Índice

Índice de figuras

1. INTRODUCCIÓN

Desde los principios de la tecnología de semiconductores, los sistemas electrónicos han venido experimentando un constante crecimiento en complejidad, debido a que las prestaciones que tiene que ofrecer una aplicación concreta son cada vez mayores y de naturalezas más diversas. Esto ha dado lugar a que un sistema tenga a la vez restricciones aparentemente incompatibles, como pueden ser de trabajo en tiempo real, de tolerancia a fallos o de procesado de grandes flujos de datos. Este incremento de complejidad y diversidad en las restricciones ha motivado la aparición de diseños híbridos en los que interactúan elementos hardware y software, ya que dichos elementos pueden complementarse para resolver problemas de distinta naturaleza.

El codiseño hardware/software es la tarea de diseñar el sistema hardware y generar el código del sistema software de manera conjunta (sistema mixto), de tal forma que el comportamiento del sistema global tenga las mismas prestaciones que en la descripción funcional y cumpla con objetivos que serían inalcanzables mediante metodologías tradicionales de diseño. Entre las alternativas para diseñar e implementar un hardware específico encontramos ASICs (Application-specific integrated circuit), FPGAs (Field-programmable gate array), CPLDs (Complex Programmable Logic Device) entre otros. El uso de ASICs posibilita desarrollos con producción a gran escala a bajo costo y es de masiva utilización en este tipo de aplicaciones. Los CPLD y las FPGA son circuitos de de alta densidad programables por el usuario en un tiempo reducido y sin la necesidad de verificación de sus componentes, tarea ya realizada por el fabricante al tratarse de un producto estándar. El procesamiento digital de señales , prototipado de ASICs , tratamiento de imágenes , reconocimiento de voz , glue logic son algunas de las aplicaciones de este tipo de dispositivos. Existen diferentes formas de llevar adelante el diseño e implementación de un sistema digital para FPGA, entre ellas tenemos la realización de un diseño esquemático , herramientas específicas (provistas por el fabricante) y la utilización de un lenguaje de descripción de hardware HDL (Hardware description language) entre los que se encuentran lenguajes como Verilog y VHDL, ambos de gran aceptación en los ambientes industrial y académico. Estos lenguajes proporcionan gran versatilidad para el desarrollo de hardware, permitiendo especificar, diseñar, simular y verificar sistemas digitales complejos, mediante el apoyo de un universo de herramientas EDA (Electronic Design Automation).

Actualmente las FPGA cuentan con una gran cantidad de recursos disponibles (Compuertas lógicas , Bloques de RAM) para implementar diseños digitales complejos. Las FPGA pueden ser usadas para implementar cualquier función lógica que un ASIC pueda realizar. Una de las grandes ventajas del uso de FPGA en la etapa de prototipado es su capacidad de reconfigurar el diseño parcial o totalmente para su actualización o corrección de errores con un costo relativamente bajo a diferencia del prototipado sobre ASICs. Durante la etapa de producción los ASIC resultan de muy bajo costo respecto de la producción de FPGA y esto se traduce en una gran ventaja para desarrollos que deben ser producidos a gran escala.

Las arquitecturas reconfigurables combinan parte de la flexibilidad del software con la

gran performace del hardware utilizando chips reconfigurables como FPGAs. Una opción de gran potencialidad son las arquitecturas reconfigurables run-time o dinámicas que se sostienen en DRL (Dynamically reconfigurable logic). Un ejemplo de esto es la familia Virtex de Xilinx, que es parcialmente reconfigurable en tiempo de ejecución, este método se conoce como run-time. Claramente este tipo de dispositivos puede ser usado como arquitecturas destino de codiseños hardware/software (HW/SW) que proveen la flexibilidad de los procesadores software y la eficiencia y rendimiento de los coprocesadores hardware.

El desarrollo de aplicaciones de software se ve limitado a los recursos disponibles en los microprocesadores comerciales. El software necesita del soporte de un procesador para su ejecución, así la elección de este elemento conlleva algunas dependencias respecto de las herramientas a utilizar , algunas de estas son : compiladores , ensambladores , depuradores y herramientas de simulación. Con ellas se logra trasladar el software a un entorno de ejecución adecuado dentro del procesador y depurarlo . El lenguaje de programación elegido debe permitir trabajar con el nivel de abstracción necesario para simplificar la tarea de desarrollo de software. El lenguaje C es uno de los más utilizados para desarrollar aplicaciones que requieran ,por ejemplo , de compilación cruzada (Cross-compilling) y se complementa con las herramientas libres de compilación y depuración como son GCC y GDB.

Los nuevos proyectos a veces requieren nuevas características de los cores existentes. El proveedor del núcleo puede hacer estas modificaciones (solución comercial) con un incremento sustancial del coste del núcleo. Otra posibilidad (Solución Ad-hoc) es el uso de cores de código abierto con el fin de crear un núcleo de desarrollo adaptable. El enfoque de código abierto tiene varias ventajas: el núcleo posee un costo muy bajo e inclusive cero, los usuarios puede tener acceso al código fuente y hay un grupo de desarrolladores que proporcionan conocimientos para mantener y mejorar el núcleo. Sin embargo, también puede tener varias desventajas como la inestabilidad (el grupo de cambio o de desarrollo desaparece), desarrollo incompleto, deficiente , documentación pobre y una mala metodología de verificación.

Los microprocesadores “softcore"(núcleo software) son aquellos cuyo hardware está íntegramente implementado en un lenguaje de descripción de hardware.Posibilitan un desarrollo confiable con facilidad de ajustar el mismo a las necesidades de aplicación que deben satisfacerse e incluso el desarrollo de microprocesadores doble o múltiple núcleo. Entre los microprocesadores softcore de código privativo más importantes encontramos MicroBlaze de Xilinx , Nios y Nios II de Altera y Cortex M1 de ARM. Como contrapartida encontramos los microprocesadores softcore Open Source (Código Abierto) OpenSPARC T1 de Sun (64 bits) y OpenRISC de Open Cores.

El procesador OpenRISC de 32 bits se comunica con sus periféricos por medio de un bus tipo Wishbone , también de código abierto. OpenRISC puede ser implementado fácilmente en chips de cualquier fabricante (Xilinx , Altera , Actel) razón por la cual es el elemento central de algunos proyectos SoC de código abierto tales como MinSoC de OpenCores u ORPSoC de OpenCores. Los sistemas embebidos SoC (System on a chip) integran los componentes de hardware necesarios para cumplir una funcionalidad específica. Una arquitectura típica SoC integra un procesador , un procesador o un núcleo

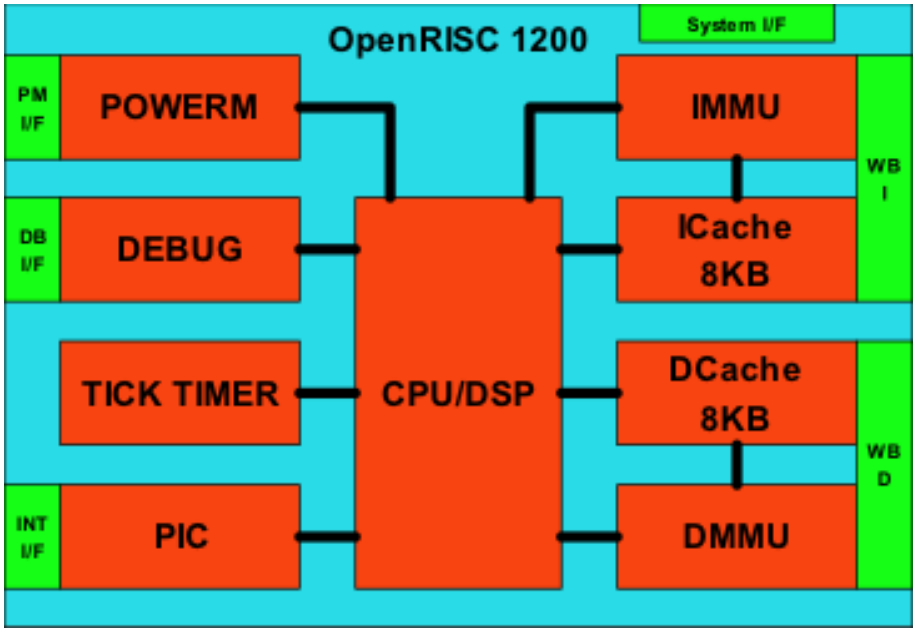

Figura 1: OpenRISC 1200 - Diagrama de Módulos

DSP como elemento central , bloques de memoria (ROM , RAM , EEPROM) , periféricos , osciladores , entre otros.

2. PROPUESTA

El objetivo principal de este trabajo es extender la capacidad de procesamiento de un soft-processor para el cálculo matricial utilizando una metodología de codiseño HW/SW y bajo el modelo de software libre. Se utilizará una plataforma que incluye una CPU (OpenRISC 1200)y algunos periféricos básicos tales como UART, ADV(Advanced Debug System) , JTAG , Ethernet, RAM , StartUp (Bootloader).Toda la implementación del proyecto posee una licencia LGPL lo que otorga al programador la capacidad de modificar el código a necesidad. Un conjunto de herramientas de desarrollo de software (compilador, ensamblador, debugger) también de código abierto ayudan al desarrollo de aplicaciones para esta arquitectura y son provistas en el proyecto OpenRISC.

El proyecto constará con las siguientes etapas:

- La primera etapa del proyecto tendrá como objetivo principal afianzar los conocimientos sobre el diseño digital en FPGA mediante lenguajes de descripción de hardware como Verilog y VHDL. Se desarrollarán módulos generales tales como multiplicadores de matrices , conversores serie-paralelo y máquinas de estado finitas. Se pretende obtener capacidad de desenvolvimiento en este tipo de implementaciones para luego aplicar dichos conocimientos en módulos OpenSource.

- Se desea en una segunda instancia poner en marcha una plataforma MinSoC sobre un kit de desarrollo XILINX XtremeDSP Starter Platform Spartan 3A DSP 1800A

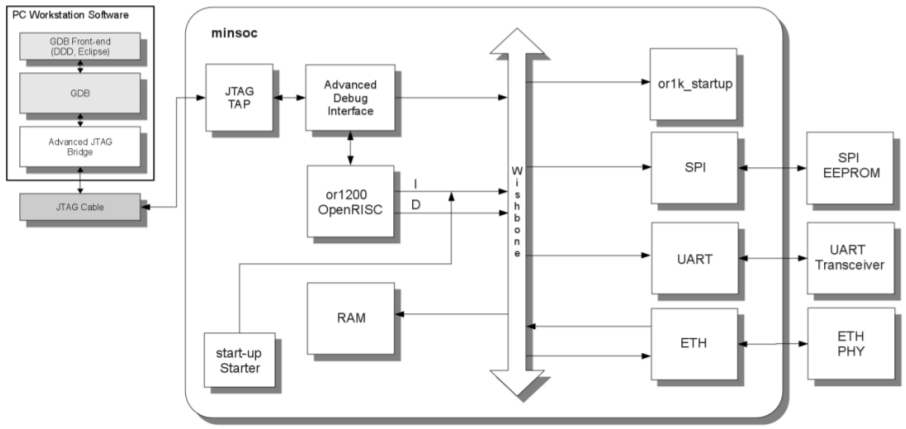

Figura 2: MinSoC - Diagrama de Módulos

Edition para luego describir y documentar en su totalidad el proceso realizado hasta lograr obtener un sistema completamente funcional. Dentro de las etapas de este proceso se tiene en primera instancia lograr el funcionamiento del SoC base. Luego se desarrollará software de prueba para verificar el desempeño del sistema analizando los procesos de carga de programas , depuración de los mismo , uso del bootloader del sistema y carga de un firmware básico en memoria SPI Flash. Finalmente se realizarán modificaciones de prueba en la implementación del hardware para interpretar completamente la estructura del diseño de la misma. Este trabajo presentará comparativas de performace en base a Benchmarks adecuados sobre el microprocesador OpenRISC frente a otros procesadores que se encuentran actualmente en el mercado resaltando ventajas y desventajas del uso de procesadores open source en aplicaciones de investigación , universitarias , desarrollos a baja escala .

- Finalmente se realizará el desarrollo e integración de un coprocesador matricial que será integrado a la implementación básica del MinSoC para aumentar la capacidad de cálculo del procesador OpenRISC. Existen al menos dos posibilidades para implementar esto, el primer caso es agregar a la arquitectura del microprocesador la funcionalidad en cuestión con la respectiva adaptación del resto de las herramientas (nuevas instrucciones , nuevos módulos) para su utilización. Un segundo caso sería añadir la funcionalidad por medio de un módulo conectado al bus común (Wishbone). Esta solución puede ser adaptada fácilmente a otras implementaciones SoC que utilicen como interconexión al bus WISHBONE. Se desarrollará, implementará y documentará el coprocesador y se realizarán los tests de performace adecuados que permitan contrastar los resultados de la ejecución en el nuevo entorno respecto de los desarrollos netamente implementados por software. Estas comparativas permitirán analizar las ventajas y desventajas de la utilización de aplicaciones co-diseñadas en hardware/software y su posibilidad de aplicación en diferentes campos.