Uso de la infraestructura docente MIPSfpga v2.0 en la asignatura *Arquitectura de Sistemas Integrados*⋆

Daniel Chaver[1], Yuri Panchul[2], Enrique Sedano[2], David M. Harris[3], Robert Owen[2], Zubair L. Kakakhel[2], Bruce Ableidinger[2], Sarah L. Harris[4]

(1) Grupo ArTeCS, Universidad Complutense de Madrid
(2) Imagination Technologies Ltd., Kings Langley, United Kingdom
(3) Harvey Mudd College, Claremont, CA, USA
(4) Electrical and Computer Engineering, University of Nevada, USA

Resumen: En este artículo se describe el uso de la infraestructura docente MIPSfpga v2.0 en las prácticas de la asignatura *Arquitectura de Sistemas Integrados*, una asignatura obligatoria en el *Grado en Ingeniería Electrónica de Comunicaciones* que se imparte en la Universidad Complutense de Madrid.

Palabras Clave: MIPS, FPGA, Arquitectura de Computadores, Infraestructura Docente

Abstract: In this paper we describe the use of the MIPSfpga v2.0 teaching infrastructure for the labs included in the course *Integrated Systems Architecture*, a compulsory subject in the fourth year of the *Electronic Engineering of Communications* degree offered at University Complutense of Madrid.

Keywords: MIPS, FPGA, Computer Architecture, Teaching Infrastructure

1 Introducción

En Junio de 2015, Imagination Technologies publicó la primera versión (v1.0) del proyecto MIPSfpga [1, 2], que consta de 3 paquetes (MIPSfpga GSG, MIPSfpga Fundamentals y MIPSfpga SoC), y que incluye como componente central el procesador soft-core microAptiv de MIPS en código abierto. El primero de los paquetes, denominado MIPSfpga Getting Started Guide (GSG), incluye, además del propio procesador soft-core, otros componentes, como una extensa guía de

⋆ Este paper se basa en un artículo anterior titulado "Practical experiences based on MIPSfpga", publicado en el Workshop on Computer Architecture Education (celebrado en la conferencia ISCA-2017). Incluye algunas modificaciones: (1) Hemos ampliado la Sección II-D (que en este paper corresponde a la Sección 2.4) y la Sección III-A (que corresponde a la Sección 3); (2) Hemos eliminado las Secciones III-B y III-C; (3) Hemos traducido el paper al español.

uso, multitud de documentación, y todas las herramientas sotware necesarias para utilizar la infraestructura. En cuanto al paquete MIPSfpga Fundamentals, incluye 9 prácticas muy completas con sus correspondientes soluciones, en las que se explica cómo configurar el hardware, cómo desarrollar y depurar programas en C y ensamblador de MIPS, o cómo extender el hardware para interactuar con una serie de periféricos. Por último, en el paquete MIPSfpga SoC, se implementa un System on Chip en torno al procesador microAptiv y se carga y ejecuta una versión reducida de Linux.

En Julio de 2017, Imagination Technologies publicó la segunda versión (v2.0) de esta infraestructura, ampliando notablemente MIPSfpga v1.0. La nueva versión se encuentra disponible a través de [3] y se describe en gran profundidad en [4] y [5]. En concreto, el paquete MIPSfpga GSG ha sido extendido con nuevas funcionalidades, entre las que destacan las siguientes: posibilidad de utilizar la placa FPGA prescindiendo del depurador BusBlaster, soporte para el Sistema Operativo Linux, o disponibilidad de los módulos de alto nivel en lenguaje VHDL. En cuanto al segundo de los paquetes, que es el que ha experimentado mayores modificaciones y que en la v2.0 se pasa a denominar MIPSfpga Labs, incluye 17 nuevas prácticas, en las que se examina la ruta de datos del procesador, la jerarquía de memoria, la modificación del core en diversas formas, o el uso de Entrada/Salida basada en interrupciones.

En este artículo, describimos en primer lugar (Sección 2) cada uno de los paquetes que componen MIPSfpga v2.0, y analizamos a continuación un caso de uso de esta infraestructura (Sección 3). Por último, comparamos MIPSfpga con otras alternativas docentes (Sección 4) y extraemos las principales conclusiones de este trabajo (Sección 5).

2 Análisis de MIPSfpga v2.0

Como hemos mencinado en la Introducción, MIPSfpga v2.0 consta de tres paquetes: MIPSfpga GSG, MIPSfpga Labs y MIPSfpga SoC. En esta sección explicaremos brevemente el contenido de cada uno de ellos (en [4] se incluye una descripción mucho más detallada) y analizaremos cómo de bien se adapta esta infraestructura docente a las *IEEE/ACM Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering* [6].

2.1 MIPSfpga GSG

El paquete MIPSfpga Getting Started Guide (GSG) nos proporciona acceso a un soft-core comercial de MIPS para su uso en una FPGA. Además, el paquete incluye los instaladores para las herramientas de programación y depuración, una guía detallada sobre el uso de la infraestructura MIPSfpga, y un conjunto de scripts y ejemplos prácticos.

El hardware necesario para utilizar MIPSfpga es una placa FPGA y un depurador Bus Blaster. Las placas que se utilizan como ejemplo en la guía de usuario son la Nexys4 DDR de Digilent y la DE2-115 de Terasic, pero se proporciona

también información detallada sobre cómo portar el sistema a otras placas más pequeñas y asequibles (como una Basys3 de Digilent o una DE0 de Altera). En cuanto al software, todo él gratuito, se necesita instalar una herramienta CAD (Vivado si utilizamos una FPGA de Xilinx o Quartus II si utilizamos una FPGA de Altera), y herramientas de programación y depuración (Imagination Codescape MIPS SDK Essential y OpenOCD respectivamente). Estas herramientas se pueden ejecutar tanto sobre sistemas operativos Windows como Linux.

El soft-core incluido en MIPSfpga es una versión ligeramente reducida del procesador microAptiv-UP, utilizado en el conocido microcontrolador PIC32MZ de Microchip, e implementa la Instruction Set Architecture (ISA) MIPS32r3 en un pipeline de 5 etapas. El core (Figura 1) incluye una Memory Management Unit (MMU) con un Translation Lookaside Buffer (TLB), caches separadas de instrucciones y datos, y diversos interfaces. En el Datasheet del core [7] se pueden ecntrar el resto de especificaciones del mismo.

La Figura 2 muestra el sistema MIPSfpga completo, que incluye el propio soft-core, los periféricos, y el bus AHB-Lite de comunicación entre ambos. Entre los periféricos se incluye la memoria principal, implementada en la block-RAM de la FPGA, y el General-Purpose I/O (GPIO), con el que se gestiona la comunicación con los LEDs, los interruptores o los botones de la placa FPGA. El sistema utiliza una señal de reloj (SI_ClkIn) y una de reset (SI_Reset_N), activa en baja. Además, aunque su uso no es imprescindible, existe un interfaz EJTAG que facilita la carga y permite la depuración de programas en el sistema MIPSfpga.

La memoria incluye dos bloques de memoria, uno a partir de la dirección física 0x00000000 (Code/Data RAM) de 256KB, y otro a partir de la dirección física 0x1fc00000 (Reset RAM) de 128KB, como ilustra la Figura 3. Tras el reset, el procesador comienza buscando instrucciones a partir de la dirección 0x1fc00000. Así pues, como mínimo, esa dirección debe contener instrucciones. Normalmente, dichas instrucciones corresponden al código de arranque que se encarga de inicializar el sistema. Tras ello, se salta al código de usuario, ubicado en el otro bloque de memoria. Aunque este es el procedimiento habitual, en códigos simples que no utilizan ciertos componentes del sistema (como las caches o la TLB), es posible prescindir del código de arranque y ubicar el código de usuario a partir de la dirección física 0x1fc00000 para comenzar la ejecución inmediatamente tras el reset.

2.2 MIPSfpga Labs

Este segundo paquete incluye una serie de prácticas de laboratorio orientadas a la enseñanza de arquitectura de computadores, diseño de SoC, o codiseño HW/SW. En la segunda versión de MIPSfpga se ha ampliado significativamente el conjunto de prácticas y los temas tratados, pasando de las 9 prácticas iniciales a 25 prácticas que cubren todos los aspectos del computador, desde la programación en alto nivel y ensamblador (Parte 1) hasta el sistema de memoria (Parte 4), pasando por la Entrada/Salida (Parte 2) y el Core (Parte 3). La Tabla 1 pro-

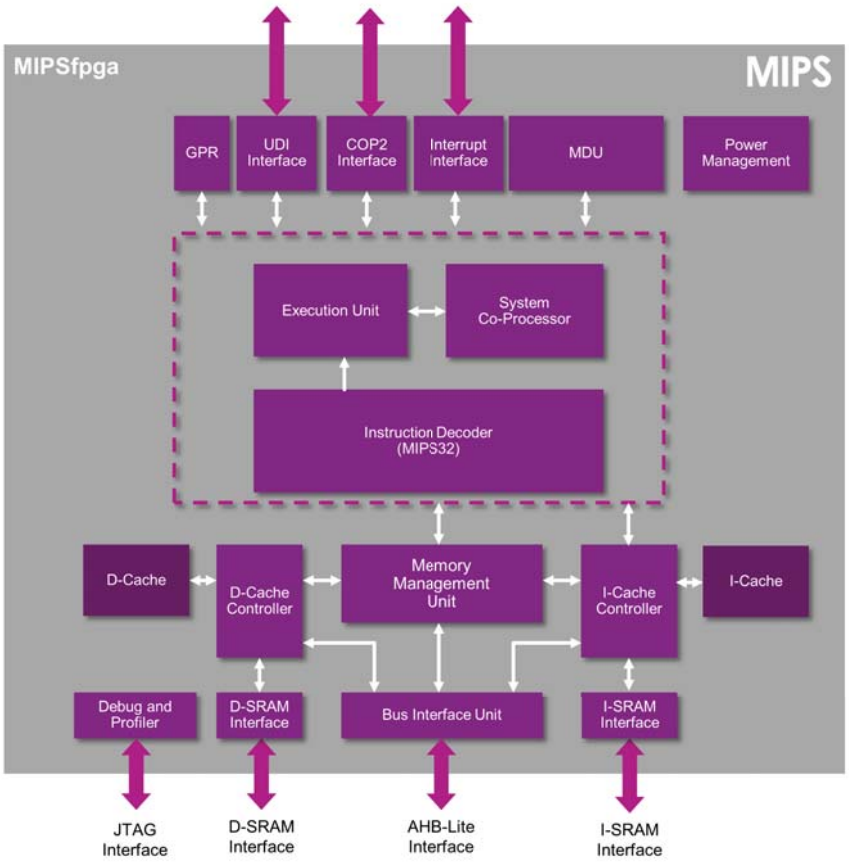

Fig. 1. Core de MIPSfpga

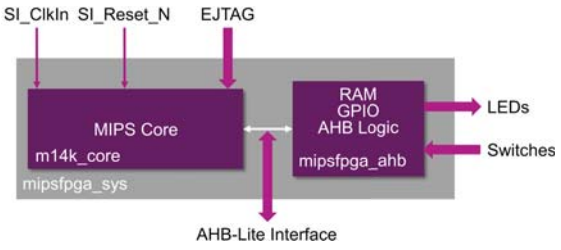

Fig. 2. Sistema MIPSfpga

porciona una breve descripción de estas 25 prácticas, cuya explicación se amplía notablemente en los siguientes párrafos.

La primera parte incluye cuatro prácticas. En la primera, se explica paso a paso cómo crear un proyecto en Vivado (para FPGAs de Xilinx) o en Quartus-

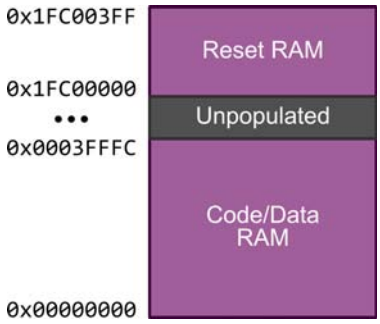

Fig. 3. Memoria física en MIPSfpga

Table 1. Prácticas incluidas en el paquete MIPSfpga Labs

#	Descripción
1	Creación de un proyecto en Vivado o Quartus-II
2	Crear, compilar, depurar y ejecutar programas en lenguaje C
3	Crear, compilar, depurar y ejecutar programas en ensamblador MIPS
4	Más ejercicios de programación en lenguaje C (opcional)
5	Ampliar el sistema con los displays de 7-segmentos de la placa
6	Ampliar el sistema con un contador
7	Ampliar el sistema con un timbre
8	Ampliar el sistema con un sensor de luz SPI
9	Ampliar el sistema con un LCD SPI
10	Comunicación por medio de interrupciones
11	Implementar un motor DMA para la comunicación entre periféricos
12	Implementar un motor Data Encryption Standard (DES)
13	Uso de los Performance Counters
14	Ejecución de una instrucción ADD y otras instrucciones aritméticas
15	Ejecución de una instrucción AND y otras instrucciones lógicas
16	Ejecución de una instrucción LW y otras instrucciones relacionadas
17	Ejecución de una instrucción BEQ y otras instrucciones relacionadas
18	Análisis de la Unidad de Gestión de Riesgos
19	Uso del Interfaz CorExtend
20	Introducción a las caches disponibles en MIPSfpga
21	Análisis de la D$ e implementación de nuevas configuraciones
22	Controlador de Cache: Análisis del acierto y el fallo
23	Controlador de Cache: Análisis de las políticas de gestión de contenido
24	Controlador de Cache: Análsis del Store Buffer y del Fill Buffer
25	Implementación de una Scratchpad

II (para FPGAs de Altera), y cómo portar MIPSfpga a otras placas. En las prácticas 2 y 3 se explica cómo crear, compilar, descargar, ejecutar y depurar programas en C y ensambaldor de MIPS en el sistema MIPSfpga. Por último, la práctica 4 propone más ejercicios de programación en C.

La segunda parte comienza con cinco prácticas sobre Entrada/Salida mapeada en memoria, en las que se da soporte a nuevos periféricos y se interactúa con ellos. Para las prácticas 7, 8 y 9 se requiere de ciertos componentes adicionales especificados en [4]. A continuación, las prácticas 10 a 12 analizan Entrada/Salida basada en interrupciones y DMA. Por último, en la práctica 13 se explica el uso de los Performance Counters disponibles en microAptiv y se proponen algunos programas sencillos en los que se evalúa analítica y experimentalmente el CPI.

Las prácticas de la tercera parte nos sumergen en el core. En las primeras cuatro prácticas (14-17) se añaliza el hardware utilizado en la ruta de datos y en la unidad de control del core para ejecutar los diferentes tipos básicos de instrucciones: Aritmética (ADD), Lógica (AND), Transferencia con Memoria (LW) y Salto Condicional (BEQ). Las cuatro prácticas siguen una estructura similar, comenzando con una extensa explicación teórica, realizando a continuación una simulación detallada de la instrucción correspondiente, y proponiendo por último una serie de ejercicios en los que se llega a implementar nuevas instrucciones (ADDIUPC, SEQ, NAND, SELEQZ/SELNEZ, LWI, LWPC o BC). En la práctica 18 se analiza la Unidad de Gestión de Riesgos. Para ello, se implementa un reloj de baja frecuencia, que nos permite observar a través de los LEDs las señales relacionadas con dicha unidad por medio de la ejecución de varios programas de ejemplo. El análisis en la placa se complementa con una simulación en Vivado de los mismos programas. Por último, en esta tercera parte se incluye una práctica sobre el uso del Interfaz CorExtend de MIPS (práctica 19). Esta funcionalidad permite al diseñador especificar e implementar sus propias instrucciones (User Defined Instructions, o UDIs), acelerando la ejecución de ciertos algoritmos o regiones críticas. La práctica describe en primer lugar el Interfaz, sus características y limitaciones, y su ubicación e interacción con el resto de estructuras del core. A continuación, se proponen una serie de ejercicios, desde los más básicos, en los que se añaden varias instrucciones (SELEQZ, NAND y SEQ), hasta los más avanzados, en los que se incluyen instrucciones de DSP o de punto flotante y se comparan diversos algoritmos por medio de los Performance Counters.

Por último, en la cuarta parte, se explora el sistema de memoria de MIPSfpga. En la práctica 20, al igual que en la 18, se implementa un reloj de baja frecuencia, que nos permite observar a través de los LEDs las señales relacionadas con los aciertos y fallos en la cache de datos, por medio de diversos códigos de ejemplo. En la práctica 21 se analiza el interfaz y organización interna de los arrays que constituyen la cache (i.e. Array de Datos, Array de Tags y Array Way-Select), se implementan y comparan nuevas configuraciones de la cache, en las que se varía el tamaño o la asociatividad, y se prueban diversas técnicas software de optimización del rendimiento. Las prácticas 22 a 24 analizan el controlador de cache. En la 22 se estudia la gestión del acierto y del fallo, primero de forma teórica, luego por medio de una simulación en la que se afianzan los conceptos teóricos, y por último a través de una serie de ejercicios. En la práctica 23 se describen las distintas políticas de gestión de contenido disponibles en las caches de microAptiv (política de reemplazo LRU, diversas políticas de escritura), se

implementan nuevas políticas (por ejemplo, una política de reemplazo FIFO) y se evalúan a través de varios códigos de ejemplo y haciendo uso de los Performance Counters. La práctica 24 explica el funcionamiento del Store Buffer (que almacena temporalmente el dato que escribe un STORE en la cache de datos) y del Fill Buffer (que almacena temporalmente el bloque a escribir en la cache de datos, proveniente de memoria, a consecuencia de un fallo), y propone una serie de ejercicios en los que se deben analizar varias secuencias de acceso a memoria. Por último, en la práctica 25 se añade una Scratchpad de instrucciones, y se compara el rendimiento de un mismo algoritmo ejecutado desde la cache de instrucciones y desde la Scratchpad.

Tras completar las prácticas del paquete MIPSfpga Labs, los estudiantes estarán preparados para afrontar proyectos más avanzados, como añadir periféricos I^2C o UART, o añadir nuevas funcionalidades al core (un prefetcher hardware, un predictor de saltos) y al sistema de memoria (un segundo nivel de cache, un predictor de vías).

2.3 MIPSfpga SoC

El último paquete de MIPSfpga 2.0 se llama MIPSfpga-SoC, y en él se explica cómo implementar un SoC centrado en el soft-core de microAptiv, y posteriormente cargar y ejecutar Linux en dicho SoC. El core de MIPS actúa como maestro de diversos periféricos (esclavos), a través del bus AHB-Lite (Figura 4). Para implementar los periféricos, se utilizan bloques IP de Xilinx (excepto el controlador GPIO, que proporciona Imagination), lo que reduce significativamente el tiempo de desarrollo. Sin embargo, dado que dichos periféricos se comunican a través de un bus Advanced eXtensible Interface (AXI), es necesario incluir un bridge AHB-Lite - AXI.

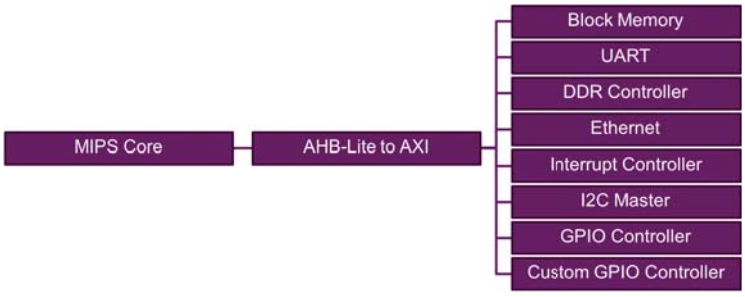

Fig. 4. Diagrama del Linux SoC

El sistema operativo Linux puede dividirse en dos partes: el Userspace y el Kernel. El primero, interactúa con el hardware a través de las llamadas al sistema que implementa el kernel. En este caso, utilizamos Buildroot. Por su parte, el kernel interactúa directamente con el hardware, proporcionando una

capa de abstracción. Se puede implementar el kernel con poco soporte hardware; en este SoC (Figura 4), incluimos una Unidad de Gestión de Memoria (MMU), un controlador de interrupciones, una serie de timers, un UART, una memoria, y un interfaz e-JTAG.

2.4 Adaptación a las *IEEE/ACM Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering*

Las asociaciones IEEE y ACM establecen en [6] once unidades fundamentales para la enseñanza de asignaturas del área de Ingeniería de Computadores, resumidas en la Tabla 2. Como tratamos de justificar en esta sección, creemos que las prácticas incluidas en MIPSfpga v2.0 cubren a la perfección estas unidades.

Table 2. Unidades fundamentales para la docencia en Ingeniería de Computadores establecidas por IEEE y ACM

Unit	**Name**
CE-CAO-1	History and overview
CE-CAO-2	Tools, standards and/or constraints
CE-CAO-3	Instruction set architecture
CE-CAO-4	Measuring performance
CE-CAO-5	Computer arithmetic
CE-CAO-6	Processor organization
CE-CAO-7	Memory system organization and architectures
CE-CAO-8	Input/Output interfacing and communication
CE-CAO-9	Peripheral subsystems
CE-CAO-10	Multi/Many-core architectures
CE-CAO-11	Distributed system architectures

El ISA de MIPS existe desde comienzos de la década de los 80 y sirvió de base a muchas otras arquitecturas posteriores, siendo por tanto indiscutible su papel en la historia de los computadores cubierta en la unidad CE-CAO-1. Las prácticas 2 y 3 del paquete MIPSfpga Labs describen cómo crear proyectos en Vivado, compilar programas en C o ensamblador de MIPS, y ejecutar y depurar estos programas en MIPSfpga, alineandose así perfectamente con las unidades CE-CAO-2 y CE-CAO-3.

En la práctica 13 se explica en detalle el uso de los Performance Counters disponibles en el core microAptiv. A continuación, se proponen una serie de ejercicios en los que se evalúa el rendimiento de distintos programas, tanto analítica como experimentalmente. Por otra parte, este recurso se utiliza ampliamente en las prácticas pertenecientes a las partes 3 y 4 de MIPSfpga Labs (Tabla 1), con el objetivo de medir eventos tales como el número de ciclos, número de instrucciones finalizadas, número de accesos y fallos a la cache de instrucciones/datos, etc. Podemos por tanto afirmar que la unidad CE-CAO-4 queda cubierta en esta infraestructura docente.

En cuanto a la unidad 5 (CE-CAO-5) de las recomendaciones del IEEE/ACM, ésta se cubre parcialmente en las prácticas 14 y 19. En la primera, se analiza en detalle la ejecución de una instrucción ADD, se analizan las señales de control relacinadas con las instrucciones aritméticas implementadas en microAptiv, se estudia la unidad aritmética incluida en este procesador, y se implementan nuevas instrucciones aritméticas. En la práctica 19, que analiza el Interfaz CorExtend, se propone un extenso ejercicio sobre aritmética en punto flotante, en el que se incluyen a través de este interfaz nuevas instrucciones de suma, multiplicación y división en punto flotante, se utilizan estas nuevas instrucciones para implementar el algoritmo de la bisección (que permite calcular las raices de una función), y se compara, por medio de los Perfromance Counters, el rendimiento de este algoritmo con el de uno en el que las operaciones en punto flotante se emulan por software.

La unidad CE-CAO-6 se cubre completamente con las prácticas 14 a 18. En estas prácticas, se analiza en gran detalle la organización del procesador microAptiv, y se compara con el procesador segmentado utilizado en [8]. Así, las prácticas comienzan con una introducción teórica, en la que se describe la unidad de control y la ruta de datos desde el punto de vista de los distintos tipos de instrucciones básicas: instrucciones aritméticas (práctica 14); instrucciones lógicas (práctica 15); instrucciones de transferencia con memoria (práctica 16); e insrucciones de salto condicional (práctica 17). Por su parte, en la práctica 18, se analiza teóricamente la Unidad de Riesgos. Después de esta profunda descripción, se realiza una simulación de la instrucción estudiada, y se propone un amplio conjunto de ejercicios.

Las prácticas 20 a 25 analizan exhaustivamente el sistema de memoria de MIPSfpga, cubriendo así la unidad CE-CAO-7. En la Sección 2.2 describimos en detalle cada una de estas prácticas. Entre muchas otras cosas, se comparan distintas configuraciones de cache, se prueban diferentes políticas de gestión de contenido, se analiza la gestión de aciertos y fallos, o se implementa una Scratchpad de instrucciones.

Las unidades CE-CAO-8 y CE-CAO-9 se cubren en las prácticas 5 a 12. En un primer grupo de prácticas (5-9), se añaden distintos periféricos, algunos de los cuales se comunican por medio del extendido bus SPI. Un segundo grupo de prácticas (10-12) introduce el uso de interrupciones y DMA para la Entrada/Salida.

Las unidades CE-CAO-10 y CE-CAO-11 no se cubren directamente en MIPSfpga, pues se centran en sistemas multi-core y en sistemas distribuidos (MIPSfpga es un sistema single-core). Sin embargo, la disponibilidad de un sistema completamente abierto y no ofuscado, hace que se puedan afrontar ejercicios y prácticas relacionados con estas unidades a modo de proyectos avanzados. Así, por ejemplo, en [11], los autores realizan profundas modificaciones al core microAptiv para implementar un multi-procesador de memoria distribuida de 120 cores.

3 Caso de uso de MIPSfpga v2.0

En esta sección describimos la utilización de MIPSfpga en la asignatura *Arquitectura de Sistemas Integrados* durante el curso 2016-17.

3.1 Resumen de la asignatura

La analizada en este artículo es una asignatura obligatoria del segundo cuatrimestre del cuarto curso de la titulación *Ingeniería Electrónica de Comunicaciones*, grado que se imparte desde hace cinco años en la Universidad Compulutense de Madrid (UCM). El objetivo fundamental de la misma es que los estudiantes adquieran conocimientos avanzados sobre la implementación y gestión del procesador, el sistema de memoria y la Entrada/Salida en los sistemas empotrados, microprocesadores y microcontroladores actuales.

En el curso 2016-17, se realizaron un total de 12 sesiones prácticas, cada una de dos horas, y 26 clases de teoría, de una hora cada una (nótese que estos números pueden variar ligeramente cada curso, dependiendo de diversos factores, por lo que pueden ser necesarias pequeñas modificaciones a lo que aquí exponemos). La planficación establecía que cada semana se realizara una sesión de laboratorio y se impartieran dos clases teóricas. Al tratarse de una asignatura con alta carga práctica (no en vano las práctias van a suponer un 50% de la calificación final), tratamos de coordinar y sincronizar de forma muy precisa ambas actividades, de forma que cada sesión de laboratorio reforzase los conceptos que se estaban tratando en la parte teórica.

3.2 Programa

Los alumnos que se matriculan en esta asignatura tienen conocimientos avanzados de diseño digital, programación en VHDL, y estructura de computadores (ISA de MIPS, procesadores mono- y multi-ciclo, sistema de Entrada/Salida), así como conocimientos básicos de programación en C++ y Sistemas Operativos. Por tanto, dada la formación inicial de los estudiantes, se puede plantear un programa ambicioso, tanto desde el punto de vista teórico como en lo que se refiere a las prácticas.

El programa se divide en cuatro módulos. En el Módulo 1 se revisan contenidos que los estudiantes ya han estudiado y deben conocer, como el ISA de MIPS, implementaciones mono- y multi-ciclo del procesador, y los conceptos básicos sobre jerarquía de cache y Entrada/Salida. Es por tanto un módulo breve, al que no se deben destinar más de 3 o 4 clases teóricas. En el Módulo 2 se describe detalladamente el procesador segmentado de [8], desde el punto de vista teórico primero, y con ejemplos y ejercicios después, y se analizan algunas técnicas microarquitectónicas de alto rendimiento, como la ejecución fuera de orden y superescalar, la predicción de saltos, el renombrado de registros, y algunas otras. Se deben destinar aproximadamente 10 clases teóricas y de problemas a este módulo. El Módulo 3 explora la jerarquía de la cache y la implementación de la memoria virtual. Los estudiantes ya deben conocer la gestión básica de

la cache (emplazamiento directo, política de post-escritura, etc.), por lo que se pueden analizar políticas y técnicas de gestión de la cache avanzadas, como diversas políticas de reemplazo de bloques, caches asociativas y caches multi-nivel, predicción de vías, cache de víctimas, cache no bloqueante, búsqueda de la palabra crítica en primer lugar, optimizaciones del compilador, etc. Al igual que en el Módulo 2, creemos que se deben dedicar alrededor de 10 clases a esta parte. Por último, el Módulo 4 introduce brevemente los Sistemas en Chip (SoC) y los Sistemas Empotrados, por lo que 3 o 4 clases teóricas serán suficientes.

3.3 Bibliografía

Indudablemente, el texto óptimo para esta asignatura es [8], tanto por el contenido teórico y práctico del mismo, como por el hecho de que las prácticas de MIPSfpga v2.0 están completamente alineadas con este libro. Concretamente, nos centramos en los capítulos 4 (*Hardware Description Languages*), 6 (*Architecture*), 7 (*Microarchitecture*) y 8 (*Memory and I/O Systems*). Además, con la compra de este libro se proporcionan conjuntos de trasparencias de cada módulo, que el profesor puede utilizar como apoyo para las clases teóricas. En nuestro caso, hemos extendido notablemente estas trasparencias, intercalando la descripción detallada del procesador y la jerarquía de cache utilizados por el sistema MIPSfpga, y comparándolos con los propuestos en el libro.

Para los problemas, hemos diseñado varias hojas de ejercicios, en las que se combinan problemas extraídos de diversos libros con otros de elaboración propia. Por último, en cuanto al material bibliográfico para la parte práctica de la asignatura, el incluido en MIPSfpga es más que suficiente, pues contiene extensos guiones de prácticas, una completa guía de uso de la infraestructura, y múltiples documentos propios de MIPS en los que se describe el procesador, la memoria, los Performance Counters, el Interfaz CorExtend, etc.

Como textos adicionales, se proponen los dos libros de John L. Hennessy y David A. Patterson [9] y [10], referencias obligadas en cualquier asignatura relacionada con la arquitectura de computadores.

3.4 Organización de las prácticas

Teniendo en cuenta los conocimientos iniciales de los alumnos, el programa de la asignatura y su ubicación en el cuarto curso de grado, creemos que la elección de la infraestructura MIPSfpga para la parte práctica de esta asignatura es totalmente acertada.

Como dijimos en la Sección 3.1, la asignatura incluye una sesión de laboratorio por semana. Durante el curso 2016-17, realizamos 12 sesiones, aunque estos números pueden variar ligeramente en otros cursos. En cualquier caso, no hay tiempo suficiente para completar las 25 prácticas de MIPSfpga Labs, por lo que tendremos que realizar una selección. La Tabla 3 recoge las prácticas elegidas para cada módulo del temario, a saber:

– La *Práctica 1* ilustra cómo crear un proyecto en Vivado, y cómo sintetizar y descargar el sistema MIPSfpga en la placa

- Para complementar la revisión del ISA de MIPS, se realizan las *Prácticas 2, 3 y 4*, en las que se ilustra cómo ejecutar y depurar un programa en MIPSfpga y se proponen varios ejercicios de programación en C y en ensamblador MIPS.
- Para complementar la revisión de la Entrada/Salida, se realiza la *Práctica 5*, en la que se añade el soporte necesario para comunicarse con los displays de 7-segmentos. Un aspecto muy interesante de esta práctica, a diferencia de las que los estudiantes han realizado en cursos anteriores, es que el dispositivo no solo se maneja a nivel software, sino que se implementa también su controlador hardware.
- El Módulo 2 de la asignatura analiza el procesador segmentado:
  - Para complementar la parte teórica, se realiza en primer lugar la *Práctica 13*, en la que se analiza el CPI de varios programas, comparando el rendimiento en el procesador segmentado de [8] con el de microAptiv, y realizando el análisis de forma analítica y, para el caso de MIPSfpga, también de modo experimental (por medio de los Performance Counters).
  - Para reforzar los conceptos teóricos también se realizan las *Prácticas 14-18*, en las que se analiza la ejecución de distintos tipos de instrucciónes, se añaden otras nuevas, y se estudia la unidad de gestión de riesgos del procesador.
- El Módulo 3 se complementa realizando la *Práctica 22*, en la que se analiza en profundidad la gestión de un acierto y un fallo de cache.
- Por último, asociado al Módulo 4, se realiza el Starter Tutorial del paquete MIPSfpga SoC, en el que se guía al estudiante en la creación de un SoC en el que se ejecuta Linux.

Table 3. Asociación entre módulos y prácticas

Módulo	**Contenido**	**Prácticas asociadas**
1	ISA de MIPS	Prácticas 2, 3 y 4
	Procesador mono/multi-ciclo	
	Gestión de la entrada/salida	Práctica 5
2	Procesador segmentado	Prácticas 13-18
3	Jerarquía de memoria	Prácticas 22-A y 22-B
4	Sistemas empotrados y SoC	MIPSfpga SoC - Starter Tutorial

Aunque, desgraciadamente, por falta de tiempo, muchas prácticas interesantes de MIPSfpga para complementar el temario se quedan en el tintero (como las Prácticas 10 y 11, en las que se estudia la gestión de la Entrada/Salida con interrupciones y DMA respectivamente; la Práctica 19, en la que se analiza el Interfaz CorExtend; la Práctica 21, en la que se experimenta con distintas configuraciones de cache; o la Práctica 23, en la que se analizan distintas políticas

de gestión de contenido), creemos que el subconjunto seleccionado cubre con bastante eficacia los conceptos más importantes de la asignatura.

La Tabla 4 muestra la planifiación específica empleada durante el curso 2016-17. En la primera clase, se reparten a los alumnos las FPGA y los BusBlaster, que conservarán durante todo el cuatrimestre para poder trabajar por su cuenta, y les pedimos que descarguen los tres paquetes de la web de Imagination Technologies [3], instalen las herramientas software en su portátil (siguiendo las indicaciones de MIPSfpga GSG), y comprueben que todo funciona correctamente. Además, se indica a los estudiantes que deben comenzar a estudiar por su cuenta el Capítulo 4 de [8], pues MIPSfpga utiliza lenguaje Verilog (salvo para los módulos de alto nivel, que están tanto en Verilog como en VHDL) y sus conocimientos se limitan a VHDL.

Table 4. Planificación específica del curso 2016-17

Sesión	**Descripción**
-	Casa: Instalar MIPSfpga antes de la primera sesión
1	Terminar instalación + **Práctica 1**
2	**Práctica 2** y **3** (C y ensamblador) + **Práctica 4**
-	Casa: Terminar Práctica 4
3	Ejercicio extra Práctica 4 + Test
4	**Práctica 5** (Displays de 7-segmentos)
-	Casa: Completar Práctica 5
5	Ejercicio extra Práctica 5 + Test
6	**Práctica 13** (Performance Counters)
-	Casa: Completar Práctica 13
7	Ejercicio extra Práctica 13 + Test
8	Trabajo en grupo (**Prácticas 14-18**)
-	Casa: Completar Prácticas 14-18
9	**Práctica 22-A** (Gestión del acierto cache)
-	Casa: Completar Práctica 22-A
10	**Práctica 22-B** (Gestión del fallo cache)
-	Casa: Completar Práctica 22-B
11	Ejercicio extra Práctica 22-B + Test
12	**MIPSfpga SoC** - Advanced Tutorial

Como se observa en la Tabla 4, cada práctica se realiza en 1 o 2 sesiones. Por ejemplo, la Práctica 13 se realiza en las sesiones 6 y 7: en la sesión 6 los alumnos comienzan a trabajar la práctica; luego, en el intervalo hasta la siguiente sesión, trabajan la práctica por su cuenta; y, para terminar, en la sesión 7, completan la práctica y responden a un examen individual sobre la misma. Como excepción, las Prácticas 14-18 se realizan en grupos de 3-4 miembros. Cada uno de los grupos realiza una práctica, prepara un trabajo en su casa, y lo expone al resto de estudiantes en una clase teórica.

3.5 Evaluación

La calificación final se calcula del siguiente modo: 0.5*EF + 0.3*PL + 0.2*TG; donde EF corresponde a la calificación del examen final, PL resulta de la calificación del laboratorio, y TG corresponde a la calificación del trabajo en grupo. Es importante evaluar adecuadamente el trabajo de laboratorio, pues la mitad de la calificación de la asignatura resulta de esta actividad, por lo que al finalizar cada práctica se realiza un examen individual sobre la misma.

3.6 Opinión de los alumnos

Para finalizar, incluimos a continuación algunas opiniones de alumnos que realizaron esta asignatura durante el curso 2016-17: "El curso, y en especial las prácticas, permiten conocer a fondo la arquitectura y la microarquitectura de un computador" (G. Diaz-Tejeiro); "Las sesiones de laboratorio nos han permitido entender muy bien el funcionamiento de un procesador comercial" (M. Sanchez); "El texto de las prácticas facilita el aprendizaje gracias a su amplio contenido" (J. Martin); "En un principio, resultaba muy complicado adaptarse a esta forma de aprendizaje. Sin embargo, una vez que nos acostumbramos, empezamos a disfrutar la asignatura, logrando entender a fondo cómo trabajan el core, la cache, la entrada/salida, etc. En mi opinión, es la mejor manera de estudiar una asignatura como esta." (A. Menendez); "Las prácticas nos permitieron afianzar los conceptos teóricos" (P. Fernandez); "Las prácticas nos mostraron cómo aplicar los conceptos de arquitectura de computadores al mundo real" (A. Villarin).

4 Trabajo relacionado

En la actualidad, podemos encontrar una gran cantidad de procesadores soft-core de muy diversas características. En esta sección, los analizamos brevemente y los comparamos con MIPSfpga. Las principales compañías de desarrollo y comercialización de FPGAs (Altera y Xilinx), ofrecen sus propios soft-cores (Nios/NiosII [12] y MicroBlaze [13] respectivamente), configurados específicamente para sus propias FPGAs. Estas alternativas presentan diversas desventajas: no son de código abierto, lo cual limita su uso significativamente; no están basados en soft-cores industriales/comerciales ni en un ISA comercial; y carecen de material docente extenso y de calidad. Por su parte, ARM también proporciona una opción de código no abierto, el Cortex M0 Design Start [14], un soft-core muy básico (compuesto únicamente de 8K puertas) y de bajo rendimiento. Se trata de un soft-core con el código ofuscado y con un soporte para depuración muy limitado (no incluye EJTAG). Además, no proporciona la posibilidad a la comunidad académica de integrarlo en silicio y, al igual que los dos anteriores, carece de buen material docente.

Existe también disponibilidad de soft-cores de código abierto. Dos opciones muy conocidas, ambas basadas en el ISA SPARC, son la familia OpenSPARC [15] y la familia LEON [16], desarrollados actualmente por Oracle (originalmente por

Sun Microsystems) y por Aeroflex Gaisler (originalmente por la European Space Agency), respectivamente. Si bien son alternativas interesantes, al igual que las descritas anteriormente no incluyen buen material docente, y están basadas en un ISA menos extendido en el mundo académico que el ISA de MIPS o el de ARM. Por último, otras dos alternativas que debemos mencionar son RISC-V [17], desarrollada originalmente por la Universidad de California, Berkeley, y openRISC, desarrollada por opencores.org [18]. Estos soft-cores están basados en ISAs no comerciales y, al igual que los anteriores, proporcionan escaso material docente.

MIPSfpga, por su parte, soluciona todas las limitaciones anteriores. Incluye un procesador soft-core industrial, de código no ofuscado, utilizado en diversos dispositivos comerciales, como el conocido PIC32MZ de Microchip. Dicho soft-core utiliza el release 3 del ISA de MIPS, ampliamente empleado en el mundo académico y con disponibilidad de multitud de documentación y soporte docente. MIPSfpga también proporciona gran cantidad de documentación, entre la que se incluye mucho material docente teórico y práctico. Gran parte de esta documentación está disponible en 5 idiomas (castellano entre ellos). Además, MIPSfpga incluye soporte para diversas FPGAs, tanto de Xilinx como de Altera, y es fácilmente extensible a otras FPGAs. Por último, debemos mencionar que Imagination Technologies ha cerrado un acuerdo con Europractice y MOSIS para ofrecer, al mundo académico, la posibilidad de integrar en silicio dos cores muy similares a microAptiv (los cores Warrior M-class 5100 y 5150).

5 Conclusiones

En este paper hemos descrito en gran profundidad la última versión de MIPSfpga (v2.0), publicada en julio de 2017, así como su aplicación a la enseñanza de la asignatura *Arquitectura de Sistemas Integrados* impartida en la UCM. Esta completa infraestructura ofrece gran cantidad de material docente, que permite cubrir perfectamente los temarios de asignaturas de Arquitectura de Computadores o Diseño SoC. Además, permite a los estudiantes enfrentarse a problemas similares a los que debe solucionar un ingeniero de computadores en la industria actual, por lo consideramos que puede ser especialmente adecuada para asignaturas avanzadas de grado (como la analizada en este paper) o asignaturas de máster.

6 Agradecimientos

Los autores agradecen la contribución de Imagination University Program, University of Nevada, Las Vegas, Imperial College London (UK), Grupo ArTeCS de la Universidad Complutense de Madrid y contratos TIN2015-65277-R, TIN2015-65316-P y Artículo-83 (nº 411-2016), Munir Hasan (IMG UK), Prashant Deokar (IMG India), Mahesh Firke (IMG India) Parimal Patel (Xilinx), Kent Brinkley (IMG USA), Rick Leatherman (IMG USA), Chuck Swartley (IMG USA), Sean Raby (IMG UK), Michio Abe (IMG Japan), Bingli Wang (IMG China),

Sachin Sundar (IMG USA), Alex Wong (Digilent Inc.), Matthew Fortune (IMG UK), Jeffrey Deans (IMG UK), Laurence Keung (IMG UK), Roy Kravitz (Portland State University), Dennis Pinto (UCM), Tejaswini Angel (Portland State University), Christian White, Gibson Fahnestock, Jason Wong, Cathal McCabe (Xilinx), Larissa Swanland (Digilent).

References

1. Kakakhel, Z., Harris, S., Harris, D.,: 'MIPSfpga: An unobfuscated commercial MIPS core and SoC that runs Linux". Embedded World 2016, Nuremberg, Germany, February 2016.
2. Harris, S., Owen, R., Sedano, E., Chaver, D.,: 'MIPSfpga: Hands-On Learning on a Commercial Soft-Core". European Workshop on Microelectronics Education, Southampton, UK, May 2016.
3. 'Imagination University Program - Teaching Resources'. https://community.imgtec.com/university/resources.
4. Harris, S., Harris, D., Chaver, D., et al.: 'MIPSfpga: Using a Commercial MIPS Soft-Core in Computer Architecture Education.'. IET Circuits, Devices and Systems, March 2017.
5. Chaver, D., et al.: 'Practical experiences based on MIPSfpga'. Workshop on Computer Architecture Education 2017 (held in conjunction with ISCA-17), June 2017.
6. 'CE2016 - Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering' IEEE and ACM, 2016.
7. Imagination Technologies Ltd., 'MIPS32® microAptiv™ UP Processor Core Family Datasheet', July 31, 2013
8. Harris, D., and Harris, S., 'Digital Design and Computer Architecture', Elsevier Science and Technology, 2a edición, 2012
9. Patterson, David A., and Hennessy, John L., 'Computer Organization and Design', Morgan Kaufmann, 5a edición, 2013
10. Hennessy, John L., and Patterson, David A., 'Computer Architecture: A Quantitative Approach', Morgan Kaufmann, 5a edición, 2011
11. Kumar H B, C., Ravi, P., Modi, G., Kapre, N..: '120-core microAptiv MIPS Overlay for the Terasic DE5-NET FPGA board', Int. Symp. on Field-Programmable Gate Arrays, Monterey, USA, February 2017
12. 'Altera - NIOS-II Processor', https://www.altera.com/products/processors/overview.html
13. 'Xilinx - MicroBlaze Soft Processor Core', http://www.xilinx.com/products/design-tools/microblaze.html
14. 'ARM - Cortex M0 Design Start', http://www.arm.com/products/designstart/index.php
15. 'Oracle - OpenSPARC', http://www.oracle.com/technetwork/systems/opensparc/index.html
16. 'Aeroflex Gaisler - LEON series Softcores', http://www.gaisler.com/
17. Waterman, A., Lee, Y., Patterson, D.A., et al., 'The RISC-V Instruction Set Manual, Volume I: User-Level ISA', 2014
18. 'OpenCores - OpenRISC', http://opencores.org/or1k/Main_Page