
{
  value = Serial.read();
  [...]
  else if (value == 'a') //TEMPERATURA CON 1 DECIMAL
  {
    digitalWrite(LED_PIN, HIGH);
    //Lectura de temperatura y envío del valor caracter a caracter por la UART
    ATemp = dht.readTemperature();
    two_integers=LOW;
    entero = ATemp * 10;
    ent1 = (int)(entero / 100); //Primer número por la izquierda
    ent2 = (int)(entero - ent1 * 100) / 10; //Segundo número por la izquierda
    dec1 = (int)(entero - ent1 * 100 - ent2 * 10); //Primer decimal
    sprintf(buf_ent1, "%d", ent1);
    sprintf(buf_ent2, "%d", ent2);
    sprintf(buf_dec1, "%d", dec1);
    if (buf_ent1[0]!='0'){two_integers=HIGH;}
    if (two_integers == HIGH)
    {
      Serial.print(' ');
      delay(50);
      Serial.print(buf_ent1);
    }
    delay(50);
    Serial.print(buf_ent2);
    delay(50);
    Serial.print('.');
    delay(50);
    Serial.print(buf_dec1);
    delay(50);
  }
}
```

Figura 93: Aplicación en Arduino – Fragmento de envío de la temperatura actual

En el fragmento anterior se puede ver que, si el carácter recibido por la UART es una “a”, se consulta el valor de temperatura, se convierte en diferentes caracteres y a continuación, se envía en función de si es un valor mayor o menor a 10.

Por otra parte, se encuentra la aplicación Linux, que se encarga de crear el entorno gráfico de la web, desde el que se puede acceder desde cualquier punto de la red local, para ejecutar el servidor web se utiliza una aplicación del paquete BusyBox, cuyo nombre es “httpd”, es un servidor simple pero más que suficiente para la tarea que se pretende desarrollar.

Para ejecutar el servidor web, se utiliza un script Shell que se encarga de consultar sensores y mostrar los datos empleando la función “echo”, enviando con esta función comandos http hacia el navegador. Mientras el servidor web esté en ejecución, la aplicación en Linux se ejecutará, realizando las tareas que se han descrito anteriormente en el flujograma.

A continuación, se muestran una serie de fragmentos de la aplicación web, que ayudan a entender el funcionamiento de la misma.

El siguiente fragmento se encarga de indicar al navegador que todos los datos que va a recibir a continuación son en formato html.

Tras indicarle el contenido, se muestran una serie de textos que conforman la cabecera de la web y el nombre de la misma en el navegador. Una vez mostrados estos mensajes, se hace la lectura de la temperatura actual y deseada, mostrando los mismo en orden.

```
#!/bin/sh
echo "Content-type: text/html"
echo ""
echo '<html>'
echo '<head>'
echo '<title>TFM D.V. - FPGA Linux</title>'
echo '</head>'
echo '<body>'
echo '<h1> Welcome to Virtex 5 Microblaze automation system using Linux </h1>'
echo '<h3 style="color:blue;">MUESAEII - EPSEVG</h3>'
echo '<h4>TFM D. Vaquerizo</h4>'
echo '<hr>'
echo '<meta http-equiv="refresh" content="10;url=web17.sh" />' #refresh the web ev
#Set initial values for the "variables"
#-----SHOW ACTUAL VALUES-----
#-----ACTUAL TEMP-----
echo '<i>Actual state of the system:</i>'
echo '<br>'
echo '- Temperature of the room:'
microcom -t 400 /dev/ttyUL1 > ATemp.log &
echo a > /dev/ttyUL1 #pull for actual temp
sleep 0.7
echo "$(cat ATemp.log)"
echo '<sup>o</sup></sup>C'
echo '<br>'
#-----DESIRED TEMP-----
echo '- Desired Temperature for the room:'
echo "$(cat DTemp.log)" #read the desired temp, initially not set
echo '<sup>o</sup></sup>C'
echo '<br>'
[...]
```

Figura 94: Aplicación web – Fragmento de arranque y muestra de valores

Justo después de realizar las tareas de este fragmento, se muestran los datos relevantes a valores de otros sensores, indicado con “[...]”.

Una vez mostrados todos los valores referentes a sensores, se realizan los pasos de comparación para decidir en si se debe o no activar alguna de las salidas, en este caso se muestra la función de comparación para activar la calefacción. Se compara el valor deseado con el valor de temperatura redondeado, del redondeo se encarga arduino, ya que no se pueden comparar datos en coma flotante usando Shell.

```
#-----Comparison to activate the Heating system-----
if [ "$(cat HManual.log)" -eq "0" ]; then #no manual mode activated
microcom -t 300 /dev/ttyUL1 > RoundTemp.log &
echo r > /dev/ttyUL1 #pull for actual temp
sleep 0.3
if [ "$(cat DTemp.log)" -gt "$(cat RoundTemp.log)" ]; then #if the desir
# Heating ON
echo 0 > /sys/class/gpio/gpio242/value #turn on Heating Relay
echo "<b>On</b>" > Heating.log
else
# GPIO OFF
echo 1 > /sys/class/gpio/gpio242/value #turn off Heating Relay
echo "Off" > Heating.log
fi
else #The manual mode is active, turn on heating
# Heating ON
echo 0 > /sys/class/gpio/gpio242/value #turn on Heating Relay
echo "<b>On</b>" > Heating.log
fi
echo '- State of the Heating system: '
echo "$(cat Heating.log)"
echo '<br>'
[...]
```

Figura 95: Aplicación web – Fragmento de comparación

Para controlar la calefacción, se consulta si el modo de funcionamiento es automático o manual, en cuyo caso se realiza la acción correspondiente, si está en manual se enciende, si está en automático, se consulta el valor de temperatura y en función de si la actual es menor o no a la deseada, se enciende o apaga en consecuencia.

Tras realizar el resto de comparaciones para activar funciones, marcado con “[...]”, se pasa al formulario web, que utilizando el método “http get”, se encarga de recibir del navegador los valores introducidos y mostrar en consecuencia los valores en la web, a continuación, se muestra el fragmento de formulario correspondiente a la temperatura deseada y el modo de funcionamiento de la calefacción.

```
#-----ENTER VALUES USING FORM-----
echo "<form method=GET action='\"$SCRIPT\"'>\n
  <table nowrap>
  [...]\n
# -- Heating related values -----
echo "<i> Configure the desired options for the Heating system</i><br>"
echo "<tr><td>Desired Temperature </TD><TD><input type='text' name='desired_temp' size=6 value=$(cat
DTemp.log)><sup>o</sup></TD></tr></table>"
echo 'Configure Heating Mode:'
echo '<br>'
if [ "$(cat HManual.log)" -eq "0" ]; then
  echo "<input type='radio' name='Heating_manual' value='0' checked> Automatic<br>\n
        <input type='radio' name='Heating_manual' value='1'> On<br>"
else
  echo "<input type='radio' name='Heating_manual' value='0'> Automatic<br>\n
        <input type='radio' name='Heating_manual' value='1' checked> On<br>"
fi
echo '<hr>'
[...]
```

Figura 96: Aplicación web – Fragmento de configuración de la calefacción

Tras configurar los parámetros deseados, se envían hacia la FPGA pulsando un botón en la web, sin embargo, el modo en el que envía los datos es como una cadena de caracteres y, por tanto, es necesario procesarlos para saber que valores ha entrado el usuario.

Para procesar los datos, al final del script se encuentran las líneas de código que almacenan cada uno de los parámetros de la web en diferentes ficheros, para ello se utiliza el comando “sed” que busca los diferentes valores y los almacena en variables, posteriormente, se guarda el contenido de las variables en ficheros .log, para ser leídos en el próximo refresco de la web.

```
if [ -z "$QUERY_STRING" ]; then #if no values entered, do nothing...
  exit 0
else #if some values have been entered, saves them in variables...
  echo "<meta http-equiv='refresh' content='0;url=web16.sh' />"
  # Just extract the data entered using "sed" command:
  DTemp=echo "$QUERY_STRING" | sed -n 's/^.*desired_temp=\\([^\&]*\\).*$/\1/p' | sed "s/%20//g"
  Hmanual=echo "$QUERY_STRING" | sed -n 's/^.*Heating_manual=\\([^\&]*\\).*$/\1/p' | sed "s/%20//g"
  Bmanual=echo "$QUERY_STRING" | sed -n 's/^.*Blinds_manual=\\([^\&]*\\).*$/\1/p' | sed "s/%20//g"
  Bdesired=echo "$QUERY_STRING" | sed -n 's/^.*Blinds_desired=\\([^\&]*\\).*$/\1/p' | sed "s/%20//g"
  RLight=echo "$QUERY_STRING" | sed -n 's/^.*Room_light=\\([^\&]*\\).*$/\1/p' | sed "s/%20//g"
  #Save the values in the different .log files to read them in every refresh of the webpage
  echo "$DTemp" > DTemp.log #XX°C
  echo "$Hmanual" > HManual.log #1/0
  echo "$Bmanual" > BManual.log #1/0
  echo "$Bdesired" > BDesired.log #1/0
  echo "$RLight" > RLight.log #1/0
fi
echo '</body>'
echo '</html>'
```

Figura 97: Aplicación web – Fragmento de almacenaje de parámetros del usuario

Tras ver los diferentes fragmentos de los scripts, se pueden consultar los scripts finales en el anexo.

Una vez programado tanto el código del arduino nano como el de la FPGA, será necesario compilar un nuevo kernel que incluya la aplicación para que a la hora de arrancar el sistema operativo no haya que pasar el fichero cada vez.

5.2.3 Compilación del nuevo kernel

Una vez realizados los múltiples scripts necesarios para poder ejecutar el servidor web desde la FPGA usando Linux, es necesario integrarlos dentro del sistema de ficheros base para que cada vez que arranque el kernel se pueda acceder a ellos.

El primer paso es descomprimir el sistema de ficheros proporcionado por Xilinx, para ello se ejecutan los siguientes comandos utilizando el terminal en Ubuntu.

```
mkdir tmp_mnt/  
gunzip -c microblaze_complete.cpio.gz | sh -c 'cd tmp_mnt/ && cpio -i'
```

El primer comando se encarga de crear una nueva carpeta, donde se guardará todo el contenido del fichero comprimido. Ejecutando el segundo comando se descomprime todo el sistema de ficheros para Microblaze Big Endian en la carpeta anteriormente creada.

Tras ejecutar ambos comandos la carpeta contiene todo el sistema de ficheros, por lo que solamente es necesario crear las carpetas deseadas y copiar dentro los scripts y ficheros para la web.

Una vez copiados, se debe comprimir de nuevo la imagen, para ello se ejecuta el siguiente comando, que comprimir el sistema de ficheros y le da el nombre deseado, en este caso "webserver14.cpio.gz".

```
sh -c 'cd tmp_mnt/ && find . | cpio -H newc -o' | gzip -9 > webserver16.cpio.gz
```

Tras obtener el sistema de ficheros definitivo, se debe compilar de nuevo el kernel, de forma que la imagen del sistema operativo cargada sea la que contiene todos los scripts y ficheros necesarios para configurar y ejecutar el servidor web en Linux.

Cargando y ejecutando el sistema operativo, tal y como se mostró en el anterior capítulo, se pueden realizar las pruebas pertinentes para ver si la aplicación funciona tal y como se ha diseñado.

5.3 Test de la aplicación

Una vez todos los elementos están interconectados y cada elemento procesador cuenta con su aplicación, el siguiente paso es verificar que la aplicación diseñada funciona tal y como se ha especificado al principio del diseño.

Para verificar la aplicación, se testean las diferentes posibilidades de configuración y salida de los sensores. Para ello, se configuran los diferentes GPIOs y la UART, y a continuación se pone en ejecución el servidor web. Una vez en ejecución es necesario comprobar que el modo de funcionamiento de la calefacción, persiana y luces es el esperado.

Para verificar si funcionan correctamente las salidas, es importante tener en consideración lo siguiente:

- Calefacción encendida conmuta el primer relé (encendiendo el led de la pcb).
- Luces apagadas activan el segundo relé.
- Persiana subida conmuta el tercer relé.

Por tanto, la configuración inversa para cada uno de los tres casos, desactiva el relé correspondiente, dejando la salida conmutada hacia el contacto normalmente cerrado. Conociendo estos aspectos, el estado inicial del sistema para el test, es con la persiana en alto, calefacción apagada y luces apagadas, por lo que los relés 2 y 3 deberían estar activados, tal y como se muestra en la siguiente figura.

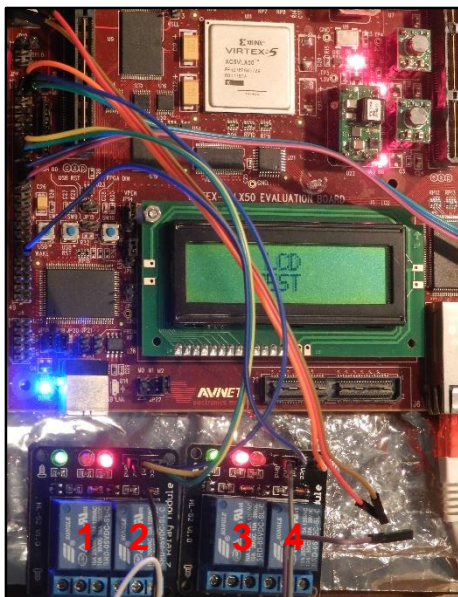


Figura 98: Estado inicial de los relés

5.3.1 Comprobación del control de la calefacción

Para realizar la primera verificación se van variando los parámetros de configuración disponibles para el usuario, de forma que al cambiarlos se pueda ver si el control de la calefacción funciona correctamente.

Cambiando el modo a manual se muestra que la calefacción está encendida (Figura 99).

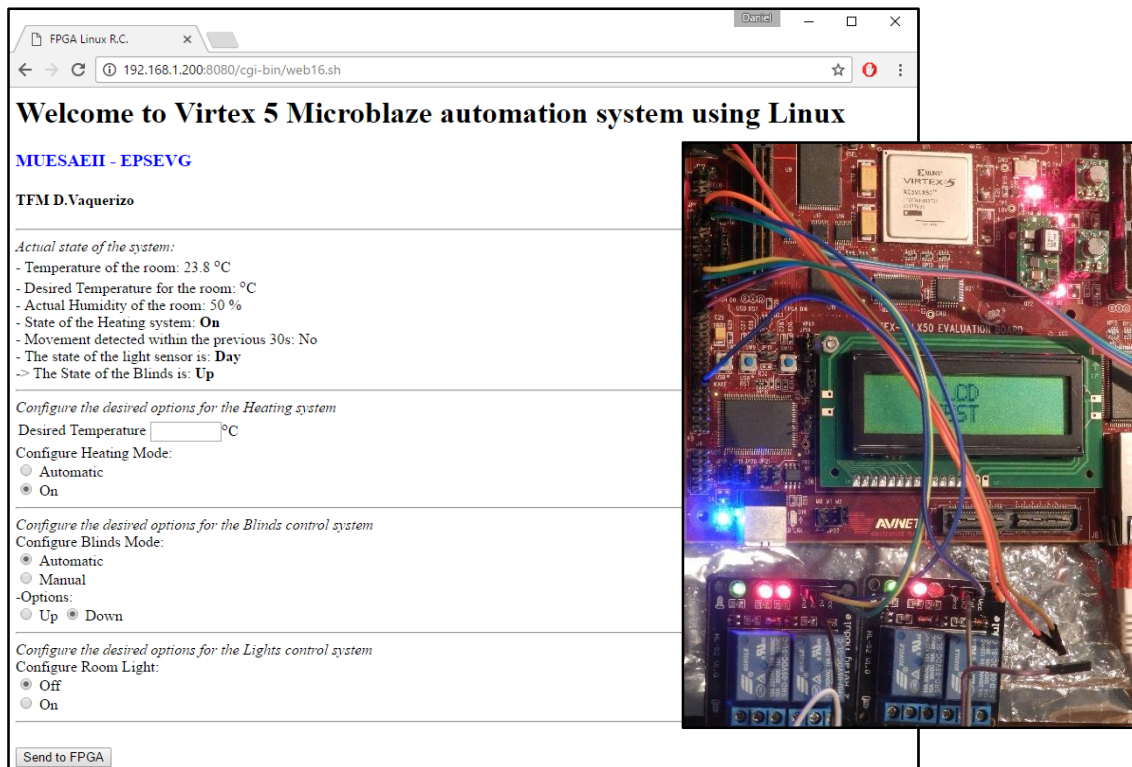


Figure 99 shows a screenshot of a web browser displaying a control interface for a Virtex 5 Microblaze automation system. The browser window title is "FPGA Linux R.C." and the address bar shows "192.168.1.200:8080/cgi-bin/web16.sh". The page content includes a welcome message, the user name "MUESAEII - EPSEVG", and the name "TFM D.Vaquerizo". The "Actual state of the system:" section lists: Temperature of the room: 23.8 °C, Desired Temperature for the room: °C, Actual Humidity of the room: 50 %, State of the Heating system: On, Movement detected within the previous 30s: No, The state of the light sensor is: Day, and The State of the Blinds is: Up. The "Configure the desired options for the Heating system" section includes a "Desired Temperature" input field (empty) and "Configure Heating Mode" with radio buttons for "Automatic" and "On". The "Configure the desired options for the Blinds control system" section includes "Configure Blinds Mode" with radio buttons for "Automatic" and "Manual", and "Options" with radio buttons for "Up" and "Down". The "Configure the desired options for the Lights control system" section includes "Configure Room Light" with radio buttons for "Off" and "On". A "Send to FPGA" button is located at the bottom of the interface.

Figura 99: WEB - Calefacción manual ON

Para comprobar que las salidas funcionan correctamente se debe comprobar si el relé número 2 ha sido conmutado por lo que el modo manual funciona correctamente. A continuación, se prueba el modo automático. Para comprobarlo se fija inicialmente un valor de temperatura inferior a la temperatura ambiente, de forma que se apague la calefacción.

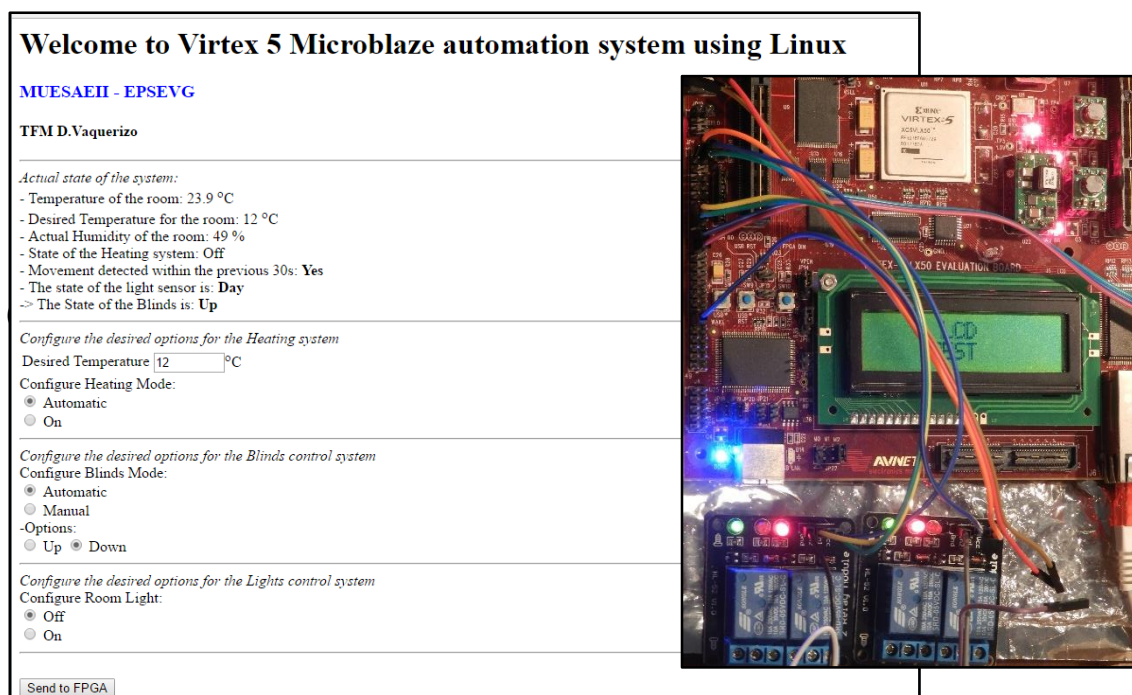


Figure 100 shows a screenshot of the same web browser control interface as in Figure 99. The "Actual state of the system:" section now lists: Temperature of the room: 23.9 °C, Desired Temperature for the room: 12 °C, Actual Humidity of the room: 49 %, State of the Heating system: Off, Movement detected within the previous 30s: Yes, The state of the light sensor is: Day, and The State of the Blinds is: Up. The "Configure the desired options for the Heating system" section now shows "Desired Temperature" set to 12 °C and "Configure Heating Mode" with radio buttons for "Automatic" and "On". The "Configure the desired options for the Blinds control system" section remains the same. The "Configure the desired options for the Lights control system" section remains the same. A "Send to FPGA" button is located at the bottom of the interface.

Figura 100: WEB - Calefacción automática, T ambiente superior a deseada

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA
Daniel Vaquerizo Cid

Con la calefacción apagada se aumenta la temperatura deseada para ver si se vuelve a encender, a continuación, se muestra la web con el nuevo valor.

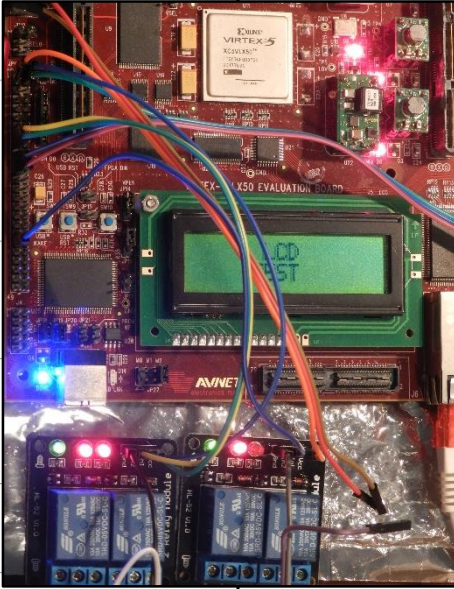
<p>Welcome to Virtex 5 Microblaze automation system using Linux</p> <p>MUESAEII - EPSEVG</p> <p>TFM D.Vaquerizo</p> <hr/> <p><i>Actual state of the system:</i></p> <ul style="list-style-type: none">- Temperature of the room: 23.9 °C- Desired Temperature for the room: 25 °C- Actual Humidity of the room: 50 %- State of the Heating system: On- Movement detected within the previous 30s: No- The state of the light sensor is: Day-> The State of the Blinds is: Up <hr/> <p><i>Configure the desired options for the Heating system</i></p> <p>Desired Temperature <input type="text" value="25"/> °C</p> <p>Configure Heating Mode:</p> <ul style="list-style-type: none"><input checked="" type="radio"/> Automatic<input type="radio"/> On <hr/> <p><i>Configure the desired options for the Blinds control system</i></p> <p>Configure Blinds Mode:</p> <ul style="list-style-type: none"><input checked="" type="radio"/> Automatic<input type="radio"/> Manual <p>-Options:</p> <ul style="list-style-type: none"><input type="radio"/> Up <input checked="" type="radio"/> Down <hr/> <p><i>Configure the desired options for the Lights control system</i></p> <p>Configure Room Light:</p> <ul style="list-style-type: none"><input checked="" type="radio"/> Off<input type="radio"/> On <hr/> <p><input type="button" value="Send to FPGA"/></p>	
--	--

Figura 101: WEB - Calefacción automática, T ambiente inferior a deseada

Aumentando la temperatura ambiente, por ejemplo, soplando al sensor, las temperaturas deberían equipararse y apagarse por tanto la calefacción. En la siguiente figura se muestra este caso.

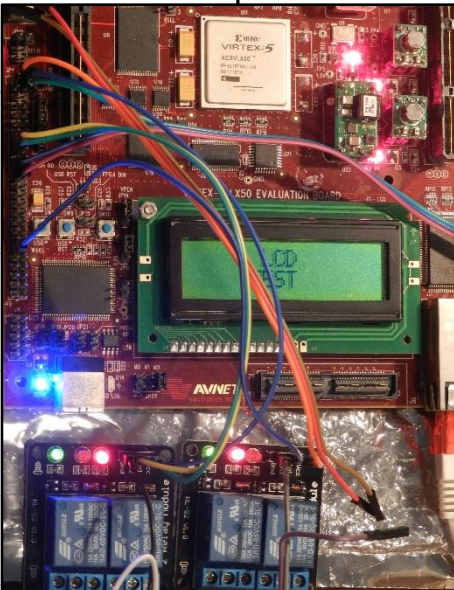
<p>Welcome to Virtex 5 Microblaze automation system using Linux</p> <p>MUESAEII - EPSEVG</p> <p>TFM D.Vaquerizo</p> <hr/> <p><i>Actual state of the system:</i></p> <ul style="list-style-type: none">- Temperature of the room: 28.1 °C- Desired Temperature for the room: 25 °C- Actual Humidity of the room: 99 %- State of the Heating system: Off- Movement detected within the previous 30s: No- The state of the light sensor is: Day-> The State of the Blinds is: Up <hr/> <p><i>Configure the desired options for the Heating system</i></p> <p>Desired Temperature <input type="text" value="25"/> °C</p> <p>Configure Heating Mode:</p> <ul style="list-style-type: none"><input checked="" type="radio"/> Automatic<input type="radio"/> On <hr/> <p><i>Configure the desired options for the Blinds control system</i></p> <p>Configure Blinds Mode:</p> <ul style="list-style-type: none"><input checked="" type="radio"/> Automatic<input type="radio"/> Manual <p>-Options:</p> <ul style="list-style-type: none"><input type="radio"/> Up <input checked="" type="radio"/> Down <hr/> <p><i>Configure the desired options for the Lights control system</i></p> <p>Configure Room Light:</p> <ul style="list-style-type: none"><input checked="" type="radio"/> Off<input type="radio"/> On <hr/> <p><input type="button" value="Send to FPGA"/></p>	
---	--

Figura 102: WEB - Calefacción automática, T ambiente superior a deseada

(II)

Tras comprobar el correcto funcionamiento de los diferentes modos de la calefacción, el siguiente paso es probar el funcionamiento de las persianas automáticas, que dependen de varios sensores.

5.3.2 Comprobación del control de las persianas

El funcionamiento esperado de las persianas es, en modo automático con luz de día, siempre se sube la persiana si es que está bajada, cuando se hace de noche, se baja la persiana si es que no lo estaba ya. Finalmente, en modo manual se sube o baja la persiana sin importar el estado del sensor lumínico.

Para determinar los valores de la persiana se emplean dos pulsadores, si no están pulsados ambos, la persiana estará abierta y si el sensor de luz recibe suficiente luz, se considera que es de día, éste es el caso inicial del test, que se muestra a continuación.

Welcome to Virtex 5 Microblaze automation system using Linux	
MUESAEII - EPSEVG	
TFM D.Vaquerizo	
<i>Actual state of the system:</i> - Temperature of the room: 23.4 °C - Desired Temperature for the room: °C - Actual Humidity of the room: 51 % - State of the Heating system: Off - Movement detected within the previous 30s: No - The state of the light sensor is: Day -> The State of the Blinds is: Up	
<i>Configure the desired options for the Heating system</i> Desired Temperature <input type="text"/> °C Configure Heating Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> On	
<i>Configure the desired options for the Blinds control system</i> Configure Blinds Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> Manual -Options: <input type="radio"/> Up <input checked="" type="radio"/> Down	
<i>Configure the desired options for the Lights control system</i> Configure Room Light: <input type="radio"/> Off <input checked="" type="radio"/> On	

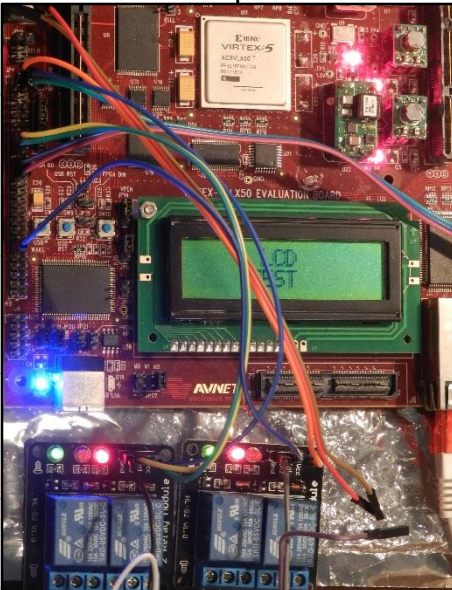


Figura 103: WEB - Persianas en automático arriba, de día

A continuación, se muestra el caso en el que el sensor detecta que se hace de noche y la persiana está subida, por tanto, se desactiva la salida del relé 3 para bajar la persiana, mientras se muestra un aviso al usuario.

Welcome to Virtex 5 Microblaze automation system using Linux	
MUESAEII - EPSEVG	
TFM D.Vaquerizo	
<i>Actual state of the system:</i> - Temperature of the room: 23.4 °C - Desired Temperature for the room: °C - Actual Humidity of the room: 51 % - State of the Heating system: Off - Movement detected within the previous 30s: Yes - The state of the light sensor is: Night - The State of the Blinds is: Up - Is dark outside! - The Blinds are being closed...	



Figura 104: WEB - Persianas en automático arriba, de noche

A continuación, se muestra el caso en que siendo de noche se detecta que la persiana está bajada (ambos pulsadores activos), en cuyo caso, se muestra el estado de los sensores, mientras se mantiene la salida desactivada para mantener la persiana abajo.

Welcome to Virtex 5 Microblaze automation system using Linux	
MUESAEII - EPSEVG	
TFM D.Vaquerizo	
<i>Actual state of the system:</i> - Temperature of the room: 23.5 °C - Desired Temperature for the room: °C - Actual Humidity of the room: 50 % - State of the Heating system: Off - Movement detected within the previous 30s: Yes - The state of the light sensor is: Night - The State of the Blinds is: Down	
<i>Configure the desired options for the Heating system</i> Desired Temperature <input type="text"/> °C Configure Heating Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> On	
<i>Configure the desired options for the Blinds control system</i> Configure Blinds Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> Manual -Options: <input type="radio"/> Up <input checked="" type="radio"/> Down	
<i>Configure the desired options for the Lights control system</i> Configure Room Light: <input type="radio"/> Off <input checked="" type="radio"/> On	

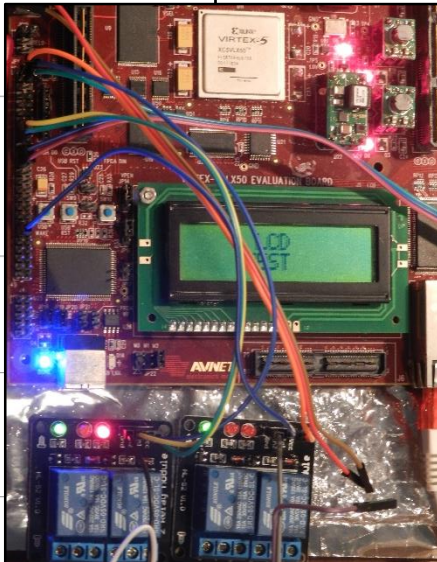


Figura 105: WEB – Persianas en automático abajo, de noche

A continuación, el paso lógico es comprobar que cuando se hace de día con la persiana bajada, ésta sube, mostrando un mensaje al usuario.

Welcome to Virtex 5 Microblaze automation system using Linux	
MUESAEII - EPSEVG	
TFM D.Vaquerizo	
<i>Actual state of the system:</i> - Temperature of the room: 23.6 °C - Desired Temperature for the room: °C - Actual Humidity of the room: 50 % - State of the Heating system: Off - Movement detected within the previous 30s: No - The state of the light sensor is: Day -> The State of the Blinds is: Down - There is plenty of light outside! - The Blinds are being opened...	
<i>Configure the desired options for the Heating system</i> Desired Temperature <input type="text"/> °C Configure Heating Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> On	
<i>Configure the desired options for the Blinds control system</i> Configure Blinds Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> Manual -Options: <input type="radio"/> Up <input checked="" type="radio"/> Down	
<i>Configure the desired options for the Lights control system</i> Configure Room Light: <input type="radio"/> Off <input checked="" type="radio"/> On	

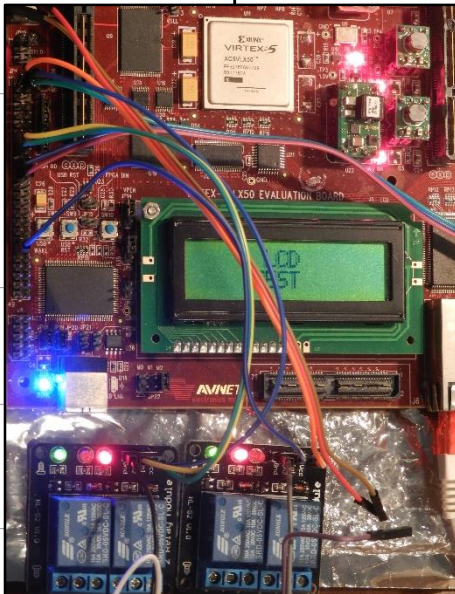


Figura 106: WEB – Persianas en automático abajo, de día

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

Una vez comprobadas las diferentes posibilidades del modo automático, se pasa a comprobar el modo manual, donde se comprueba que, al seleccionar la posición deseada, la salida cambia en consecuencia.

La primera posibilidad es seleccionar la posición deseada abajo mientras la persiana se encuentra subida, en cuyo caso se muestra un mensaje de aviso y se desactiva la salida.

Welcome to Virtex 5 Microblaze automation system using Linux	
MUESAEII - EPSEVG	
TFM D.Vaquerizo	
<i>Actual state of the system:</i> - Temperature of the room: 23.4 °C - Desired Temperature for the room: °C - Actual Humidity of the room: 51 % - State of the Heating system: Off - Movement detected within the previous 30s: No - The State of the Blinds is: Up - The Blinds are being closed...	
<i>Configure the desired options for the Heating system</i> Desired Temperature <input type="text"/> °C Configure Heating Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> On	
<i>Configure the desired options for the Blinds control system</i> Configure Blinds Mode: <input type="radio"/> Automatic <input checked="" type="radio"/> Manual -Options: <input type="radio"/> Up <input checked="" type="radio"/> Down	
<i>Configure the desired options for the Lights control system</i> Configure Room Light: <input type="radio"/> Off <input checked="" type="radio"/> On	

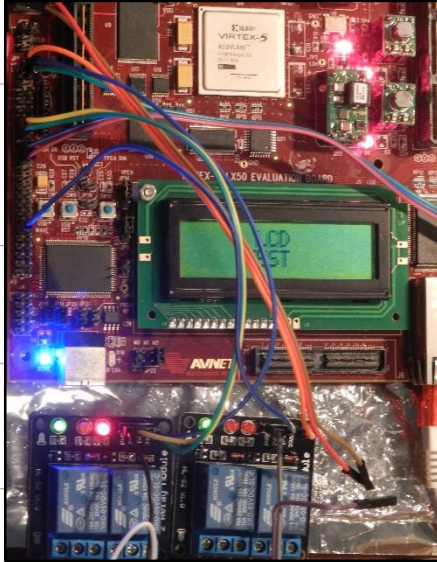


Figura 107: WEB - Persianas modo manual abajo, persiana subida

Una vez la persiana se encuentra abajo, se muestra el mensaje de posición de la persiana, dejando de mostrar el aviso de persiana bajando.

Welcome to Virtex 5 Microblaze automation system using Linux	
MUESAEII - EPSEVG	
TFM D.Vaquerizo	
<i>Actual state of the system:</i> - Temperature of the room: 23.1 °C - Desired Temperature for the room: °C - Actual Humidity of the room: 49 % - State of the Heating system: Off - Movement detected within the previous 30s: Yes - The State of the Blinds is: Down	
<i>Configure the desired options for the Heating system</i> Desired Temperature <input type="text"/> °C Configure Heating Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> On	
<i>Configure the desired options for the Blinds control system</i> Configure Blinds Mode: <input type="radio"/> Automatic <input checked="" type="radio"/> Manual -Options: <input type="radio"/> Up <input checked="" type="radio"/> Down	
<i>Configure the desired options for the Lights control system</i> Configure Room Light: <input type="radio"/> Off <input checked="" type="radio"/> On	

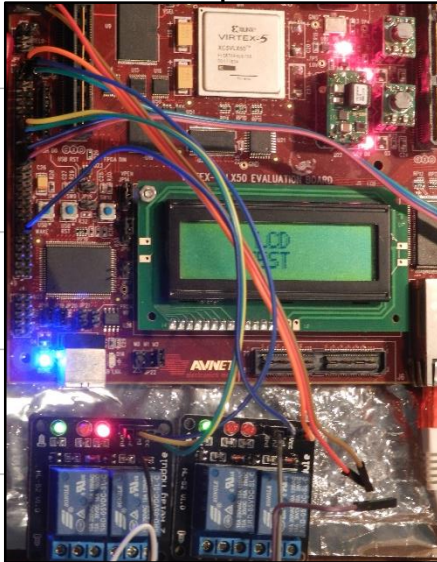


Figura 108: WEB - Persianas modo manual abajo, persiana bajada

Tras probar el modo manual para persiana bajada, se prueba el modo manual para persiana subida, que, partiendo del estado anterior, debería mostrar un aviso al usuario

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA

Daniel Vaquerizo Cid

y activar la salida para comenzar a subir la persiana.

Welcome to Virtex 5 Microblaze automation system using Linux

MUESAEII - EPSEVVG

TFM D.Vaquerizo

Actual state of the system:

- Temperature of the room: 23.3 °C
- Desired Temperature for the room: °C
- Actual Humidity of the room: 50 %
- State of the Heating system: Off
- Movement detected within the previous 30s: No
- The State of the Blinds is: Down - The Blinds are being opened...

Configure the desired options for the Heating system

Desired Temperature °C

Configure Heating Mode:

- Automatic
- On

Configure the desired options for the Blinds control system

Configure Blinds Mode:

- Automatic
- Manual

-Options:

- Up Down

Configure the desired options for the Lights control system

Configure Room Light:

- Off
- On

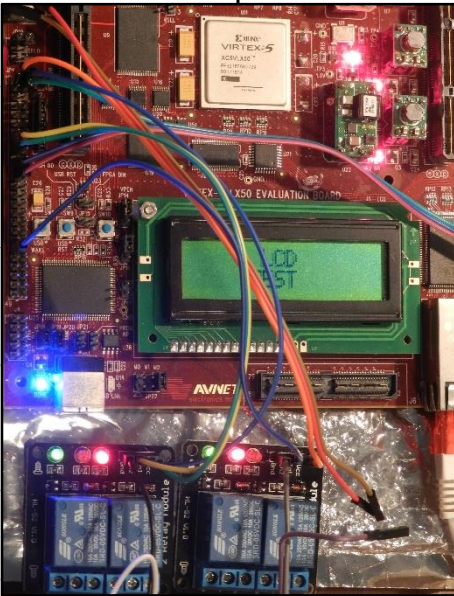


Figura 109: WEB - Persianas modo manual arriba, persiana bajada

Tras enrollar la persiana hacia arriba, dado que ambos sensores detectan que se encuentra completamente subida, se muestra el mensaje de persiana subida.

← → ↻ 192.168.1.200:8080/cgi-bin/web16.sh ☆ 🔴 ⋮

Welcome to Virtex 5 Microblaze automation system using Linux

MUESAEII - EPSEVVG

TFM D.Vaquerizo

Actual state of the system:

- Temperature of the room: 23.3 °C
- Desired Temperature for the room: °C
- Actual Humidity of the room: 49 %
- State of the Heating system: Off
- Movement detected within the previous 30s: Yes
- The State of the Blinds is: Up

Configure the desired options for the Heating system

Desired Temperature °C

Configure Heating Mode:

- Automatic
- On

Configure the desired options for the Blinds control system

Configure Blinds Mode:

- Automatic
- Manual

-Options:

- Up Down

Configure the desired options for the Lights control system

Configure Room Light:

- Off
- On

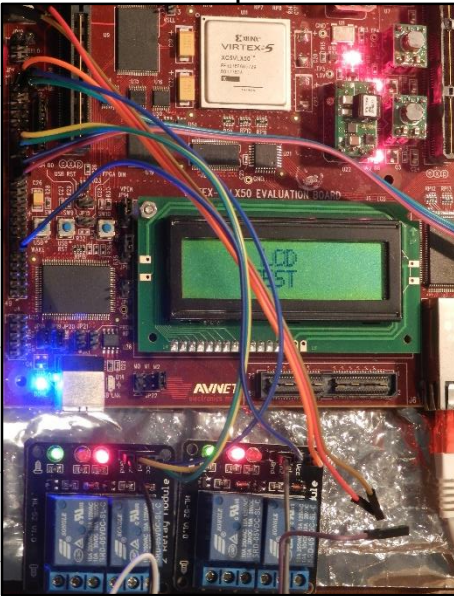


Figura 110: WEB - Persianas modo manual arriba, persiana subida

5.3.3 Comprobación del control las luces

Tras verificar que el control sobre las persianas funciona correctamente, queda probar el funcionamiento de las luces de la habitación y el del sensor de movimiento.

Para ello, partiendo del estado inicial, se activa la luz, por tanto, el relé número dos debería desactivarse.

Welcome to Virtex 5 Microblaze automation system using Linux	
MUESAEII - EPSEVG	
TFM D.Vaquerizo	
<i>Actual state of the system:</i> - Temperature of the room: 23.8 °C - Desired Temperature for the room: °C - Actual Humidity of the room: 49 % - State of the Heating system: Off - Movement detected within the previous 30s: No - The state of the light sensor is: Day -> The State of the Blinds is: Up	
<i>Configure the desired options for the Heating system</i> Desired Temperature: <input type="text"/> °C Configure Heating Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> On	
<i>Configure the desired options for the Blinds control system</i> Configure Blinds Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> Manual -Options: <input type="radio"/> Up <input checked="" type="radio"/> Down	
<i>Configure the desired options for the Lights control system</i> Configure Room Light: <input type="radio"/> Off <input checked="" type="radio"/> On	

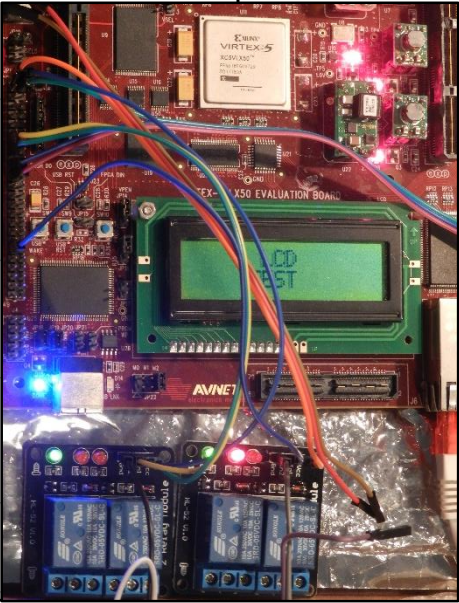


Figura 111: WEB – Luces activadas

Apagando las luces, se vuelve al estado inicial, donde se mantiene activo tanto el relé 2 como el 3, ya que la calefacción no se encuentra activa.

Welcome to Virtex 5 Microblaze automation system using Linux	
MUESAEII - EPSEVG	
TFM D.Vaquerizo	
<i>Actual state of the system:</i> - Temperature of the room: 23.8 °C - Desired Temperature for the room: °C - Actual Humidity of the room: 49 % - State of the Heating system: Off - Movement detected within the previous 30s: Yes - The state of the light sensor is: Day -> The State of the Blinds is: Up	
<i>Configure the desired options for the Heating system</i> Desired Temperature: <input type="text"/> °C Configure Heating Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> On	
<i>Configure the desired options for the Blinds control system</i> Configure Blinds Mode: <input checked="" type="radio"/> Automatic <input type="radio"/> Manual -Options: <input type="radio"/> Up <input checked="" type="radio"/> Down	
<i>Configure the desired options for the Lights control system</i> Configure Room Light: <input checked="" type="radio"/> Off <input type="radio"/> On	

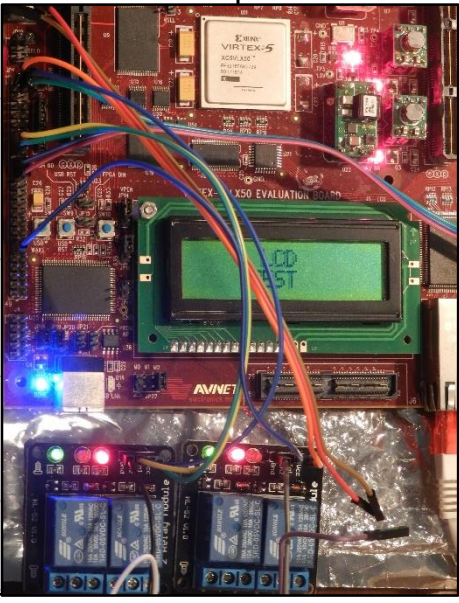


Figura 112: WEB – Luces desactivadas

5.3.4 Comprobación del sensor de movimiento

Tras comprobar las diferentes acciones de control solamente queda comprobar que el sensor de movimiento se muestra correctamente en la web, se puede ver en las dos figuras previas, que con el simple movimiento realizado mientras se realizaban las diferentes comprobaciones, ha detectado movimiento y por tanto ha pasado de inactivo (Figura 111) a activo (Figura 112).

Con todos los elementos comprobados se puede concluir el test sobre la web y el hardware y dado que tanto la aplicación web como los diferentes elementos empleados, desde sensores, hasta sistema operativo han funcionado correctamente, la aplicación se da por concluida y se puede pasar a ver las conclusiones del trabajo.

CAPÍTULO 6

CONCLUSIONES

En este capítulo final se presenta una visión global de los resultados del trabajo, evaluando para ello los diferentes objetivos marcados para ver si se han cumplido parcial o completamente. Tras el análisis de objetivos se plantean las conclusiones y por último las líneas de mejora posibles sobre el trabajo desarrollado.

6.1 Análisis de objetivos

El primero de los objetivos era la creación del softcore Microblaze cumpliendo una serie de requisitos, para posteriormente ser implementado en la placa de desarrollo. Durante el desarrollo del capítulo 2 se ha podido ver cómo se diseña el procesador, se añaden los periféricos y se configura para poderlo ajustar a las especificaciones del diseño inicialmente marcadas, una vez concluido el diseño se ha podido realizar la implementación del mismo sobre la FPGA Virtex 5 con lo que el primer objetivo se ha cumplido completamente.

El segundo de los objetivos consistía en diseñar un kernel Linux completamente funciona, basando para ello el diseño en el sistema microprocesador Microblaze y sus periféricos. Durante el capítulo 3 se ha podido obtener tanto la descripción del hardware, para poder ajustar el sistema operativo al diseño como el sistema de ficheros necesario para su funcionamiento. Con los elementos esenciales se ha compilado satisfactoriamente el kernel personalizado, que tras diferentes pruebas ha demostrado ser completamente funcional y poder manipular los diferentes elementos incluidos durante el proceso de diseño del hardware microprocesador. Por tanto, el objetivo de diseño del kernel se ha cumplido completamente también.

Finalmente, el tercero de los objetivos principales consistía en diseñar una aplicación que demostrase el correcto funcionamiento del conjunto FPGA-Microblaze-Linux. La parte de diseño de la aplicación ha consistido finalmente en añadir algunos sensores y hardware externo al sistema, sin embargo, tras diseñar el script que se ejecuta bajo Linux, se han realizado varias pruebas de funcionalidad donde se demostraba la conexión al servidor web, la interacción con la web y el control de los periféricos para realizar tareas de control. Tras comprobar el funcionamiento del conjunto se puede concluir que el punto de desarrollo de la aplicación ha sido cumplido.

6.2 Conclusiones

Con el diseño totalmente funcional, se puede decir que se han cumplido los diferentes objetivos que se habían fijado para este trabajo, dado que, la creación del sistema basado en Microblaze es completamente funcional, y capaz de ejecutar Linux sin problema. Y, la parte de desarrollo software donde se ha obtenido el kernel personalizado para Microblaze y la propia aplicación web, ha permitido que éste se ejecute sin problemas sobre Microblaze y que se pudiesen controlar cada uno de los periféricos sin problemas desde la aplicación final.

Analizando el proceso de diseño es posible ver la complejidad que tiene el diseño de un sistema embebido empleando una FPGA, ya que durante el desarrollo se han empleado multitud de herramientas, para poder realizar las diferentes fases, pasando por el diseño del hardware, diseño del software y el desarrollo de la aplicación, convirtiendo así el conjunto de FPGA y Microblaze en un sistema totalmente funcional.

Además, el hecho de emplear diferentes entornos complica enormemente el proceso de diseño, dado que, para poder llegar a la propia aplicación, ha sido necesario trabajar con herramientas de Xilinx, herramientas del kernel Linux y herramientas de Windows, donde cada una de las aplicaciones deben estar correctamente configuradas para poder entender los ficheros que resultan de cada uno de los pasos previos.

Por otro lado, vale la pena destacar que tras tener el sistema funcionando, se puede modificar siempre que se desee el funcionamiento de la aplicación o los periféricos asociados a Microblaze para añadir nuevas funcionalidades o protocolos de comunicación como podría ser el CAN-bus, incluir un segundo microprocesador, etc, lo que otorga al sistema de una flexibilidad mucho mayor que cuando se emplea un microprocesador como podría ser Arduino, ya que en ese caso, el desarrollo está limitado al diseño de la placa en el momento de adquirirla.

Para concluir, emplear un sistema basado en Microblaze aporta una última ventaja y es, que se puede migrar a cualquier otra FPGA más moderna o potente, lo que permitiría mantener el diseño hardware y el kernel, aunque se pase a otra placa de desarrollo completamente diferente. Y, por otra parte, la ventaja principal sobre los microprocesadores es que, aunque en este caso se ha empleado una distribución de Linux normal, se puede realizar un cambio de sistema operativo para emplear uno en tiempo real, donde las tareas de monitorización y control se realicen bajo un estricto esquema temporal.

6.3 Líneas de mejora

Una vez concluidos todos los pasos de desarrollo durante el trabajo, viendo los resultados obtenidos, se plantean una serie de líneas de mejora que pretenden mejorar el conjunto, así como la funcionalidad del mismo.

Un primer punto de mejora podría ser la implementación del diseño en un sistema con Microblaze multi-core para ver si se mejora notablemente la velocidad del sistema, así como las posibilidades de computo del mismo, además sería interesante poder realizar el diseño sobre una FPGA con bus AXI, dado que la placa empleada solamente permitía PLB.

El segundo punto de mejora podría ser la inclusión de una interfaz inalámbrica entre el sistema de sensores y la FPGA, para ello se puede emplear un accesorio que transforme la uart a bluetooth o cualquier otro protocolo, lo que permitiría incluir un sistema de sensores en cada una de las habitaciones del hogar haciendo mejorando la monitorización y control sobre el hogar.

La aplicación ha funcionado correctamente, sin embargo, dados los limitados

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA
Daniel Vaquerizo Cid

conocimientos de programación html, se puede mejorar el aspecto de la misma, incluir una serie de gráficas donde se pueda ver el histórico de datos obtenidos, varias pestañas o incluso ver la temperatura de diferentes habitaciones si se desea, buscar la forma de poder acceder a la aplicación desde el exterior de la red local, etc.

Además, un punto que no se ha tratado y que podría resultar de utilidad es el control de accesos, pudiendo colocar un lector de tarjetas RFID, mostrando en la web la última persona en entrar o salir.

BIBLIOGRAFIA

- [1] Xilinx® Virtex™-5 LX50 Evaluation Kit User Manual, 2006
- [2] Xilinx GitHub repository, 2017
<https://github.com/xilinx>
- [3] Xilinx Wiki - Linux, 2016
<http://www.wiki.xilinx.com/>
- [4] The Processor Local Bus (v4.6) Xilinx, 2013
https://www.xilinx.com/products/intellectual-property/plb_v46.html
- [5] ELI BILLAUER, Linux on Microblaze, 2011
<http://billauer.co.il/blog/2011/08/linux-microblaze/>
- [6] JEFF JOHNSON, List and comparison of FPGA companies, 2011
<http://www.fpgadeveloper.com/2011/07/list-and-comparison-of-fpga-companies.html>
- [7] SVEN ANDERSSON, FPGA design from scratch, 2011
<http://svenand.blogdrive.com/archive/170.html#.WIOUIBvhCUk>
- [8] Advanced Bash – Scripting Guide, 16.1 Basic Commands, Revision 10, 2014
<http://tldp.org/LDP/abs/html/>
- [9] Xilinx Forums – SDK and device-tree ISE 14.7, 2014
<https://forums.xilinx.com/t5/Embedded-Linux/SDK-and-devicetree-from-ISE14-7/td-p/488974>
- [10] ROBIN COXE, Porting Linux to a Xilinx Microblaze Processor on the Virtex-6, 2011
<https://es.scribd.com/doc/63732567/Microblaze-Linux-on-Xilinx-ML605>
- [11] Registered Guest Resources, Xilinx, 2016
https://www.xilinx.com/guest_resources/gnu/
- [12] A Tutorial on the Device Tree (Zynq), Xillybus, 2016
<http://xillybus.com/tutorials/device-tree-zynq-1>
- [13] JAVIER GONZALEZ GONZALEZ, Setting up the environment for the Xilinx ZC702, 2013
<https://javigon.com/2013/03/27/setting-up-the-environment-for-the-xilinx-zc702-zynq-7000-soc-ise-design-suite-14-x/>
- [14] Build and modify a rootfs filesystem, Xilinx, 2013
<http://www.wiki.xilinx.com/Build+and+Modify+a+Rootfs>

AGRADECIMIENTOS

En primer lugar, quiero dar las gracias al director del proyecto, Mariano López, primero de todo por darme la idea inicial para desarrollar el trabajo, por los consejos durante la realización del mismo y sobre todo por el apoyo y la cantidad enorme de material que me ha prestado desinteresadamente.

A todos los compañeros del máster, que siempre han aportado esa pequeña chispa que hace falta para mantener el humor durante los momentos de mayor estrés o incluso aburrimiento, por los buenos momentos dentro y fuera de clase y sobre todo a mi buen amigo Adrián, que me ha apoyado, ayudado siempre que ha podido y lo más importante, aguantado durante los años de carrera, prácticas en empresa y máster.

También quiero agradecer el apoyo y la paciencia a Marko, Pau y Alejandro, porque sé que soy un hueso duro de roer y a veces cuesta aguantarme, quiero darles las gracias por haberme aportado muchísimo, he aprendido mucho de ellos y siempre han estado dispuestos a ofrecer su ayuda para solventar todo tipo de problemas.

También quiero dar gracias a la familia y amigos, que en mayor o menor medida siempre han estado ahí, apoyándome para poder tirar adelante durante todos estos años de estudios, en especial a mis padres y mi hermana, a los que les debo todo. Y, por último, a Ángel, Silvia, Sonia y Núria, que han sido una fuente inagotable de ideas, apoyo y motivación.

ANEXO

A continuación, se muestran los códigos tanto de Linux como de Arduino.

- Configuración de los periféricos (FPGA)

```
#!/bin/sh
echo "Configuring the peripherals..."
mknod /dev/ttyUL1 c 204 188
echo "UART Configured"
ifconfig eth0 192.168.1.200 up
echo "Ethernet Configured"
echo 252 > /sys/class/gpio/export
echo 253 > /sys/class/gpio/export
echo 254 > /sys/class/gpio/export
echo 255 > /sys/class/gpio/export
echo "Registered switch buttons gpios"
echo in > /sys/class/gpio/gpio252/direction
echo in > /sys/class/gpio/gpio253/direction
echo in > /sys/class/gpio/gpio254/direction
echo in > /sys/class/gpio/gpio255/direction
echo "Configured switch buttons as inputs"

echo 245 > /sys/class/gpio/export
echo 246 > /sys/class/gpio/export
echo 247 > /sys/class/gpio/export
echo "Registered push buttons gpios"
echo in > /sys/class/gpio/gpio245/direction
echo in > /sys/class/gpio/gpio246/direction
echo in > /sys/class/gpio/gpio247/direction
echo "Configured push buttons as inputs"

echo 248 > /sys/class/gpio/export
echo 249 > /sys/class/gpio/export
echo 250 > /sys/class/gpio/export
echo 251 > /sys/class/gpio/export
echo "Registered LEDs gpios"
echo out > /sys/class/gpio/gpio248/direction
echo out > /sys/class/gpio/gpio249/direction
echo out > /sys/class/gpio/gpio250/direction
echo out > /sys/class/gpio/gpio251/direction
echo "Configured LEDs as outputs"
echo 1 > /sys/class/gpio/gpio251/value
echo 1 > /sys/class/gpio/gpio248/value

echo 241 > /sys/class/gpio/export
echo 242 > /sys/class/gpio/export
echo 243 > /sys/class/gpio/export
echo 244 > /sys/class/gpio/export
echo "Registered JP11 gpios"
echo out > /sys/class/gpio/gpio241/direction
echo out > /sys/class/gpio/gpio242/direction
echo out > /sys/class/gpio/gpio243/direction
echo out > /sys/class/gpio/gpio244/direction
echo "Configured JP11 gpios as outputs"
echo "All peripherals configured"
```

- Aplicación de monitorización WEB (FPGA)

```
#!/bin/sh
echo "Content-type: text/html"
echo ""
echo '<html>'
echo '<head>'
echo '<title>TFM D.V. - FPGA Linux</title>'
echo '</head>'
echo '<body>'
echo '<h1> Welcome to Virtex 5 Microblaze automation system using Linux </h1>'
#echo ''
echo '<h3 style="color:blue;">MUESAEII - EPSEVG</h3>'
echo '<h4>TFM D.Vaquerizo</h4>'
echo '<hr>'
echo '<meta http-equiv="refresh" content="10;url=web16.sh" />' #refresh the web ev
#Set initial values for the "variables"
#-----SHOW ACTUAL VALUES-----
#-----ACTUAL TEMP-----
echo '<i>Actual state of the system:</i>'
echo '<br>'
echo '- Temperature of the room:'
microcom -t 400 /dev/ttyUL1 > ATemp.log &
echo a > /dev/ttyUL1 #pull for actual temp
sleep 0.7
echo "$(cat ATemp.log)"
echo ' <sup>o</sup>C'
echo '<br>'
#-----DESIRED TEMP-----
echo '- Desired Temperature for the room:'
echo "$(cat DTemp.log)" #read the desired temp, initially not set
echo ' <sup>o</sup>C'
echo '<br>'
#-----ACTUAL HUMIDITY-----

microcom -t 400 /dev/ttyUL1 > HRoom.log &
echo h > /dev/ttyUL1 #pull for actual temp
echo '- Actual Humidity of the room: '
sleep 1
echo "$(cat HRoom.log)"
echo ' %'
echo '<br>'
#-----Comparison to activate the Heating system-----
if [ "$(cat HManual.log)" -eq "0" ]; then #no manual mode activated
    #if [ "$(cat DTemp.log)" -gt "$(cut -c1-2 ATemp.log)" ]; then #If the de
    microcom -t 300 /dev/ttyUL1 > RoundTemp.log &
    echo r > /dev/ttyUL1 #pull for actual temp
    sleep 0.3
    sleep 0.3
    if [ "$(cat DTemp.log)" -gt "$(cat RoundTemp.log)" ]; then #If the desir
        # Heating ON
        echo 0 > /sys/class/gpio/gpio242/value #turn on Heating Relay
        echo "<b>On</b>" > Heating.log
    else
        # GPIO OFF
        echo 1 > /sys/class/gpio/gpio242/value #turn off Heating Relay
        echo "Off" > Heating.log
    fi
fi
else #The manual mode is active, turn on heating
    # Heating ON
    echo 0 > /sys/class/gpio/gpio242/value #turn on Heating Relay
    echo "<b>On</b>" > Heating.log
fi
echo '- State of the Heating system: '
echo "$(cat Heating.log)"
echo '<br>'
#-----END - Comparison to activate the Heating system-----
```

```
#-----ACTUAL MOVEMENT-----
echo '- Movement detected within the previous 30s:'
microcom -t 100 /dev/ttyUL1 > Movement.log &
echo d > /dev/ttyUL1 #pull movement status
sleep 0.1
if [ "$(cat Movement.log)" -eq "1" ]; then
    echo " <b>Yes</b>"
else
    echo " No"
fi
echo '<br>'
#-----Comparison to control the electric Blinds-----
#pull for LDR and Blinds state
microcom -t 100 /dev/ttyUL1 > LDR.log &
echo l > /dev/ttyUL1 #pull LDR state
sleep 0.1
microcom -t 100 /dev/ttyUL1 > Blinds.log &
echo b > /dev/ttyUL1 #pull Blinds state
sleep 0.1
if [ "$(cat BManual.log)" -eq "0" ]; then #Blinds work in automatic mode
    if [ "$(cat LDR.log)" -eq "0" ]; then #The LDR does not detect light
        echo "- The state of the light sensor is: Night<br>"
        #The LDS does not detect light and Blids are UP
        if [ "$(cat Blinds.log)" -eq "1" ]; then
            echo '- The State of the Blinds is: <b>Up</b>'
            echo '<b> - Is dark outside!</b><br>'
            echo " - The Blinds are being closed...<br>"
            # Closes the Blinds
            echo 1 > /sys/class/gpio/gpio243/value #roll Blinds down (Relay OF
        else
            echo "- The State of the Blinds is: Down"
            echo 0 > /sys/class/gpio/gpio243/value
        fi
    else #The LDR detects LIGHT and Blinds are down
        echo "- The state of the light sensor is: <b>Day</b><br>"
        if [ "$(cat Blinds.log)" -eq "0" ]; then
            echo "- The State of the Blinds is: Down"
            echo '<b> - There is plenty of light outside!</b><br>'
            echo " - The Blinds are being opened...<br>"
            # Opens the Blinds
            echo 0 > /sys/class/gpio/gpio243/value #roll Blinds up (Relay ON)
        else
            echo "- The State of the Blinds is: <b>Up</b>"
            echo 1 > /sys/class/gpio/gpio243/value
        fi
    fi
else # The Blinds work in MANUAL mode
    if [ "$(cat Blinds.log)" -eq "1" ]; then
        echo '- The State of the Blinds is: <b>Up</b>'
        if [ "$(cat BDesired.log)" -eq "0" ]; then #Roll down Blinds
            echo " - The Blinds are being closed...<br>"
            echo 1 > /sys/class/gpio/gpio243/value #Roll Blinds down (Relay OFF)
        fi
    else
        echo "- The State of the Blinds is: <b>Up</b>"
        echo 1 > /sys/class/gpio/gpio243/value
    fi
fi
```

```
else # The Blinds work in MANUAL mode
  if [ "$(cat Blinds.log)" -eq "1" ]; then
    echo '- The State of the Blinds is: <b> Up</b>'
    if [ "$(cat BDesired.log)" -eq "0" ]; then #Roll down Blinds
      echo " - The Blinds are being closed...<br>"
      echo 1 > /sys/class/gpio/gpio243/value #Roll Blinds down (Relay OFF)
    fi
  else
    echo '- The State of the Blinds is: Down'
    if [ "$(cat BDesired.log)" -eq "1" ]; then #Roll up Blinds
      echo " - The Blinds are being opened...<br>"
      echo 0 > /sys/class/gpio/gpio243/value #Roll Blinds up (Relay ON)
    fi
  fi
fi

echo '<br>'
#-----END - Comparison to control the electric Blinds-----
echo '<hr>'
#-----END - SHOW ACTUAL VALUES-----
#-----ENTER VALUES USING FORM-----
echo "<form method=GET action=\"${SCRIPT}\">\\"
  '<table nowrap>'
  # -- Heating related values -----
  echo "<i> Configure the desired options for the Heating system</i><br>"
  echo '<tr><td>Desired Temperature </TD><TD><input type="text" name="desired_temp" \\'
  'size=6 value=$(cat DTemp.log)><sup>o</sup></sup></TD></tr></table>'
  echo 'Configure Heating Mode:'
  echo '<br>'
  if [ "$(cat HManual.log)" -eq "0" ]; then
    echo '<input type="radio" name="Heating_manual" value="0" checked> Automatic<br>\'
    '<input type="radio" name="Heating_manual" value="1" checked> On<br>'
  else
    echo '<input type="radio" name="Heating_manual" value="0"> Automatic<br>\'
    '<input type="radio" name="Heating_manual" value="1" checked> On<br>'
  fi
  echo '<hr>'
  #--Blinds Manual Mode values -----
  echo "<i> Configure the desired options for the Blinds control system</i><br>"
  echo 'Configure Blinds Mode:'
  echo '<br>'
  if [ "$(cat BManual.log)" -eq "0" ]; then
    echo '<input type="radio" name="Blinds_manual" value="0" checked> Automatic<br>\'
    '<input type="radio" name="Blinds_manual" value="1" checked> Manual<br>'
  else
    echo '<input type="radio" name="Blinds_manual" value="0"> Automatic<br>\'
    '<input type="radio" name="Blinds_manual" value="1" checked> Manual<br>'
  fi
  #Blinds manual mode:
  echo "-Options: "
  echo "<br>"
  if [ "$(cat BDesired.log)" -eq "1" ]; then #UP
    echo '<input type="radio" name="Blinds_desired" value="1" checked> Up\'
    '<input type="radio" name="Blinds_desired" value="0"> Down'
  else #DOWN
    echo '<input type="radio" name="Blinds_desired" value="1"> Up\'
    '<input type="radio" name="Blinds_desired" value="0" checked> Down'
  fi
  echo '<br>'
  echo '<hr>'

```



```

#--Lights Manual Mode values -----
echo "<i> Configure the desired options for the Lights control system</i><br>"
echo 'Configure Room Light:'
echo '<br>'
if [ "$(cat RLight.log)" -eq "0" ]; then
    echo '<input type="radio" name="Room_light" value="0" checked> Off<br>'
    echo '<input type="radio" name="Room_light" value="1" checked> On<br>'
    echo 0 > /sys/class/gpio/gpio241/value #turn off Lights Relay
else
    echo '<input type="radio" name="Room_light" value="0"> Off<br>'
    echo '<input type="radio" name="Room_light" value="1" checked> On<br>'
    echo 1 > /sys/class/gpio/gpio241/value #turn on Lights Relay
fi
echo '<hr>'
echo '<br><input type="submit" value="Send to FPGA">'
echo '</form>'
#-----END - ENTER VALUES USING FORM-----

if [ "$REQUEST_METHOD" != "GET" ]; then
    echo "<hr>Script Error:\\"
    echo "<br>Usage error, cannot complete request, REQUEST_METHOD!=GET.\\"
    echo "<br>Check your FORM declaration and be sure to use METHOD=\\\"GET\\\".<br>"
    exit 1
fi

if [ -z "$QUERY_STRING" ]; then #If no values entered, do nothing...
    exit 0
else #If some values have been entered, saves them in variables...
    echo '<meta http-equiv="refresh" content="0;url=web17.sh" />'
    # Just extract the data entered using "sed" command:
    DTemp=`echo "$QUERY_STRING" | sed -n 's/^.*desired_temp=\([^\&]*\).*$/\1/p' | sed "s/%20/ /g"`
    #YY=`echo "$QUERY_STRING" | sed -n 's/^.*val_y=\([^\&]*\).*$/\1/p' | sed "s/%20/ /g"`
    Hmanual=`echo "$QUERY_STRING" | sed -n 's/^.*Heating_manual=\([^\&]*\).*$/\1/p' | sed "s/%20/ /g"`
    Bmanual=`echo "$QUERY_STRING" | sed -n 's/^.*Blinds_manual=\([^\&]*\).*$/\1/p' | sed "s/%20/ /g"`
    Bdesired=`echo "$QUERY_STRING" | sed -n 's/^.*Blinds_desired=\([^\&]*\).*$/\1/p' | sed "s/%20/ /g"`

    RLight=`echo "$QUERY_STRING" | sed -n 's/^.*Room_light=\([^\&]*\).*$/\1/p' | sed "s/%20/ /g"`
    #Save the values in the different .log files to read them in every refresh of the webpage
    echo "$DTemp" > DTemp.log #XX°C
    echo "$Hmanual" > HManual.log #1/0
    echo "$Bmanual" > BManual.log #1/0
    echo "$Bdesired" > BDesired.log #1/0
    echo "$RLight" > RLight.log #1/0

fi
echo '</body>'
echo '</html>'

exit 0
    
```

- Aplicación de adquisición de datos (Arduino)

```
#include "DHT.h"
#define DHTPIN 2
#define B_SUP_PIN 10
#define B_INF_PIN 11
#define PIR_PIN 12
#define LED_PIN 13
#define DHTTYPE DHT22 // DHT 22 (AM2302), AM2321
float ATemp = 21.57;
int humidity_read = 99, ent1 = 0, ent2 = 0, dec1 = 0, dec2 = 0, entero = 0;
char value = '0';
bool movement_detected = HIGH, Blinds = HIGH, Light = HIGH, temperature_to_char = HIGH, two_integers = HIGH;
char buf_ent1[50], buf_ent2[50], buf_dec1[50], buf_dec2[50], buf_h1[50], buf_h2[50];
DHT dht(DHTPIN, DHTTYPE);
void setup()
{
  Serial.begin(9600);
  pinMode(B_SUP_PIN, INPUT_PULLUP);
  pinMode(B_INF_PIN, INPUT_PULLUP);
  dht.begin();
}

void loop() {
  //-----Send over UART the requested data-----
  if (Serial.available())
  {
    value = Serial.read();

    //Humidity requested-----
    if (value == 'h')
    {
      humidity_read = (int)(dht.readHumidity()+0.5f);
      delay(50);
      digitalWrite(LED_PIN, HIGH);
      int entero = humidity_read;
      int ent1 = (int)entero / 10; //Primer número por la izquierda
      int ent2 = (int)(entero - ent1 * 10); //Segundo número por la izquierda
      sprintf(buf_h1, "%d", ent1);
      sprintf(buf_h2, "%d", ent2);
      delay(50);
      Serial.print(buf_h1);
      delay(50);
      Serial.print(buf_h2);
      delay(50);
    }
    // STATE OF MOVEMENT REQUESTED-----
    else if (value == 'd')
    {
      digitalWrite(LED_PIN, HIGH);
      if (digitalRead(PIR_PIN) == HIGH) {
        delay(50);
        Serial.print('1');
      }
      else {
        delay(50);
        Serial.print('0');
      }
    }
  }
}
```

```
//TEMPERATURE WITH 2 DECIMALS REQUESTED -----
else if (value == 'a') //TEMPERATURE WITH 2 DECIMALS
{
    digitalWrite(LED_PIN, HIGH);
    //Reads temperature and stores its value as different characters to send over UART
    ATemp = dht.readTemperature();
    two_integers=LOW;
    entero = ATemp * 10;
    ent1 = (int)(entero / 100); //Primer número por la izquierda
    // Serial.println(ent1);
    ent2 = (int)(entero - ent1 * 100) / 10; //Segundo número por la izquierda
    // Serial.println(ent2);
    dec1 = (int)(entero - ent1 * 100 - ent2 * 10); //Primer decimal
    // Serial.println(dec1);
    entero = (ATemp * 100);
    dec2 = (int)(entero - ent1 * 1000 - ent2 * 100 - dec1 * 10); //Segundo decimal
    // Serial.println(dec2);
    sprintf(buf_ent1, "%d", ent1);
    sprintf(buf_ent2, "%d", ent2);
    sprintf(buf_dec1, "%d", dec1);
    sprintf(buf_dec2, "%d", dec2);
    if (buf_ent1[0]!='0'){two_integers=HIGH;}
    if (two_integers == HIGH)
    {
        Serial.print(' ');

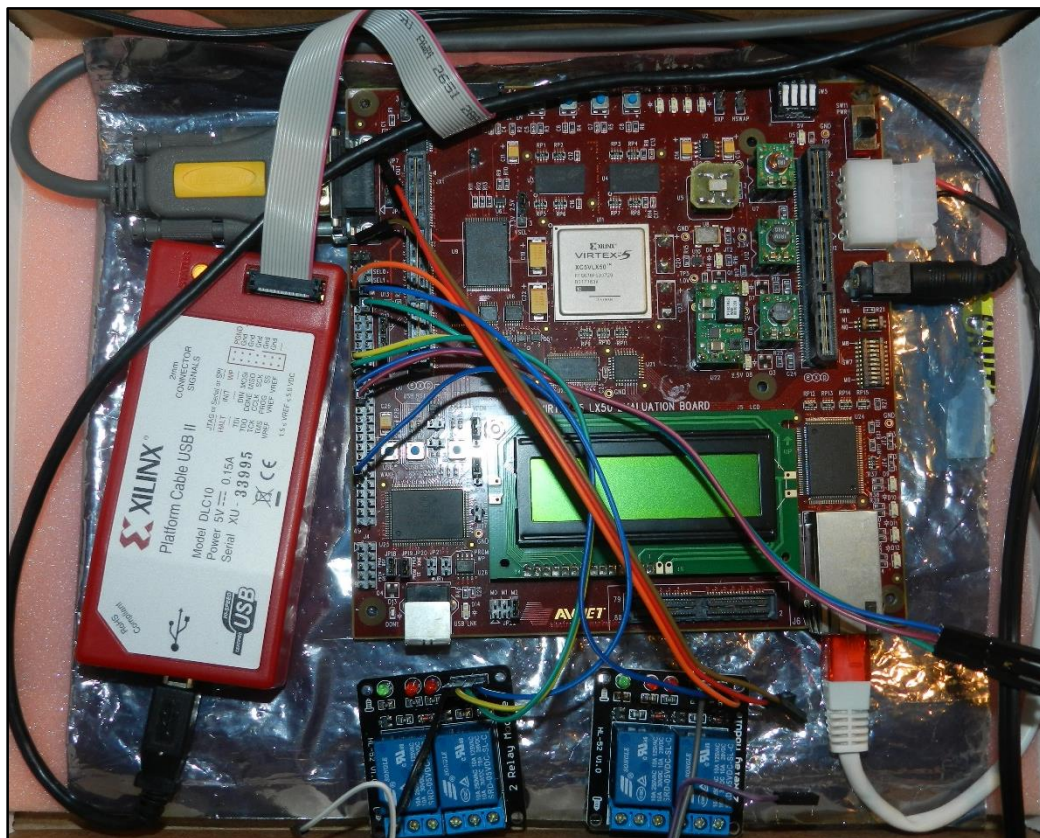
        delay(50);
        Serial.print(buf_ent1);
    }
    delay(50);
    Serial.print(buf_ent2);
    delay(50);
    Serial.print('.');
    delay(50);
    Serial.print(buf_dec1);
    delay(50);
}
//TEMPERATURE WITHOUT DECIMALS REQUESTED -----
else if (value == 'r')
{
    if (two_integers == HIGH)
    {
        delay(50);
        Serial.print(buf_ent1);
        delay(50);
    }
    Serial.print(buf_ent2);
    delay(150);
}
//STATE OF LIGHT SENSOR REQUESTED -----
else if (value == 'l') //LIGHTS
{
    if (analogRead(A1) > 620)
    { delay(50);
      Serial.print('0');//poca luz
    }
    else
    { delay(50);
      Serial.print('1');//bastante luz
    }
}
}
```

Implementación de un kernel Linux sobre un procesador tipo software utilizando una FPGA Daniel Vaquerizo Cid

```
//STATE OF THE BLINDS REQUESTED -----  
else if (value == 'b')  
{  
  bool B_SUP_State = digitalRead(B_SUP_PIN);  
  bool B_INF_State = digitalRead(B_INF_PIN);  
  if ((B_SUP_State == LOW) & (B_INF_State == LOW)) //Blinds closed  
  {  
    delay(50);  
    Serial.print('0');  
  }  
  else //Blids open or mid opened  
  { delay(50);  
    Serial.print('1');  
  }  
}  
else {  
  digitalWrite(LED_PIN, LOW);  
}  
}  
}
```

A continuación, se puede ver el montaje empleado para las diferentes pruebas llevadas a cabo durante el desarrollo de este proyecto.

- Montaje de la FPGA y relés



- **Montaje del Arduino y sensores**

