PROPUESTA PARA LA ELABORACIÓN DE PRÁCTICAS DE CODISEÑO DE BAJO COSTE

P. Guerra Gutiérrez[1], I. García Vargas[2], R. Senhadji Navarro[2] y G. Jiménez Moreno[2]
[1]Departamento de Ingeniería Electrónica. ETS de Telecomunicación
Universidad Politécnica de Madrid. España. E-mail: pguerra@die.upm.es
[2]Departamento de Arquitectura y Tecnología de Computadores. ETS de Ingeniería Informática
Universidad de Sevilla. España. E-mail:{ignacio,raouf,gaji}@atc.us.es

En esta comunicación se presenta una propuesta para la elaboración de prácticas de sistemas digitales basados en codiseño. A partir de un estudio de los distintos elementos presentes en sistemas de este tipo y teniendo como objetivo la utilización de recursos de bajo coste, se proponen distintas alternativas. Como ejemplo de aplicación se describe un pequeño coprocesador de imágenes, que ha sido utilizado por los autores para realizar sesiones de prácticas durante dos cursos consecutivos alcanzando resultados satisfactorios.

1. Introducción

La elaboración de prácticas de laboratorio en el área de electrónica digital, en especial aquellas relacionadas con el diseño de sistemas, es siempre una tarea compleja. El elevado coste del material necesario, tanto del hardware como del software de diseño (*CAD*), hace que el número de puestos disponibles en los laboratorios de muchas escuelas de ingeniería no sea suficiente para cubrir las necesidades docentes. Además, los alumnos especialmente motivados no siempre tienen la posibilidad de experimentar más allá de lo estrictamente necesario, debido a que dicho material está disponible exclusivamente en las aulas de laboratorio.

La existencia de tarjetas de prototipos de bajo coste con dispositivos programables de cierta complejidad junto a la creciente disponibilidad de herramientas de software libre o con licencia para estudiantes permite suplir algunas de estas deficiencias. De este modo, resulta posible la creación de nuevos puestos de laboratorios con un presupuesto muy reducido e incluso existe la posibilidad de que los alumnos más interesados pueden disponer de todo el material necesario en su propia casa por un coste inferior a 100 EUR.

La realización de pequeños sistemas empotrados basados en codiseño hardware/software (HW/SW) posee un alto valor docente, permitiendo abordar proyectos de cierta complejidad sin demasiada dificultad. De esta manera, es posible abordar en una misma práctica conceptos que van desde el diseño digital hasta la programación software, dependiendo del perfil de los alumnos y de la asignatura. Los autores han realizado un estudio sobre los elementos presentes en un sistema basado en codiseño así como las distintas alternativas existentes para su uso en prácticas de bajo coste. Como resultado, se ha elaborado una práctica consistente en el diseño de un pequeño coprocesador de imágenes que integra todos los elementos propios de un sistema empotrado: microprocesador, periféricos y software dedicado. La práctica propuesta ha sido utilizada por los autores para impartir docencia en la asignatura de Tecnología de Microcontroladores de la Ingeniería Técnica en Informática de Sistemas [1,2].

En el siguiente apartado se describe la metodología propuesta así como se expone el material necesario para la elaboración de las prácticas: tarjeta de prototipo, herramientas, procesadores y otros módulos hardware. En el apartado 3 se resalta la experiencia docente a lo largo de los dos últimos cursos. Por último se cierra el trabajo con las principales conclusiones a las que se han llegado.

2. Estudio de los distintos elementos que integran un sistema basado en codiseño

2.1. La tarjeta de prototipo

La tarjeta de prototipo utilizada para el desarrollo de prácticas de codiseño ha sido la Digilent Spartan-3 Starter Board (Digilent Inc, Pullman, WA, USA)[3] (Fig.1), la cual gira en torno a la *FPGA* de Xilinx Spartan-3 200-4ft256 (*SP3-200k*) [4]. Su elección estuvo motivada por su bajo coste, la adecuación de las prestaciones a los objetivos marcados y la disponibilidad de herramientas de diseño libres, dentro del programa universitario de Xilinx (Xilinx Inc, San Jose, CA USA).

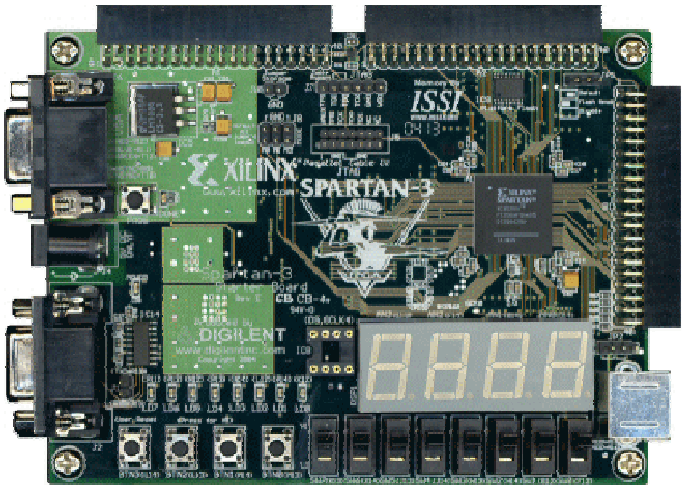

Figura 1: Tarjeta Digilent Spartan-3 Starter Board

La tarjeta dispone de una memoria *FLASH* donde almacenar la programación de la *FPGA*, 1Mb de *SRAM*, un puerto serie, un conector *VGA*, un display 7-segmentos, 7 *LEDs*, 4 botones y 8 switches. Dispone, además, de un reloj de 50 MHz y de un zócalo para colocar un reloj externo. Tres conectores de expansión permiten ampliar la funcionalidad de la tarjeta con diferentes módulos de expansión: ampliaciones de memoria, conectores *USB*, conectores de red, etc. [3]. Estos elementos hacen de la tarjeta una plataforma muy versátil para la realización de prácticas de laboratorio y proyectos fin de carrera.

2.2. Metodología y herramientas

El flujo de diseño utilizado por el alumno para el desarrollo del sistema basado en codiseño consta de cinco etapas clásicas: partición HW/SW, descripción hardware y programación software, cosimulación, síntesis/implementación y programación de la *FPGA*. La Fig.2 muestra el flujo de diseño y las herramientas necesarias para la realización de cada etapa, las cuales son todas de libre distribución o con licencia para estudiantes. Asimismo, para facilitar al alumno la ejecución de la práctica, se han desarrollado *scripts* que automatizan parte de las tareas del flujo de diseño, al tiempo que reducen las posibilidades de que se produzcan errores de implementación.

En la primera etapa, partición HW/SW, se decide qué tareas van a ser ejecutadas por el microprocesador de propósito general y cuáles van a ser implementadas en hardware específico. Existen numerosas técnicas y herramientas para abordar esta problemática [5,6]. Los principales parámetros a considerar son la velocidad, el área y el coste de desarrollo. Las soluciones hardware ofrecen buenas prestaciones en velocidad aunque el área y el coste pueden ser elevados. Por contra, las soluciones software son más flexibles y de menor coste. Las técnicas de codiseño buscan un equilibrio entre ambos esquemas. Una posibilidad es que las tareas más críticas sean implementadas en hardware específico mientras que el resto sean programadas para ser ejecutadas por el microprocesador. El

hardware diseñado actúa, por tanto, como coprocesador. Esta estrategia ha sido utilizada con éxito en numerosas aplicaciones [7] y es por la que se ha optado para la elaboración de las prácticas.

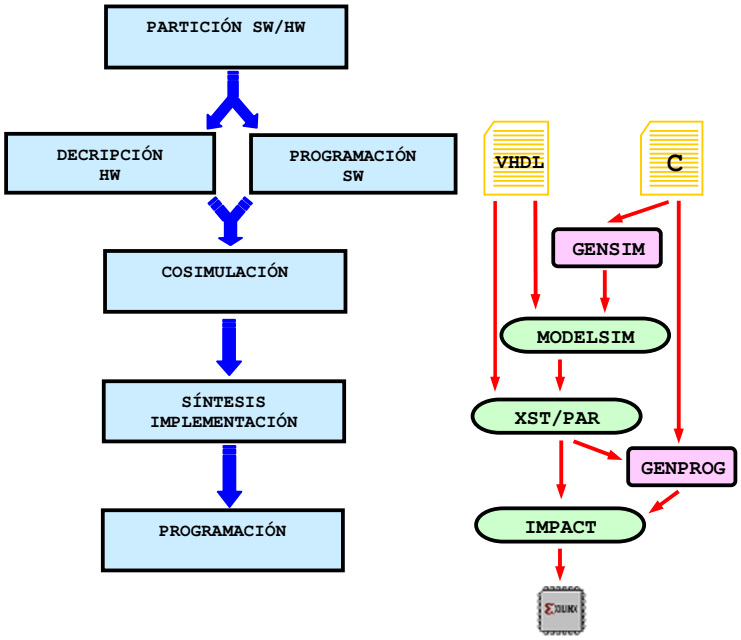

Figura 2. Metodología y herramientas

Una vez finalizada la partición HW/SW comienza la etapa de descripción hardware y programación software. En esta etapa se describe el hardware, se parametriza si fuese necesario el microprocesador y se programan las tareas que éste va a ejecutar. Los diferentes módulos que integran el diseño, incluido el microprocesador, se describen en un lenguaje de descripción de hardware (*HDL*), en concreto las prácticas desarrolladas hacen uso del lenguaje VHDL'93. Por otra parte, la programación del microprocesador se realiza utilizando un lenguaje de alto nivel (ANSI C).

En la etapa de cosimulación se simula el sistema completo, tanto el hardware como el software. Para ello, se necesita una descripción *HDL* del sistema que incluya el código máquina que debe ejecutar el procesador. En primer lugar, se compila el código fuente con el compilador de ANSI C para sistemas pequeños *SDCC* [8]; a continuación, el ejecutable resultante se incorpora, mediante la aplicación *DATA2MEM* (Xilinx Inc, San Jose, CA USA), a la descripción *HDL* de la *ROM* del microprocesador. *DATA2MEM* genera un paquete *VHDL* que permite especificar el contenido de la *ROM* sin necesidad de modificar los ficheros *VHDL* que describen el microprocesador. Se evita así una posible fuente de errores durante el desarrollo de las prácticas. Para automatizar este proceso, los alumnos disponen del *script GENSIM*, desarrollado por los autores, que invoca convenientemente a las aplicaciones *SDCC* y *DATA2MEM*. Finalmente, la descripción *HDL* del sistema se simula con la herramienta *MODELSIM* (Mentor Graphics Wilsonville, OR USA) [9].

En las dos últimas etapas del flujo de diseño se realizan los procesos de síntesis/implementación y de programación de la *FPGA*. La herramienta de síntesis *XST* (Xilinx Inc, San Jose, CA USA) efectúa la síntesis lógica e implementación del diseño y genera el archivo de configuración (*bitstream*) necesario para programar la *FPGA*. El último paso del flujo de diseño consiste en descargar el bitstream en la *FPGA* utilizando la herramienta *IMPACT* (Xilinx Inc, San Jose, CA USA).

Debido a que el código máquina que debe ejecutar el microprocesador forma parte de la descripción *HDL* de la *ROM*, se debería, en principio, repetir el proceso de síntesis e implementación cada vez que se modifica algún elemento del software del sistema. Este proceso puede ser costoso en

tiempo y, por lo tanto, puede ralentizar el avance de la práctica. Para evitar este problema y agilizar el proceso de desarrollo y depurado del SW, se ha creado un *script* llamado *GENPROG* que actualiza directamente los campos del *bitstream* que se corresponden con la configuración de la *ROM*, actualizando así el código máquina sin necesidad de repetir la síntesis. *GENPROG* invoca las herramientas *SDCC* y *DATA2MEN*: de este modo el código máquina generado por el compilador SDCC es incluido en el *bitstream* mediante *DATA2MEM*.

2.3. Procesadores

La elección del procesador es crítica, puesto que se trata del elemento central del diseño que pretendemos abordar. Como etapa previa al desarrollo del laboratorio, se evaluador distintos procesadores teniendo en cuenta la disponibilidad de herramientas para el desarrollo y depurado de software, la complejidad para la creación de nuevos periféricos, los requisitos de área, la velocidad obtenida y la calidad de la documentación existente. Puesto que la plataforma está orientada hacia alumnos, cuestiones como la sencillez, tanto de programación como de creación de nuevos periféricos, o la documentación disponible, primaron sobre las prestaciones alcanzadas. Debido a que los recursos de la *FPGA* son limitados, el área ocupada ha sido un factor determinante.

De entre los procesadores disponibles en la red que han sido objeto de estudio en este trabajo, se presentan cuatro ejemplos atendiendo a la clasificación clásica de los procesadores en función de su juego de instrucciones. Como ejemplo de *CISC* se ha seleccionado un 8051[10,11], como ejemplo de *RISC* se han seleccionado una variante del SPARC V8 (LEON-2) [12] y un procesador abierto con licencia *GNU* (openRisc) [13], finalmente un procesador optimizado para Java (*JOP*) [14,15] en representación de arquitecturas más experimentales de aplicación específica (*ASIP*). La Tabla 1 muestra los resultados de la síntesis en su configuración mínima de los cuatro procesadores sobre una *FGPA* Spartan 3 de Xilinx.

	JOP	**LEON-2**	**OpenRisc**	**8051**	**SP3-200k**
Slices	1222	4177	2399	1459	1920
BRAMs	5	6	4	6	12
Frec. max. (MHz)	60.79	34.15	50.59	28.50	--

Tabla 1: Comparativa entre procesadores

De entre los procesadores considerados inicialmente, se decidió montar las prácticas en torno al 8051 por ser este uno de los procesadores mejor estudiado dentro del programa de la asignatura a la que está enfocada la práctica y porque, dentro de los procesadores convencionales, es el único que puede ser sintetizado sobre la *FPGA* disponible, restando algo de área para la introducción de periféricos adicionales, como puede ser una *UART* (Universal Asynchronous Receiver Transmitter). Sin embargo, esta implementación del 8051 no está exenta de limitaciones, siendo las más importantes la ausencia de interrupciones y un interfaz para la memoria de almacenamiento poco flexible.

2.4. Otros módulos hardware

A lo largo de los últimos años se ha venido desarrollando una gran cantidad de descripciones hardware de numerosos dispositivos que se distribuyen mediante licencia *GPL*. Se pueden encontrar módulos como controladores de E/S, procesadores de señal, módulos para comunicaciones, procesadores matemáticos, etc. Opencores [16] es el caso más significativo. Con estos módulos previamente diseñados el alumno puede desarrollar sistemas relativamente complejos sin demasiada dificultad.

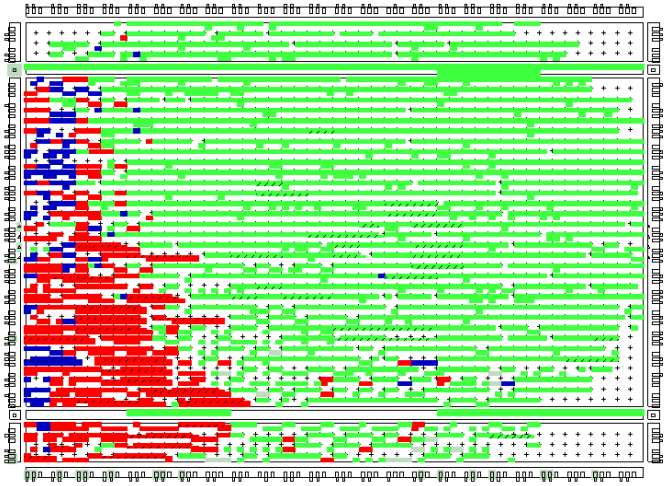

Figura 3. Floorplan del diseño sintetizado sobre la FPGA XCS3-200k. Se distinguen los tres bloques principales del diseño: el microprocesador en verde, el módulo CORDIC en rojo y la UART en azul.

3. Experiencia docente

La propuesta descrita ha sido utilizada por los autores para impartir sesiones de prácticas en la asignatura de Tecnología de Microcontroladores de tercer curso de Ingeniería en Informática de Sistemas durante los cursos 2004/2005 y 2005/2006, habiendo cursado este laboratorio unos 50 alumnos en cada curso. Se trata de una asignatura optativa de 5 créditos *ECTS* multidisciplinar aunque propia de la especialidad, cuyos contenidos están ligados tanto a la estructura de computadores como al software, e incluso la electrónica. Esto se debe a que el microcontrolador no deja de ser un computador en un solo chip, y por tanto, su estudio es el mismo que el de un computador completo, con todos sus niveles.

En esta asignatura se estudian con detalle diferentes microcontroladores, entre ellos el 8051, y se realizan prácticas cuyo objetivo es la programación de estos micros para una determinada aplicación, y dentro del plan de estudios sus contenidos se ampliarán en otras de cursos superiores de la carrera de Ingeniero en Informática, como puede ser Arquitectura de Sistemas en Tiempo Real. Los alumnos que cursan esta asignatura no tienen porqué conocer lenguajes de especificación hardware como el VHDL ni estar habituados a utilizar herramientas de diseño. Sin embargo, esto no es un inconveniente para introducir en la programación del curso la noción de codiseño y realizar prácticas relacionadas con este concepto, siempre y cuando la parte hardware esté muy elaborada. De esta manera, los alumnos sin demasiado esfuerzo adicional pueden programar microcontroladores en un contexto de diseño de SoCs (System on Chip).

A los alumnos se les propone realizar un sencillo coprocesador de imágenes capaz de realizar giros. La tarjeta de prototipo debe recibir por el puerto serie la imagen y el ángulo de giro, realizar la rotación y posteriormente enviar la imagen procesada por el puerto serie. Para ello los alumnos disponen además del microcontrolador 8051, de una *UART* y de un módulo *CORDIC* (COrdinate Rotation DIgital Computer, [17]).

El modelo *VHDL* de la *UART* es un proyecto de Open Core. Se trata de un pequeño controlador serie (con funciones limitadas respecto a una *UART* completa) capaz de transmitir/recibir datos a una velocidad configurable por el diseñador. Genera 2 tipos de interrupción (por líneas separadas): dato enviado y dato recibido. El módulo *CORDIC* es un *core* diseñado por Xilinx que permite realizar rotaciones, calcular senos y cosenos, arco tangentes, etc. Puede configurarse de diferentes maneras proporcionando diferentes funcionalidades y formatos de datos mediante la herramienta de Xilinx para generación de módulos denominada *CORE Generator*. La Fig.3 muestra la síntesis final sobre la

FPGA de la tarjeta prototipo, se observa que la práctica totalidad del área disponible se dedica a la implementación del microcontrolador.

El alumno dispone de dos sesiones de 4 horas para diseñar el sistema completo. Por un lado, debe interconectar los distintos módulos que se le proporcionan. Para ello, debe multiplexar los puertos del microcontrolador, pues no dispone de tantos puertos de salida como hacen falta, y añadir un divisor de reloj mediante un *DCM* (Digital Clock Manager) [4], pues la síntesis del microcontrolador sobre la *FPGA* de la tarjeta de prototipado no soporta el reloj de 50 MHz (véase la Tabla 1). Por otro lado, el alumno debe programar el 8051 para que controle la comunicación serie, recorra píxel a píxel la imagen e interactúe con el módulo *CORDIC* para realizar la rotación de coordenadas.

Además de las herramientas descritas en la sección 2.2 los autores proporcionan a los alumnos herramientas para evitar la manipulación de formatos gráficos. Se le proporciona un programa para convertir un archivo gráfico en un archivo de formato plano (sin compresión ni cabecera) que contiene también el ángulo que se desea girar la imagen. También se le proporciona un programa para el proceso inverso. La imagen original y rotada son trasferidas entre el PC y la tarjeta mediante la herramienta *Hyperterminal* de Windows.

Los alumnos deben entregar una memoria que describa el proceso seguido y que contenga el programa C del microcontrolador. También deben responder a un cuestionario en el que se le preguntan cuestiones relacionadas tanto con el hardware como con el software del sistema diseñado. Una de las cuestiones más interesantes que se le proponen al alumno es la de esbozar una solución al problema de discretización que tiene la matriz general de rotación cuando se aplica a una imagen digital. Para ello se les invita a utilizar la descomposición de Paeth [18] que se muestra en la Ec.1.

$$R = \begin{pmatrix} 1 & \tan(\frac{\phi}{2}) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sin\phi & 1 \end{pmatrix} \begin{pmatrix} 1 & \tan(\frac{\phi}{2}) \\ 0 & 1 \end{pmatrix} \quad (1)$$

En este caso el módulo *CORDIC* debe configurarse exclusivamente para calcular las funciones trigonométricas seno y coseno (la tangente puede obtenerse por división), y a partir de ellas el microcontrolador debe realizar las translaciones unidimensionales que propone Paeth.

La experiencia docente fue muy satisfactoria. Muchos alumnos expresaron su asombro ante la multitud de posibilidades que les ofrece la lógica programable y les llamo la atención la aparente sencillez del proceso de diseño, cuestión ésta que ayuda a desmitificar la percepción generalizada entre los alumnos de que la electrónica es demasiado compleja. Prácticamente todos los grupos finalizaron correctamente la práctica e incluso algunos alumnos mostraron interés en adquirir este tipo de tarjetas para seguir desarrollando sistemas de este tipo. Las encuestas de la Evaluación de la Docencia del curso 2004/2005 corroboran estas impresiones.

4. Conclusiones y trabajo futuro

Se ha realizado una propuesta para la elaboración de prácticas de codiseño HW/SW que incluye sólo herramientas de libre distribución o con licencia para estudiante y que hace uso de tarjetas de prototipo de bajo coste como plataformas de desarrollo, cuya capacidad es suficiente para albergar dos de los microprocesadores analizados. Como ejemplo de aplicación se ha diseñado un pequeño coprocesador de imágenes que ha sido utilizado por los autores para realizar sesiones de prácticas durante dos cursos consecutivos en la asignatura de Tecnología de Microcontroladores de la Ingeniería Técnica en Informática de Sistemas con resultados satisfactorios

En la actualidad se trabaja con el código *VHDL* del *core* del 8051 con el objeto de corregir algunas de sus limitaciones. De este modo, se encuentra ya disponible una nueva interfaz de acceso a la memoria externa que permitirá mapear los registros de los periféricos directamente en memoria,

aspecto que permite un mejor aprovechamiento el juego de instrucciones así como de disponer de un espacio de E/S mayor (esta implementación del 8051 dispone de 4 puertos de E/S); se pretende ahora abordar las modificaciones necesarias sobre el código existente para poder disponer de interrupciones.

Igualmente, se está trabajando en la realización en un pequeño entorno visual de desarrollo para facilitar a los alumnos la integración de la tarjeta de prototipo y las distintas herramientas. Entre las facilidades que proporciona el entorno destacan las siguientes: parametrización del microcontrolador, definición del pinout, simulación mediante animación gráfica, etc.

A más largo plazo, se estudia la posibilidad de emplear esta plataforma en otras asignaturas, bien partiendo de un nivel de abstracción más bajo en el que el alumno realizará sus propios periféricos, bien a un nivel más alto, en el que el alumno haría uso de herramientas como *POLIS* para el particionado HW/SW en lugar del particionado manual considerado actualmente.

Referencias

[1] Practicas de la asignatura. http://www.atc.us.es/~raouf/practicatm.zip

[2] Senhadji, R et al. *Metodología para la realización de prácticas de codiseño*. V Jornadas de Computación Reconfigurable y Aplicaciones, pp 395-400 (2005).

[3] Digilent, Inc., http://www.digilentinc.com

[4] Xilinx Inc. DS099. Spartan-3 FPGA Family. http://www.xilinx.com

[5] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A, Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, E. Sentovich, K. Suzuki, B. Tabbara, Hardware-Software Co-Design of Embedded Systems: The Polis Approach. Kluwer Academic Press , Junio (1997).

[6] H. Hsieh, L. Lavagno, C. Passerone, C. Sansoe, A. Sangiovanni-Vincentelli. Modeling Micro-controller Peripherals for High-Level Co-simulation and Synthesis. Proc. International Workshop on Hardware-Software Codesign, Marzo (1997).

[7] Cabrera, S. Sánchez-Solano, P. Brox, A. Barriga and R. Senhadji. Hardware/software codesign of configurable fuzzy control systems. Applied Soft Computing, Volumen 4(3), pp 203-322 (2004).

[8] Small Device C Compiler, http://sdcc.sourceforge.net

[9] Mentor Graphics Co., http://www.model.com

[10] Oregano Systems. *8051 IP Core*. http://www.oregano.at

[11] The UCR Dalton Project, *8051 core*. http://www.cs.ucr.edu/~dalton

[12] Gaisler Research. *LEON2 processor*. http://www.gaisler.com

[13] OpenRISC project, http://www.opencores.org/pnews.cgi/list/or1kM.

[14] Schoeberl. *JOP: A Java Optimized Processor for Embedded Real-Time Systems*. PhD Thesis, Fakultaet fuer Informatik: Vienna University of Technology (2005).

[15] M. Schoeberl. *Design Rationale of a Processor Architecture for Predictable Real-Time Execution of Java Programs*. 10th International Conference on Real-Time and Embedded Computing Systems and Applications (2004).

[16] Opencores, http://www.opencores.org

[17] Ray Andraka. *A survey of CORDIC algorithms for FPGAs.* Proceedings of 6th ACM/SIGDA international symposium on FPGAs, pp 191-200 (1998).

[18] A.W. Paeth. *A fast algorithm for general raster rotation*. In Graphics Interface, pp.77-81 (1986).