

# PULPito: Una plataforma heterogénea ARM/RiscV para el desarrollo de un sistema operativo híbrido

Juan Antonio Andrés Sáez<sup>1</sup> y Javier Garrido Salas<sup>2</sup>

Departamento de Tecnología Electrónica y de las Comunicaciones,  
Escuela Politécnica Superior, Universidad Autónoma de Madrid

*Resumen*— Este artículo presenta una nueva plataforma hardware heterogénea basada en arquitecturas ARM y RiscV, diseñada para el desarrollo de un nuevo tipo de sistema operativo, capaz de ejecutar juegos de instrucciones de diferentes arquitecturas en un mismo binario y usando un mismo espacio de direccionamiento.

Un sistema heterogéneo multinúcleo se caracteriza por la coexistencia de tantos sistemas operativos, como número de arquitecturas presentes en el sistema, siguiendo un esquema de multiprocesamiento asimétrico (AMP). Una característica habitual en dichos sistemas heterogéneos es que cada SO dispone de su propio espacio de direccionamiento realizándose la sincronización entre ellos mediante memoria compartida, señalización por buses compartidos, generación de interrupciones o diferentes sistemas de entrada/salida. Si varios núcleos del sistema heterogéneo pertenecen a una misma arquitectura es posible la existencia de un sistema operativo de multiprocesamiento simétrico (SMP) para dichos núcleos. En el presente trabajo se ha tenido como objetivo el desarrollo un único sistema operativo híbrido, capaz de planificar los procesos y recursos de todos los núcleos presentes en un mismo sistema heterogéneo, con diferentes arquitecturas, y conservando todos los beneficios un multiprocesamiento simétrico, se ha realizado el diseño de una plataforma hardware que ofrece las dos arquitecturas RISC de mayor crecimiento de los últimos años.

*Palabras clave*— Sistema operativo híbrido, Sistema heterogéneo, AMP, RiscV, Zynq, FPGA.

## I. INTRODUCCIÓN

LA evolución arquitectural en los sistemas de computación ha sido constante desde sus orígenes. En la última década diferentes fabricantes de procesadores CISC como Intel y AMD han adaptado sus procesadores a los nuevos escenarios de baja latencia y alto ancho de banda entre CPU y periféricos, y a su vez entre las diferentes CPUs dentro de un mismo sistema. Concretamente la adaptación arquitectural desde los sistemas multinúcleo hasta los sistemas multiprocesador-multinúcleo conforman el paradigma actual [1].

Intel ha adaptado su arquitectura desde su primer bus compartido hasta la reciente arquitectura Quick-Path [2], con el fin de incrementar las prestaciones de sus sistemas multi-procesador. Por su parte AMD ha seguido un camino de cambio arquitectural muy

similar con su arquitectura HyperTransport [3]. De forma paralela fabricantes de procesadores RISC como ARM o MIPS han introducido nuevas arquitecturas optimizadas para procesadores multinúcleo, como es el caso de la arquitectura ARMv8 [4]

Cada una de las arquitecturas CISC/RISC ha tenido su propia evolución en el mercado en base a factores de negocio, de producción y de innovación, pero siempre con segmentos de mercado diferenciados según la arquitectura. Mientras que las arquitecturas CISC han predominado en el segmento de los ordenadores personales, las arquitecturas RISC han tenido mayor presencia en el segmento de los dispositivos móviles. El debate RISC-CISC ha sido constante en los últimos años [5] y la aparición de nuevos procesadores basados en arquitecturas abiertas, como RiscV [6], lo han estimulado aún más.

En la actualidad existe un nuevo debate sobre la convergencia de ambas arquitecturas debido en primer lugar a la necesidad de los fabricantes de arquitecturas CISC de aumentar su presencia en el segmento de los dispositivos móviles y, por otro lado, a la demanda del mercado de aumentar las prestaciones en estos dispositivos, que ya incorporan arquitecturas RISC, para obtener prestaciones similares a los ordenadores personales convencionales [7].

En paralelo a la evolución de las arquitecturas CISC y RISC, el software se ha ido adaptando al hardware en el que debe ser ejecutado. De esta manera sistemas operativos como Windows o Linux han ido incorporando diferentes versiones para cada una de las arquitecturas presentes en el mercado.

Por regla general los sistemas multiprocesador pueden clasificarse en sistemas homogéneos, donde todos los procesadores del sistema tienen una misma arquitectura, o sistemas heterogéneos, donde los distintos procesadores del sistema pueden tener arquitecturas diferentes. Al mismo tiempo, y desde un punto de vista software, un sistema también puede clasificarse según disponga de un multiprocesamiento simétrico (SMP) o asimétrico (AMP).

<sup>1</sup>e-mail: juanantonio.andres@uam.es.

<sup>2</sup>e-mail: javier.garrido@uam.es.

A modo de ejemplo la siguiente figura muestra un sistema heterogéneo formado por dos procesadores ARM Cortex-A7 y un procesador adicional ARM Cortex-M4 [8]:

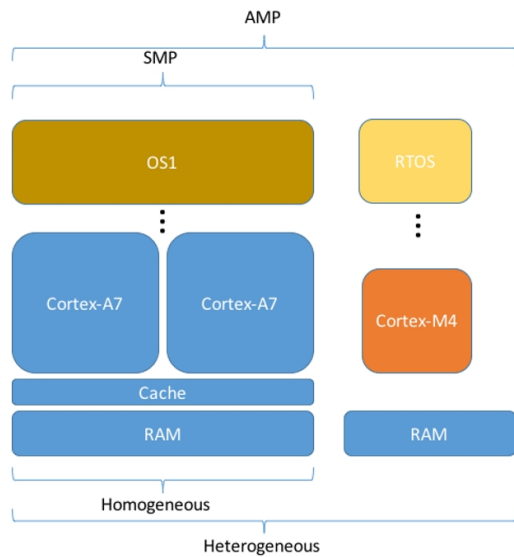


Fig. 1. Sistema heterogéneo TQMa7x

Aunque típicamente un software SMP se apoya en un sistema homogéneo, donde todos los núcleos tienen la misma arquitectura, es posible encontrar modelos de software SMP ejecutándose en sistemas heterogéneos donde los núcleos, a pesar de tener arquitecturas diferentes, comparten un mismo juego de instrucciones (ISA o *Instruction Set Architecture*). De forma análoga es habitual que un sistema heterogéneo tenga asociado un modelo de software AMP.

A diferencia de los modelos de multiprocesamiento SMP o AMP, conceptos ampliamente adoptados en la literatura científica, es posible encontrar denominaciones alternativas para modelos de software que no encajan claramente en los sistemas SMP o AMP.

La aparición de trabajos con referencias a *sistemas operativos híbridos*, capaces de gestionar en paralelo los procesos que requieren tiempo real de los que no requieren tiempo real, introducen un nuevo concepto de modelado software. Algunos trabajos [9] han logrado integrar, en un mismo sistema operativo, un núcleo de Linux con un núcleo de MicroC/OS-II, apoyándose en un procesador Intel PXA270 (con arquitectura ARM). Otros trabajos hacen referencia a S.O. híbridos sobre sistemas homogéneos [10] que también cumplen esta característica.

Más recientemente el concepto *híbrido* ha sido ligado a sistemas multi-procesador que incorporan GPGPUs (*General-Purpose Computing on Graphics Processing Units*) [11] [12]. Estas GPUs permiten acelerar los procesos que requieren mayor capacidad de computación paralela y que permiten a la CPU central ejecutar procesos de menor complejidad.

En general el adjetivo *híbrido* no ha sido normalizado debido al bajo número de publicaciones que hacen referencia a este concepto, tanto en sistemas operativos, como en arquitecturas de sistemas. En este estudio se denomina *sistema operativo híbrido* el sistema que permite soportar dos o más arquitecturas o juegos de instrucciones (ISA), simultáneamente.

Según esta definición un *sistema operativo híbrido* tendría como característica principal la gestión única de todo el espacio de direccionamiento de memoria, de todos los procesos así como de todos los recursos del S.O. respecto a los diferentes procesadores del sistema, independientemente de la arquitectura que tengan. Aunque inicialmente podría parecer que un *sistema operativo híbrido* no puede alcanzar un multi-procesamiento simétrico (SMP), similar al de un sistema homogéneo, técnicamente es posible diseñar un *sistema operativo híbrido* con estas características.

Por ello, el diseño de un *sistema operativo híbrido* ejecutándose sobre un sistema heterogéneo, y con procesadores de diferentes arquitecturas, permitiría obtener diversas ventajas sobre otros modelos de multi-procesamiento AMP:

1. La capacidad para ejecutar desde un mismo sistema operativo los procesos compilados para distintas arquitecturas. Ya no sería necesaria la intercomunicación entre distintos sistemas operativos de modo que el S.O. central podría gestionar dicha sincronización de forma transparente.
2. La disponibilidad de tener un mismo proceso compilado para las diferentes arquitecturas del sistema heterogéneo, siendo el S.O. el encargado de decidir cuál es la arquitectura más adecuada en función de diferentes métricas o parámetros.
3. En general un S.O. de este tipo permite simplificar la comunicación entre procesos de distintas arquitecturas, dado que al ejecutarse sobre un mismo S.O. se pueden utilizar los mismos recursos IPC como si de un sistema SMP se tratase. Esto evitaría el uso de software o librerías de sincronización entre S.O. que tienen una alta dependencia en el sistema heterogéneo de que se trate.
4. Al igual que sucede con los procesos del nivel de aplicación, los controladores de un sistema operativo híbrido también podrían ser compilados para diferentes arquitecturas, eliminando la necesidad de pre-asignar los dispositivos de entrada/salida a procesadores específicos, pudiendo todos los procesadores acceder a todos los dispositivos de forma totalmente transparente.

5. Al no ser necesaria la sincronización entre diferentes S.O., los mecanismos de sincronización se simplifican con una leve optimización en la latencia total del sistema, reduciendo el consumo y aumentando las prestaciones finales.

## II. DISEÑO DE UN SISTEMA OPERATIVO HÍBRIDO

El diseño de un *sistema operativo híbrido* es una tarea compleja con requerimientos y necesidades muy particulares. Aunque no es el objetivo de este trabajo, los requerimientos básicos de un diseño con las características mencionadas son:

- Disponer de una plataforma HW heterogénea, que albergue procesadores de diferentes arquitecturas y que permita un acceso total, por parte de los procesadores, a toda la memoria disponible del sistema, así como a los periféricos y recursos de entrada/salida.
- Disponer de un compilador para cada arquitectura y un enlazador o *linker* multi-arquitectura, capaz de generar un único binario con código máquina de diferentes arquitecturas (ISAs).
- Una gestión inter-procesador de la coherencia de cachés así como un modelo de memoria común entre las diferentes arquitecturas del sistema homogéneo.
- Un sistema inter-procesador de bloqueo o protección de regiones de memoria que impida el acceso concurrente de varios procesadores.

La complejidad en el diseño de un *sistema operativo híbrido* es elevada porque en la actualidad no existen trabajos o herramientas que faciliten cada uno de estos requerimientos. Este trabajo se ha centrado en el primer requerimiento mencionado: Disponer de una plataforma hardware heterogénea, que permita el diseño de dicho S.O. híbrido.

Para diseñar una plataforma heterogénea, con las características básicas mencionadas, es importante conocer qué características deberá tener el *sistema operativo híbrido* que será ejecutado en ella. Qué módulos conformarán el sistema operativo o si estará diseñado a partir de otros S.O. conocidos. Cuáles serán las arquitecturas soportadas: ¿RISC o CISC?, ¿en 32 o en 64 bits?, ¿con qué organización de bits (*endianness*)?, ¿*big* o *little*?. Qué requerimientos de memoria tendrá el S.O.: Tipo, tamaño, velocidad... Qué periféricos de entrada/salida deberá soportar el sistema y si será necesario un soporte hardware para facilitar la depuración del S.O. (y en general de todo el software), etc...

Para la elaboración de este trabajo se han seleccionado dos arquitecturas RISC de gran relevancia:

- Una arquitectura con un elevado volumen de

mercado: *ARM*.

- Una arquitectura con un elevado crecimiento de su ecosistema: *RiscV*.

La coexistencia de ambas arquitecturas en la misma plataforma heterogénea plantea diferentes ventajas:

1. Ambas disponen de procesadores de 32 y 64 bits así como de una misma organización de bits.
2. Ambas disponen de herramientas, compiladores, sistemas operativos comunes como Linux, Zephyr, etc, así como diverso hardware de depuración.

Aunque ARM es un núcleo propietario, su elevada presencia en el mercado facilita la presencia de numerosos procesadores que puedan ser integrados en la plataforma. En el caso de RiscV, dado que se trata de una arquitectura abierta [6], su integración mediante FPGAs también simplifica el diseño de la plataforma heterogénea. A pesar de que RiscV es una nueva arquitectura, recientemente [13] se ha demostrado el interés que numerosas empresas muestran al considerar la arquitectura RiscV como sucesora de ARM en segmentos de mercado de IoT, dispositivos móviles o sistemas de bajo consumo en la nube.

## III. PLATAFORMA HETEROGÉNEA PULPITO

PULPito es una plataforma heterogénea, basada en arquitecturas ARM y RiscV, que ha sido desarrollada a partir de la plataforma PULPino [14] y que a su vez es una versión reducida de la plataforma PULP [15].

Aunque PULPino ofrece una única arquitectura RiscV en su variante SoC, su versión FPGA resulta interesante ya que fue diseñada sobre un procesador Zynq [16]. Los procesadores Zynq ofrecen una arquitectura polivalente que incluye FPGAs de diferentes tamaños, un procesador ARM de doble núcleo Cortex-A9 y diferentes periféricos en un único SoC.

La posibilidad de sintetizar un soft-core RiscV sobre la FPGA del procesador Zynq, ofrece la suficiente flexibilidad para diseñar una plataforma heterogénea que incluya ambas arquitecturas y todos los periféricos necesarios para diseñar un *sistema operativo híbrido*.

Los procesadores Zynq reúnen numerosas ventajas a la hora de diseñar un sistema heterogéneo:

- La existencia de numerosos controladores hardware directamente accesibles desde la lógica programable: DDR, FLASH, Quad-SPI, etc... Esto minimiza el número de controladores a ser implementados y hace que la interconexión con ella sea una gran ventaja.

- Una matriz de alto rendimiento que permite una interconexión múltiple entre los periféricos del punto anterior, las memorias del sistema y la lógica programable mediante buses AXI [18]. Entre las conexiones disponibles cabe destacar la existencia de un puerto maestro de coherencia de caché con la lógica programable, denominado ACP (*Accelerator Coherency Port*). Este puerto permitirá un acceso directo a las cachés de nivel 1 y 2, permitiendo una coherencia de cachés entre el procesador RiscV y los núcleos ARM.
- La posibilidad de crear bucles (*loopbacks*) entre la lógica programable y los periféricos presentes en el procesador sin que las transacciones en ambos sentidos interbloqueen las unas con las otras. Este tipo de características serán explotadas en el diseño de PULPito con el fin de optimizar el rendimiento del *sistema operativo híbrido*.
- La disponibilidad de diferentes capacidades de la lógica programable en función del procesador Zynq seleccionado. Los diferentes dispositivos que ofrece la familia Zynq ofrecen desde 28.000 celdas lógicas en el XC7Z010, hasta 444.000 en el XC7Z100. La posibilidad de sintetizar un procesador RiscV, con todas las extensiones habilitadas, junto a los periféricos mínimos necesarios determinarán cual es el dispositivo Zynq más adecuado.

Además la familia de procesadores Zynq está disponible en numerosas placas, fabricadas por diferentes integradores, por lo que la elección de la placa de desarrollo es un factor determinante para lograr el diseño planteado. Los siguientes criterios fueron aplicados para seleccionar la placa de desarrollo más adecuada:

- Disponer de un dispositivo Zynq con un tamaño de lógica programable lo suficiente grande para albergar los núcleos RiscV con todas sus extensiones.
- Permitir el acceso desde la lógica programable a los diferentes buses y conectores externos que faciliten una correcta depuración del sistema heterogéneo.
- Que sea una placa de desarrollo económica.

Si bien la plataforma original PULPino fue diseñada sobre una placa ZedBoard [17], la placa de desarrollo finalmente seleccionada se denomina Z-Turn [19] del integrador MYiR:

La plataforma PULPito [20] ha sido diseñada para soportar algunos de los periféricos y conectores de la placa de desarrollo ZTurn y ofrecer así todos los elementos necesarios para realizar el diseño de un

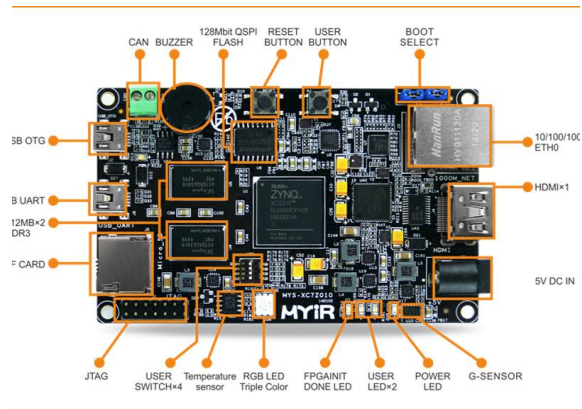


Fig. 2. Placa de desarrollo ZTurn

*sistema operativo híbrido*. Sobre esta base PULPito está formado por los siguientes elementos:

- Código verilog y system-verilog del soft-core RiscV, sintetizable para la lógica programable de la placa de desarrollo ZTurn (con un dispositivo Zynq XC7Z020).
- Herramientas JTAG y depuradores para cada una de las arquitecturas (ARM y RiscV)
- Periféricos y buses necesarios para una correcta integración con los núcleos de la arquitectura ARM que están fuera de la lógica programable, incluyendo un adecuado acceso al puerto ACP que permitirá una coherencia de caché entre ambas arquitecturas.
- Archivos de soporte de la placa ZTurn para la herramienta de síntesis Xilinx Vivado 2017.1 y posteriores versiones.

Dado que la plataforma original [14] fue pensada para diseñar un SoC basado en la arquitectura RiscV únicamente, los bloques externos al procesador (comúnmente denominados '*uncore*'), han sido modificados para permitir la coexistencia de las dos arquitecturas seleccionadas.

Este nuevo rediseño de la plataforma original permitirá una comunicación bidireccional entre los buses principales de ambas arquitecturas, así como el aprovechamiento de los puertos especiales que garantizan una correcta coherencia de memoria (especialmente en la memoria de instrucciones).

Tal y como muestra el diagrama de bloques, la nueva plataforma [20] se apoya tanto en la lógica programable como en los bloques propios del sistema de procesamiento (PS) que ofrece el procesador Zynq.

Los puntos fundamentales del diseño son:

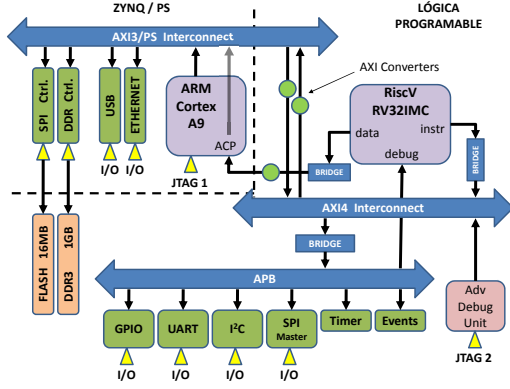


Fig. 3. Diagrama de bloques de PULPito

- La lógica programable (PL) recibe señales de reset y de reloj generadas por el PS de Zynq. Los procesadores ARM operarán a una frecuencia de reloj de 666 Mhz, mientras que el procesador RiscV operará con una frecuencia de 50 Mhz (libre de *glitches* gracias a los generadores de reloj propietarios)
- La comunicación entre la lógica programable y la no-programable se realizará con tres buses AXI4. Como los procesadores ARM disponen de buses AXI3 (en lugar de AXI4) es necesario instanciar tres convertidores de transacciones AXI (marcados por un círculo verde en el diagrama de bloques). En el procesador RiscV se han reutilizado adaptadores AXI especiales de la plataforma original [14] que permitan la interconexión con el PS.
- Cada procesador dispone de su propio interfaz JTAG (IEEE 1149.1) para depuración. En el caso de la arquitectura ARM el interfaz JTAG ya está correctamente enrutado en la placa ZTurn a un conector USB y Xilinx provee el software y todas las herramientas necesarias. En el caso de la arquitectura RiscV el interfaz JTAG será enrutado a pines disponibles para la lógica programable. Más adelante se describirá el adaptador JTAG utilizado, así como las herramientas software necesarias para poder depurar en la arquitectura RiscV.
- Las matrices de interconexiones de buses AXI (tanto en PS como en PL) estarán configuradas con los mismos direccionamientos de memoria para maestros y esclavos. El objetivo es que el software del *sistema operativo híbrido* acceda a los diferentes bloques independientemente de si un módulo de dicho software ha sido compilado para una arquitectura u otra.
- Dado que los procesadores seleccionados no soportan instrucciones transaccionales (o *atómicas*), los buses quedarán liberados entre accesos consecutivos (una escritura y una lec-

tura por ejemplo). Este hecho permitiría, desde otra arquitectura, escrituras o lecturas sucias en la misma posición de memoria. El diseño actual no contempla un sistema señalización, o acceso exclusivo a una región de memoria, que evite un acceso concurrente. Los mecanismos de sincronización entre ambas arquitecturas quedarán aplazados a un futuro estudio cuando una primera versión *sistema operativo híbrido* esté disponible. En este sentido la definición de un *modelo de consistencia de memoria*, común a ambas arquitecturas, será un objetivo prioritario en el futuro.

Según los puntos mencionados, junto al diagrama de bloques presentado, la configuración de la lógica no-programable queda representada en la siguiente figura:

En el repositorio principal de la plataforma PULPito [20] es de código abierto sobre licencia Apache y en él se pueden encontrar todos los bloques y fuentes necesarios para poder sintetizar una versión operativa con la que empezar el diseño del primer *sistema operativo híbrido* basado en arquitecturas ARM y RiscV.

#### IV. DEPURACIÓN

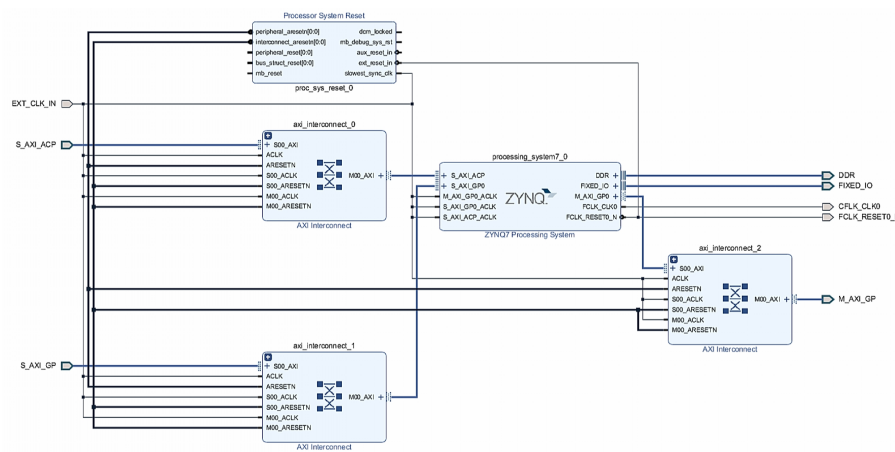
Uno de los elementos esenciales para el diseño de un *sistema operativo híbrido* es la disponibilidad de un sistema de depuración efectivo, capaz de controlar la ejecución de los procesadores de cada una de las arquitecturas, al mismo tiempo que ofrezca un acceso total a todos los registros de cada banco.

Tal y como se ha visto en la sección anterior, el diseño de PULPito contempla la presencia de diferentes interfaces JTAG (IEEE 1149.1) para cada una de las arquitecturas. Aunque el procesador Zynq ofrece la posibilidad de poner en cascada el puerto TAP del PS con un segundo puerto de usuario TAP en la lógica programable, este hecho obligaría a que las herramientas usasen un mismo adaptador JTAG. Si las herramientas de depuración software necesitasen un acceso en exclusiva a dicho adaptador esto representaría un serio problema. Para evitarlo, la plataforma PULPito dispone de dos puertos JTAG independientes, donde será necesario conectar dos adaptadores JTAG, cada uno de los cuales será controlador por un depurador independiente.

Los adaptadores JTAG USB usados en la plataforma PULPito son:

1. Adaptador JTAG de acceso al DAP (*Debug Access Port*) de la arquitectura ARM y también al TAP de Xilinx (que permitirá cargar el flujo de bits en la FPGA del Zynq) : JTAG-HS3 [21].





2. Adaptador JTAG de acceso al TAP de la arquitectura RiscV : CJMCU-232H [22] basado en un chipset de FTDI de bajo coste. El software original *gdbserver* (denominado *debug bridge* en la plataforma PULP), ha sido modificado para soportar este nuevo adaptador JTAG USB.

## V. RESULTADOS Y CONCLUSIONES

Los resultados de la síntesis de la plataforma PULPito muestran una ocupación del 28.7% de Slices, un 17.3% de LUTs y un 7.14% de registros (todos flip-flops).

Una vez cargado el flujo de bits en la lógica programable del procesador Zynq, se ha verificado un correcto acceso a los procesadores por parte de los depuradores de ambas arquitecturas mediante los adaptadores JTAG que ofrece la plataforma.

Los diferentes ejemplos, compilados a partir de las herramientas de software libre que ofrecen ambas arquitecturas, y disponibles en el repositorio principal de la plataforma PULPito, muestran cómo ambos procesadores disponen del mismo direccionamiento y de la misma capacidad de acceso a los bloques hardware de cada arquitectura.

Los resultados obtenidos demuestran que la plataforma heterogénea PULPito es adecuada para el diseño de un *sistema operativo híbrido*, gracias a un direccionamiento general por ambas arquitecturas, un sistema de coherencia de memoria unificado y un acceso global a los recursos hardware por parte del software que corre en cada una de ellas.

## REFERENCIAS

## V. RESULTADOS Y CONCLUSIONES

Los resultados de la síntesis de la plataforma PULPito muestran una ocupación del 28.7% de Slices, un 17.3% de LUTs y un 7.14% de registros (todos flip-flops).

Una vez cargado el flujo de bits en la lógica programable del procesador Zynq, se ha verificado un correcto acceso a los procesadores por parte de los depuradores de ambas arquitecturas mediante los adaptadores JTAG que ofrece la plataforma.

Los diferentes ejemplos, compilados a partir de las herramientas de software libre que ofrecen ambas arquitecturas, y disponibles en el repositorio principal de la plataforma PULPito, muestran cómo ambos procesadores disponen del mismo direccionamiento y de la misma capacidad de acceso a los bloques hardware de cada arquitectura.

Los resultados obtenidos demuestran que la plataforma heterogénea PULPito es adecuada para el diseño de un *sistema operativo híbrido*, gracias a un direccionamiento general por ambas arquitecturas, un sistema de coherencia de memoria unificado y un acceso global a los recursos hardware por parte del software que corre en cada una de ellas.

## REFERENCIAS

- [1] John Mellor-Crummey, *From Multicore to Multisocket*, Comp 522; Lecture 6, Department of Computer Science, Rice University, 2013.
- [2] N.A. Kurd, S. Bhamidipati, C. Mozak, J.L. Miller, P. Mosalikanti, T.M. Wilson, A.M. El-Husseini, M. Neidengard, R.E. Aly, M. Nemani, M. Chowdhury, R. Kumar, *A Family of 32 nm IA Processors*, IEEE Journal of Solid-State Circuits, 2011.
- [3] P. Conway, B. Hughes, *The AMD Opteron CMP Ninth Bridge Architecture: Now and in the future*, Hot Chips, 2006.
- [4] *ARM Discloses Technical Details Of The Next Version Of The ARM Architecture*, ARM Holdings, 2011.
- [5] Ciji Isen, Lizy K. John, Eugene John, *A Tale of Two Processors: Revisiting the RISC-CISC Debate*, Proceedings of the 2009 SPEC Benchmark Workshop on Computer Performance Evaluation and Benchmarking, 2009.
- [6] A. Waterman, Y. Lee, D. Patterson, K. Asanovic, *The RISC-V instruction set manual. volume I: User-level ISA, version 2.0*, EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-54, 2014.
- [7] E. Blem, J. Menon, K. Sankaralingam, *Power Struggles: Revisiting the RISC vs. CISC Debate on Contemporary ARM and x86 Architectures*, 19th IEEE International Symposium on High Performance Computer Architecture (HPCA2013), 2013.
- [8] URL: <https://www.tq-embedded.com/en/Pim/Product-collection/Embedded-module-TQMa7x>, TQ Systems GmbH, 2016
- [9] M. Liu, ZL. Shao, M. Wang, HX. Wei, TM. Wang, *Implementing hybrid operating systems with two-level hardware interrupts*, 28th IEEE International Real-Time Systems Symposium (RTSS), 2007.
- [10] H. Takada, S.I. Iiyama, T. Kindaichi, S. Hachiya, . (2002). *Linux on ITRON: a hybrid operating system architecture for embedded systems*, Applications and the Internet (SAINT) Workshops IEEE, 2002.
- [11] G. Alfonsi, Stefania A. Ciliberti, M. Mancini, *Performances of Navier-Stokes Solver on a Hybrid CPU/GPU Computing System*, 11th International Conference on Parallel Computing Technologies. Volumen: 6873, 2011.
- [12] C.M. Ng, *Novel Hybrid GPU-CPU Implementation of Parallelized Monte Carlo Parametric Expectation Maximization Estimation Method for Population Pharmacokinetic Data Analysis*, American Association of Pharmaceutical Scientists Journal, Volumen: 15, Número: 4, 2013.
- [13] URL: <https://riscv.org/2018/05/risc-v-workshop-in-barcelona-proceedings>, RISC-V Workshop in Barcelona Proceedings, 2018.
- [14] A. Traber, F. Zaruba, S. Stucki, A. Pullini, G. Haugou, E. Flamand, L. Benini, *PULPino: A small single-core RISC-V SoC*, 3rd RISC-V Workshop, 2016.
- [15] D. Rossi, F. Conti, A. Marongiu, A. Pullini, I. Loi, M. Gautschi, L. Benini, *PULP: A parallel ultra low power platform for next generation IoT applications*, Hot Chips 27 Symposium (HCS) IEEE, 2015.
- [16] M. Santarini, *Zynq-7000 EPP sets stage for new era of innovations*, Xcell Journal, 2011.
- [17] *Digilent's ZedBoard Zynq*, F.P.G.A. Dev. board documentation, Google Scholar, 2014.
- [18] URL: <https://developer.arm.com/products/architecture/amba-protocol/amba-3>, ARM
- [19] URL: <http://www.myrirtech.com/list.asp?id=502>, MYiR
- [20] URL: <https://github.com/HCTLab/pulpito>, GitHub.
- [21] URL: <https://store.digilentinc.com/jtag-hs3-cable>,
- [22] URL: <http://www.ftdichip.com/Products/ICs/FT232H>, FTDI.