Implementación de un procesador académico simple así como de un entorno de programación y depuración para el mismo

J. Ruiz, D. Guerrero, I. Gomez y J. Viejo
Departamento de Tecnología Electrónica
Universidad de Sevilla
Sevilla, España
jonathanruizpaez@gmail.com, guerre@dte.us.es , igomez@us.es, julian@dte.us.es

Resumen—**En este trabajo se desarrolla un procesador académico para su uso en la asignatura Estructura de Computadores de primer curso de las nuevas titulaciones de Grado en Ingeniería Informática. En las prácticas de la asignatura se aplicarán los conceptos de sistemas digitales que ya poseen los alumnos al diseño e implementación de este procesador con el objetivo de que interactúen con una instancia real del mismo desde distintos puntos de vista: modificándola para aumentar su funcionalidad, programándola para comprobar su funcionamiento y analizando su estado interno a medida que ejecuta instrucciones. La posibilidad de que el alumno traslade el diseño teórico a una implementación funcional es fundamental para incrementar su motivación en el aprendizaje.**

Temática-entrenador; laboratorio; sistemas digitales; procesador; docencia

I. INTRODUCCIÓN

A la hora de preparar el proyecto docente de una asignatura, siempre debe comenzarse por establecer claramente cual es el objetivo básico que se persigue. Una vez que está claramente establecido se pasa a construir el programa temático más apropiado para que, tras llevar a cabo el proceso de enseñanza-aprendizaje, se hayan sentado las bases para que el alumno haya alcanzado el nivel de conocimiento previsto en el mismo. El programa de la asignatura consta de un conjunto de temas estructurados de tal forma que se realice la introducción de los conocimientos de forma ordenada y progresiva. En el proceso habitual de enseñanza-aprendizaje de las asignaturas técnicas, esta introducción se realiza en clases teóricas. Sin embargo, está claramente reconocido que para lograr el éxito docente es necesario completar los conocimientos que se adquieren en estas clases teóricas con un conjunto adecuado de prácticas de laboratorio cuya misión fundamental es aclarar y reforzar los conceptos básicos introducidos en teoría, así como potenciar las habilidades prácticas enfrentando al alumno con sistemas reales que se deben manipular. En las materias técnicas electrónicas este aspecto adquiere mucha más relevancia pues los contenidos temáticos siempre hacen referencia al modo de operación de los sistemas electrónicos reales. Es por ello que debe realizarse una elección cuidadosa y adecuada para las prácticas de laboratorio en las asignaturas del campo de la electrónica.

En este trabajo vamos a presentar un procesador académico sencillo cuya descripción constituye un tema en la docencia de la asignatura de Estructura de Computadores de primer curso de las nuevas titulaciones de Grado en Ingeniería Informática de la Universidad de Sevilla. Este tema es previo a otros centrados en la explicación de procesadores comerciales. La extrema simplicidad del procesador propuesto ha permitido exponer en su totalidad su funcionalidad en las clases de teoría así como explicar de forma exhaustiva el diseño de una implementación tanto de forma esquemática como en lenguajes de descripción de hardware. También ha facilitado la implementación en hardware de un sistema completo basado en dicho procesador sobre placas de prototipado de FPGA así como software de ensamblado-desensamblado y herramientas para la programación y depuración de dicho sistema. Esto ha hecho posible la realización de prácticas de laboratorio en las que el alumno interactúa con una implementación real del procesador visto en teoría desde distintos puntos de vista: modificándola para aumentar su funcionalidad, programándola y analizando su estado interno a medida que ejecuta instrucciones, siendo ventajosa esta actividad frente a otras basadas sólo en simulación [1]. De esta manera se trabaja tanto a nivel de transferencia entre registros (RT), como a nivel de manejo de instrucciones del procesador (ISP) constituyendo el sistema diseñado una buena herramienta de adquisición de conceptos importantes que luego deben ser utilizados en el estudio de sistemas comerciales. La posibilidad de que el alumno traslade el diseño teórico a una implementación funcional supone, además, añadir un factor de especial motivación. En [2], [3] y [4] se desarrollan trabajos sobre las ventajas que conlleva la introducción del aprendizaje activo en materias relacionas con los sistemas embebidos y los microcontroladores.

La organización del artículo es como sigue: en la sección siguiente vamos a presentar los contenidos de la asignatura Estructura de Computadores. Luego describiremos el procesador académico mencionado. Posteriormente vamos a presentar el planteamiento general y objetivos de la práctica que proponemos. En la sección V mostramos el montaje que hemos realizado para preparar la práctica. La sección VI está dedicada a mostrar la guía de desarrollo experimental que se le

aporta al alumno para que trabaje en el laboratorio. Por último destacamos las conclusiones más importantes del trabajo.

II. Organización de los Contenidos de la Asignatura

Para entender los aspectos positivos que supone la introducción de esta nueva práctica en la asignatura de Estructura de Computadores es necesario que hagamos una introducción a su contenido y objetivos. Estructura de Computadores es una asignatura obligatoria en las titulaciones de Grado de Ingeniería Informática de la Universidad de Sevilla que se imparte en el segundo cuatrimestre del primer curso. Tiene como objetivo básico mostrar al alumno el modo de operación interno de un computador. Con este objetivo podemos destacar los siguientes bloques temáticos en la asignatura:

- Sistemas digitales.
- Diseño de un computador simple.
- Procesadores comerciales en explotación.

Respecto a los dos últimos bloques mencionados fue necesario elegir tanto el computador simple como el procesador comercial a exponer en clases de teoría y laboratorio. Como sistema comercial se optó por un microcontrolador ATmega328P [5] con la arquitectura AVR de Atmel que ya había sido usado en prácticas de otras asignaturas de electrónica de la titulación, de forma que los docentes ya estaban familiarizados con él. Además, hay mucha información didáctica y placas de desarrollo educativas de hardware muy asequibles basadas en dicho microcontrolador [6]. En cuanto al computador simple cuyo diseño interno debe exponerse en teoría, es deseable que éste sea parecido a los sistemas comerciales actualmente en explotación, especialmente a aquel que se verá en un tema posterior. Sin embargo, no debemos olvidar que estamos en una asignatura de introducción, por lo que no debe plantearse el diseño de un sistema digital demasiado complejo. Por ello se optó por diseñar un procesador específicamente para la docencia de este bloque lo suficientemente simple como para que se pudiera exponer su estructura de la forma más exhaustiva posible y que el alumnado pudiera llevar a cabo su análisis, implementación y modificación con el material de laboratorio propio de esta asignatura. A este procesador se le denominó PAS (Procesador Académico Simple).

III. Descripción del Procesador Académico Simple

A la hora de modelar el PAS se tuvieron en mente los siguientes objetivos:

- Simplificar su modelo de usuario para que sea fácilmente entendible por el alumnado.
- Procurar que el modelo de usuario sea parecido al de sistemas comerciales actuales (especialmente al procesador que se usará en el tema posterior) siempre que esto no aumente demasiado su complejidad.
- Dar una implementación de referencia de la unidad de procesado de datos que sea lo más sencilla posible.
- Priorizar la facilidad de implementación de la unidad de control sobre la simplicidad de la unidad de procesado de datos que se dará como referencia (la unidad de control debe poder ser implementable por el alumnado).

A. Modelo de usuario

Tras cierto número de revisiones de la arquitectura por parte de los docentes se obtuvo el procesador que ahora presentamos. Para simplificar su unidad de datos y de control el procesador tiene las siguientes características:

- Arquitectura Harvard.
- Reducido número de registros visibles y todos del mismo tamaño.
- Todas las instrucciones ocupan una palabra de memoria de código.
- Reducido número de instrucciones y de formatos de instrucción que además comparten campos en las mismas posiciones.
- ISP tipo load/store con un único modo de direccionamiento para operandos en memoria (con la excepción de las instrucciones de manejo de pila)
- Entrada/Salida mapeada en memoria.
- Ausencia de interrupciones, modos de operación especiales, MMU o caches.

Para programar el PAS el usuario necesita conocer el formato y semántica de sus instrucciones (ISP) así como el conjunto de registros a los que hacen referencia las mismas, es decir, el conjunto de registros visibles. También necesita un modelo del sistema de memoria a la que accederá el programa. La semántica de las instrucciones es básicamente la misma de sus homólogas en la arquitectura AVR8 [5]. El resto del modelo se detalla a continuación.

1) Memoria y registros visibles

El PAS cuenta con un espacio de memoria de datos de 256x8 bits y un espacio de memoria de código de 256x16 bits. Sus registros visibles son de 8 bits, disponiendo de ocho de propósito general y tres de propósito específico. Los de propósito general se etiquetan R0, R1, R2, R3, R4, R5, R6 y R7. Los de propósito específico son los siguientes:

- PC o contador de programa: Contiene la dirección de la próxima instrucción que se ejecutará. Se inicializa a cero y se va incrementando a medida que se ejecutan las instrucciones.
- SR o registro de estado: Sus bits útiles se etiquetan Z (cero), V (desbordamiento), N (negativo) y C (carry). Indican, respectivamente, si la última instrucción aritmética/lógica generó un resultado con todos los bits a cero, un resultado no

representable en complemento a 2, un resultado que es negativo al interpretarlo en complemento a dos o si provocó carry/borrow.

- SP o puntero de pila: Sirve para implementar la estructura de pila en memoria. Codifica la dirección de la primera posición libre, siendo su valor inicial $FF. El puntero de pila se va decrementando a medida que se apilan datos y se incrementa al desapilarlos. Estos incrementos y decrementos se hacen de forma automatizada al realizar llamadas y retornos de subrutina.

2) Juego de instrucciones

Se pretende que el juego de instrucciones ensamblador del PAS sea básicamente un subconjunto muy reducido del de la arquitectura AVR. Esto no implica que el formato del código máquina en ambos sistemas sea similar. Por el contrario, el formato de instrucciones del PAS es muchísimo más simple que el del AVR. El código de operación del PAS tiene una longitud fija de 5 bits, lo que permite disponer un máximo de 32 códigos de operación. Tal y como puede verse en la tabla I, hay únicamente tres formatos de instrucción que además comparten campos en las mismas posiciones. Los códigos de operación se han asignado de forma que se simplifica la decodificación. Además se han usado básicamente los mismos nemónicos que en el ensamblador de AVR para facilitar su introducción posterior.

B. Implementación propuesta a los alumnos

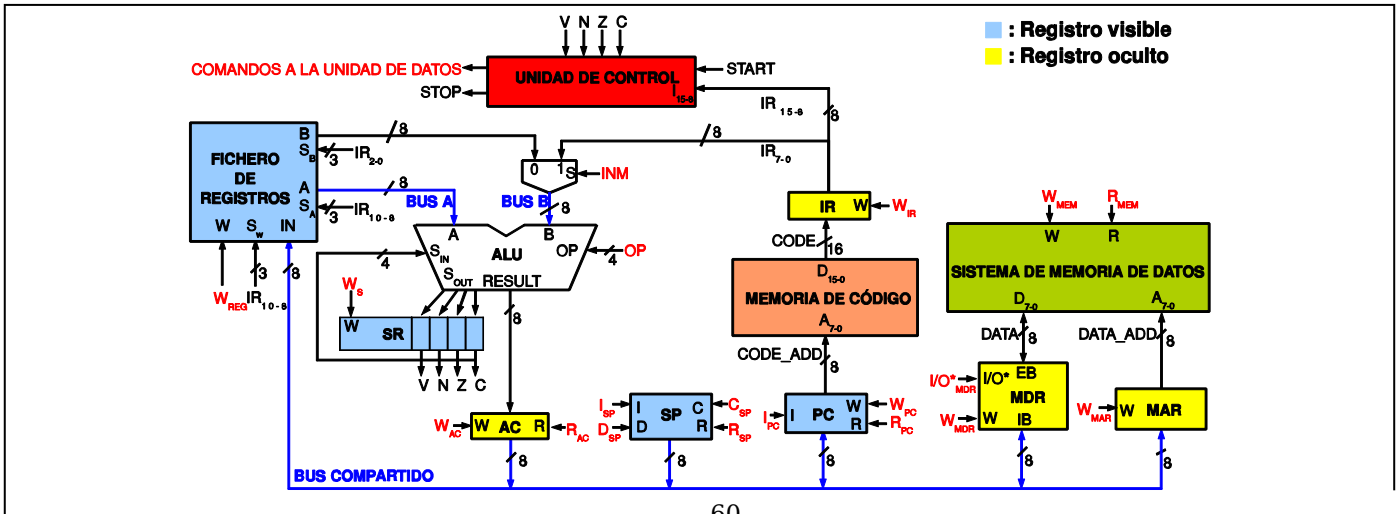
A la hora de implementar la arquitectura de un procesador es necesario definir su estructura externa e interna. La unidad de procesado propuesta para el PAS se ilustra en la figura 1. Los registros visibles de propósito general se han implementado con una pequeña RAM síncrona de 8x8 (fichero de registros) de dos puertos de lectura y uno de escritura. Para realizar las operaciones aritmético/lógicas se usa una ALU de 8 bits. La mayor parte de los registros se interconectan por un mismo bus de datos común. Además de estos recursos se ha añadido a la unidad de procesado cuatro registros ocultos: el registro de 16 bits IR se encarga de almacenar el código de la instrucción en ejecución; el registro de 8 bits MAR se encarga de direccionar la memoria de datos; el registro de 8 bits MDR sirve de interfaz entre el bus de datos del sistema de memoria y el bus interno de datos del procesador; el registro AC almacena temporalmente la salida de la ALU. Respecto a la unidad de

TABLA I. FORMATOS DE INSTRUCCIÓN DEL PAS

formato	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A instrucción con operando registro	código de operación					registro destino (fuente en ST)			-	-	-	-	-	registro fuente (registro base en ST/LD)		
B instrucción con operando memoria o inmediato									dato inmediato / dirección del dato							
C instrucción de salto						condición de salto			dirección de salto							

control, ésta se interconecta mediante las siguientes señales:

- W_{MEM} (salida): Indica al sistema de memoria que desea realizarse un acceso en escritura.
- R_{MEM} (salida): Indica al sistema de memoria que desea realizarse un acceso en lectura.
- STOP (salida): Indica al usuario que el sistema se ha detenido.
- W_{AC}, W_{REG}, W_{PC}, W_{MAR} (salidas): Cuando están activas hacen que el dato en el bus común interno se escriba, respectivamente, sobre AC, sobre el fichero de registros, sobre PC o sobre MAR.
- I/O^*_{MDR} (salida): Cuando vale 1 el contenido de MDR se pone sobre el bus interno común. Cuando vale 0 el contenido de MDR se pone sobre el bus de datos externo.
- W_{MDR} (salida): Cuando está activa MDR se escribe con el dato en el bus interno común (si I/O^*_{MDR} vale 1) o con el dato del bus externo (si I/O^*_{MDR} vale 0).
- W_{IR} (salida): Cuando está activa hace que IR se escriba con la salida de la memoria de código.
- W_{SR} (salida): Al activarse hace que SR se escriba con las salidas de overflow, carry, negative y zero de la ALU.
- I_{SP}, I_{PC} (salidas): Cuando están activas hacen que se incrementen SP y PC respectivamente.
- D_{SP}, C_{SP} (salidas): Cuando están activas hacen que SP se decremente o se ponga a 0 respectivamente.



- R_{AC}, R_{PC}, R_{SP} (salidas): Activas hacen que pongan su contenido en el bus común AC, PC, SP o MDR respectivamente.
- INM (salida): Si vale 1, los bits 7-0 de IR (dato inmediato) se ponen sobre la entrada de operando B de la ALU. En caso contrario se pone en B el contenido del registro de propósito general indicado por los bits 2-0 de IR.
- OP_{3-0} (salidas): Indican a la ALU que operación debe realizar.
- I_{15-8} (entrada): Indican el código de operación de la instrucción así como la condición de salto.
- START (entrada): Señal de comienzo de ejecución.

Respecto a la estructura externa, los siguientes buses conectan el PAS con el exterior:

- CODE_ADD (salida): Bus de 8 líneas para direccionar una memoria de código de lectura incondicional.
- DATA_ADD (salida): Bus de 8 líneas para direccionar una memoria de datos.
- CODE (entrada): 16 líneas conectadas a la salida de la memoria de código.
- DATA (entrada/salida): 8 líneas conectadas a la entrada/salida de la memoria de datos.

IV. Planteamiento y Objetivos de la Propuesta de Práctica de Laboratorio

Para cubrir los distintos bloques temáticos desde la perspectiva de laboratorio se desarrollan una serie de prácticas de carácter obligatorio. En esta práctica concreta se implementa y programa el PAS presentado en clases de teoría. La ejecución de esta práctica requiere manejar gran parte de los conceptos básicos que se introducen a lo largo de la asignatura. Concretamente el alumno debe:

- conocer cómo opera un sistema digital estructurado en Unidad de Datos y Unidad de Control.
- desarrollar las habilidades de diseño e implementación.
- implementar y comprobar físicamente el sistema diseñado tanto a nivel de micro-operación como a nivel de macro-operación.

Esto cubre gran parte de los contenidos temáticos de la asignatura. Una vez concretado el objetivo de la práctica hay que detallar su contenido. Al igual que en las otras prácticas el enunciado de la misma se da a conocer por escrito a los alumnos antes de su realización. Los enunciados incluyen un apartado con cuestiones que los alumnos deben responder en un estudio teórico antes de la realización de la sesión de laboratorio, es decir, antes de la parte experimental. Con esto se pretende que el alumno estudie los conceptos tratados en la práctica antes de su realización para que pueda realizarla de forma provechosa. En la parte teórica de esta práctica se pide al alumno que realice programas sencillos en ensamblador para el juego de instrucciones original del PAS, modifique su unidad de control para añadirle nuevas instrucciones y reescriba los programas anteriores usando las nuevas instrucciones. Para ello deberá modificar el archivo Verilog *UnidadDeControl.v* que se le proporciona. En la parte experimental el alumno deberá probar sus modificaciones a la unidad de control ejecutando los programas que ha escrito.

V. Montaje de la Práctica

A. *Hardware*

La primera tarea a realizar fue la elección de la plataforma de implementación del hardware. Elegimos la placa de prototipado de FPGA Basys2 [7] mostrada en la figura 2 que ya se había usado en prácticas previas por lo que alumnos y docentes estaban familiarizados con ella. La placa incluye una FPGA Spartan-3E de Xilinx [8]. Un sistema completo basado en el PAS fue implementado en Verilog, que es el lenguaje de descripción de hardware que se enseña en las clases teóricas. El sistema implementado incluye los siguientes componentes:

- el procesador.
- dispositivos de entrada-salida mapeados en memoria.
- la memoria de código (instrucciones o programa).
- la memoria de datos.
- una unidad de depuración.

Hay que hacer notar que la plataforma utilizada no permite implementar la unidad de procesado de datos exactamente como se describe en clases de teoría. En particular, la FPGA utilizada no permite buses con capacidad de alta impedancia. No obstante esto es transparente para el alumno ya que todas las líneas de interfaz con la unidad de procesado de datos se mantienen tal y como son expuestas en las clases de teoría y la tarea a realizar en la práctica se centra en el diseño de la unidad de control. Respecto a los dispositivos de entrada-salida, un total de 8 de los disponibles en la placa de prototipado se mapearon en el espacio de direccionamiento de la memoria de datos del PAS. Las memorias externas al procesador son básicamente como las presentadas en teoría, salvo que disponen de puertos adicionales para poder ser accedidas por la unidad de depuración. Dicha unidad es un componente que se comunica con el PC a través del puerto serie y permite a éste leer y escribir la memoria de código y datos. Además, la unidad de depuración hace posible depurar los programas descargados mediante:

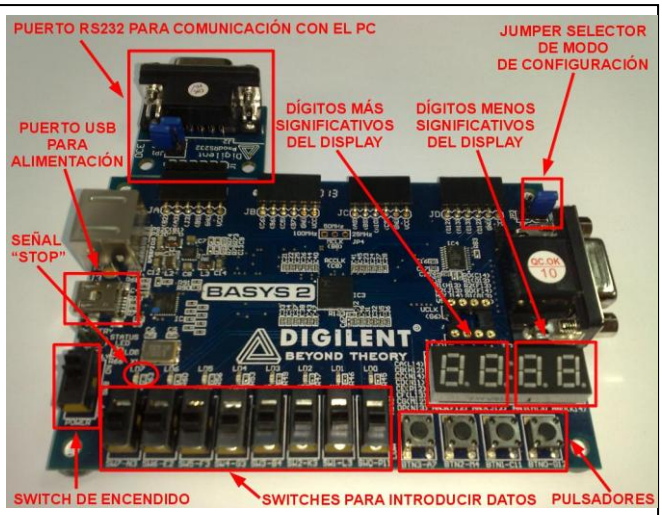


Figura 2. Placa de prototipado empleada en la práctica

- Ejecución de ciclos de reloj individuales.
- Ejecución de instrucciones individuales.
- Inspección del contenido de la memoria y los registros.
- Inspección del estado de las líneas de control.
- Habilitación/inhibición del reloj del procesador.

B. Software

Además de un ensamblador y un desensamblador para el PAS, fue necesario desarrollar un programa que permitiera interaccionar con la unidad de depuración desde los ordenadores del laboratorio. Su interfaz gráfica se muestra en la figura 3. La parte superior de la ventana de la interfaz contiene una serie de controles para interactuar con la unidad de depuración que se describen a continuación:

- Campo para la ruta del puerto serie: Aparece relleno por defecto con la ruta correspondiente al puerto serie de los ordenadores del laboratorio, por lo que en principio los alumnos no deben cambiarla.
- Campo de fichero de código binario (BIN): este campo debe contener la ruta del fichero binario cuyo contenido se escribirá en la memoria de código cuando se inicialice el PAS. Al rellenarse se muestra en la parte superior derecha de la ventana el contenido de la memoria desensamblado. Normalmente no hay que rellenar este campo ya que se suele usar un fichero en formato ensamblador para especificar el programa que deberá escribirse en la memoria de código.
- Campo de fichero de código ensamblador (ASM): este campo debe contener la ruta de un fichero de código ensamblador correspondiente al programa que será escrito en la memoria de código. Al rellenar este campo la herramienta ensamblará el fichero informando de los posibles errores y escribirá la ruta del fichero binario resultante en el campo comentado anteriormente. Rellenar este campo no es obligatorio, pero es la forma habitual de trabajar con la herramienta.
- Campo de fichero de datos: si deseamos inicializar la memoria de datos con el contenido de un fichero binario, podemos indicar su ruta en este campo.
- Botón de establecimiento de conexión e inicialización: al pulsarlo el software establece una conexión a través del puerto RS232 con la unidad de depuración, ordena inicializar el PAS y escribe las memorias de código y datos con el contenido de los ficheros especificados. Únicamente tras esto podremos interactuar con la unidad de depuración.
- Botón *Mostrar Unidad De Datos*: al pulsar este botón aparecerá una ventana con un dibujo esquemático de la unidad de datos del PAS. Durante las ejecuciones ciclo a ciclo se resaltarán en esta ventana las señales que se vayan activando.
- Botón *Habilitar Reloj/Deshabilitar Reloj*: Si el reloj del PAS está detenido, este control permite ponerlo en marcha para que se ejecute el programa de forma autónoma hasta encontrar una instrucción de STOP. Si el reloj está ya en marcha permite detenerlo para inspeccionar el estado del procesador y la memoria.
- Botón *Activar Start*: Le indica a la unidad de depuración que genere un pulso alto en la señal de inicio del PAS.
- Botón *Ejecutar Un Ciclo:* Le indica a la unidad de depuración que habilite el reloj durante un solo ciclo.
- Botón *Ejecutar Una Instrucción*: Le indica a la unidad de depuración que habilite el reloj hasta que se ejecute la instrucción en curso o hasta que el procesador se encuentre en el estado de espera al que se llega tras ejecutar la instrucción STOP.

Cuando el reloj del PAS está detenido se muestra el estado del procesador incluyendo el contenido de las memorias de código y datos, el de los registros, el estado de las líneas de control y los valores de los buses de datos conectados a la ALU. Además, la instrucción en curso aparece resaltada. El contenido de las posiciones memoria de datos puede ser editado manualmente. También es posible guardar el contenido de la memoria de datos en un fichero binario pulsando sobre el botón *Guardar Memoria Datos*.

VI. DESARROLLO DE LA PRÁCTICA EN EL LABORATORIO

Como ya hemos mencionado, la tarea del alumno en laboratorio consiste en probar sus modificaciones al PAS ejecutando los programas que ha escrito en el estudio teórico. Con estos objetivos se le facilita un conjunto de ficheros que permiten implementar el procesador PAS sobre la placa de

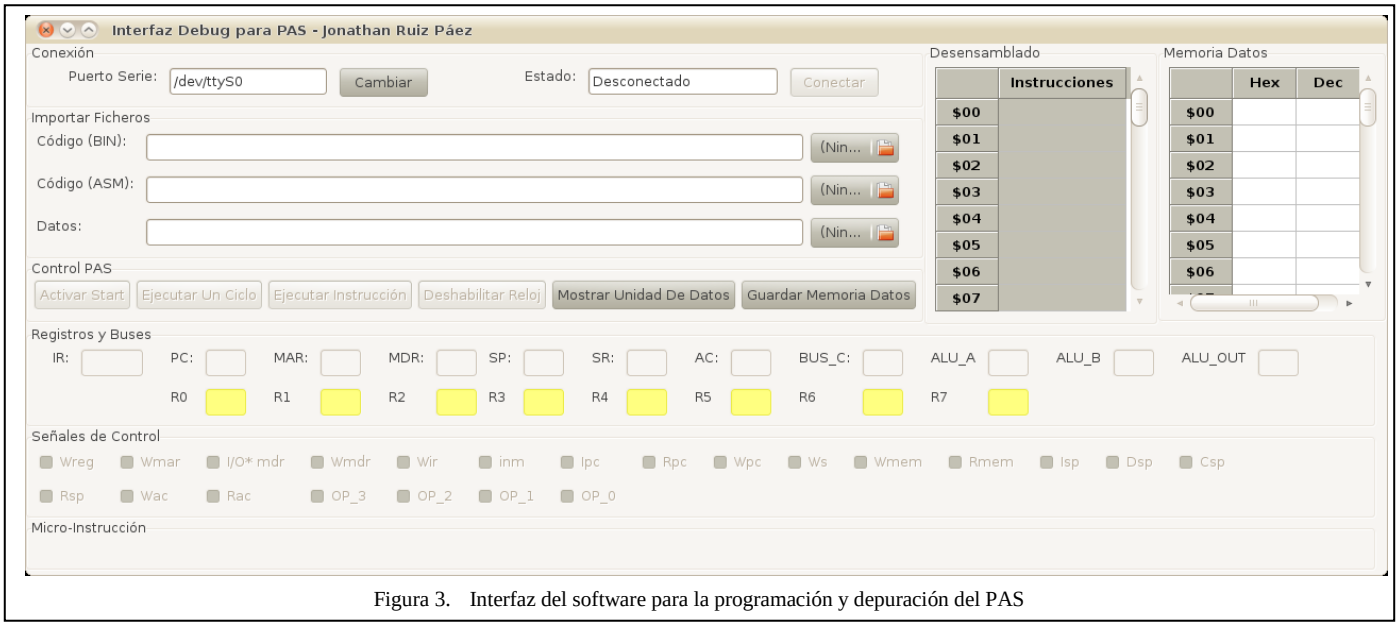


Figura 3. Interfaz del software para la programación y depuración del PAS

prototipado así como una guía detallada de los pasos que debe seguir. Estos pasos se agrupan en dos partes que se describen a continuación:

A. Uso de la arquitectura original

En esta parte se indica al alumno que configure la placa para que en el arranque la FPGA se configure con el *bitstream* contenido en la memoria no volátil de la misma. Dicho *bitstream* se corresponde con la implementación original del PAS. Tras conectar la placa al puerto RS232 del PC y alimentarla el alumno debe cargar en la memoria de código del PAS los programas que escribiese en la primera parte del estudio teórico y comprobar si funcionan tal y como esperaba.

B. Uso de la arquitectura modificada

En esta parte el alumno debe implementar y probar su modificación de la arquitectura PAS. Para ello debe usar el entorno de desarrollo sobre FPGA de Xilinx ISE [9] con el que ya se familiarizó en sesiones de laboratorio previas. Tras incluir al proyecto su versión modificada del fichero *UnidadDeControl.v* e implementar el sistema en la FPGA de la placa de desarrollo debe cargar en la memoria de código del procesador los programas que usan las nuevas instrucciones y comprobar si funcionan tal y como esperaba.

VII. CONCLUSIONES

En este trabajo se ha desarrollado un procesador académico para su uso en la asignatura Estructura de Computadores de primer curso de las nuevas titulaciones de Grado en Ingeniería. Esta asignatura tiene como objetivo básico presentar al alumno el modelo de operación básico de un sistema basado en microprocesador, de modo que esta contribución permite cubrir gran parte del contenido temático. Para complementar la exposición teórica de este procesador se ha desarrollado una práctica de laboratorio en la que el alumno implementa, programa y modifica el diseño del mismo siendo capaz de comprobar su funcionalidad y analizar su estado a medida que ejecuta las instrucciones. Para ello se ha desarrollado en Verilog una descripción sintetizable de un sistema completo basado en el procesador académico así como un conjunto de herramientas de software que permiten interactuar con dicho sistema. El que el alumno implemente el diseño teórico del procesador visto en teoría en una instancia física e interactúe con ella incrementa notablemente su motivación en el aprendizaje.

AGRADECIMIENTOS

Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación del Gobierno de España a través del proyecto TEC2011-27936 (HIPERSYS).

REFERENCIAS

[1] Sandro Neves Soares and Flávio Rech Wagner. T&D-Bench—Innovative Combined Support for Education and Research in Computer Architecture and Embedded Systems . IEEE Transactions on Education. Vol. 54. No. 4. November 2011.

[2] Antonio Carpeño, Jesús Arriaga, Javier Corredor, and Javier Hernández .The Key Factors of an Active Learning Method in a Microprocessors Course. IEEE Transactions on Education. Vol. 54, No.2, May 2011.

[3] Debiec, P.; Byczuk, M.Teaching Discrete and Programmable Logic Design Techniques Using a Single Laboratory Board. IEEE Transactions on Education. Vol. 54. No. 4. November 2011.

[4] Kim, J. An Ill-Structured PBL-Based Microprocessor Course Without Formal Laboratory. IEEE Transactions on Education.Vol. 55, No.1. February 2012.

[5] Atmel ATmega328P datasheet, http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf

[6] Arduino homepage, http://arduino.cc/

[7] Digilent Basys2 Board Reference Manual, http://www.digilentinc.com/Data/Products/BASYS2/Basys2_rm.pdf

[8] Spartan-3 Generation FPGA User Guide, http://www.xilinx.com/support/documentation/user_guides/ug331.pdf

[9] Xilinx ISE 11 User Guides, http://www.xilinx.com/support/documentation/dt_ise11-1_userguides.htm