DISEÑO E IMPLEMENTACIÓN DE UN MICROPROCESADOR EN DISPOSITIVO DE LÓGICA PROGRAMABLE PARA SU UTILIZACIÓN EN SISTEMAS COLABORATIVOS

Polimeni Julián, Rossi Grad Sebastián, Puga Gerardo, Aróztegui Walter, Rapallini José

UIDET CeTAD - Departamento de Electrotecnia – Facultad de Ingeniería – UNLP
Calle 116 y 48, Piso 2 – (1900) La Plata
josrap@gmail.com, walter.aroztegui@gmail.com

INTRODUCCIÓN

El objetivo principal de este trabajo es el de realizar la implementación de un microprocesador para sistemas colaborativos mediante el uso de lenguajes de descripción de hardware, comprobando su funcionamiento en un dispositivo de lógica programable. Adicionalmente, se consideró como un caso de interés, el análisis y la puesta en práctica de una serie de técnicas profesionales para el desarrollo de sistemas trazables y respaldados por evidencia de funcionamiento: control de versiones, herramientas de productividad, verificación y validación de sistemas.
El trabajo se enmarca en el proyecto de sistemas colaborativos, que consiste en el estudio, modelización y caracterización de un estándar de microsistemas autónomos especializados que puedan interactuar entre sí colaborando con sus distintas funciones en la resolución de un problema o acción específica sobre el medio que comparten. Los estudios teóricos de la interrelación de microsistemas con protocolos propios y con determinados grados de inteligencia colectiva, no centralizada, sino independiente, llevan a la implementación de procesos inteligentes, tratando de utilizar arquitecturas de procesadores lo más sencillas posibles para la ejecución de decisiones provenientes de módulos de sensado y actuación, que interactúan con el entorno físico. Es en este contexto donde se pretende que se desenvuelva la presente implementación de un microprocesador, con las características de sencillez y operatividad necesarias, en su rol de parte integral de módulo de inteligencia en un microsistema.

PARTE EXPERIMENTAL

Metodología y herramientas

Para poder realizar la descripción circuital fue necesario hacer uso de un lenguaje de descripción de hardware (VHDL). En cuanto al entorno de descripción se utilizó el software Quartus de Altera y para las simulaciones se hicieron uso de dos programas diferentes, ModelSim, también perteneciente a Altera y un software de testeo unitario para HDL denominado VUnit. Las pruebas en hardware fueron llevadas a cabo utilizando un kit de desarrollo DE0 [1][2].
Al llevar a cabo las distintas tareas del proyecto de forma organizada se hizo necesario el mantener una metodología de trabajo estructurada, ordenada y repetitiva, que permitió acelerar los procesos de desarrollo y de pruebas y, a la vez, optimizarlos en cuanto a tiempos y recursos. La forma en la que se trabajó fue la denominada bottom-to-top [7], es decir, se comenzó con los circuitos más pequeños, tanto en tamaño como en complejidad, para luego ascender en dichas escalas repitiendo la metodología de trabajo en cada etapa en la que se atravesó. La forma de proceder en cualquiera de estas etapas fue la siguiente:

- *Diseño del circuito*: En esta etapa se proponen distintos modelos para el circuito en cuestión, analizando la necesidad de sincronismo, los puertos de entrada y salida, su comportamiento y su composición. [3][4]
- *Verificación*: Una vez descripto el circuito inicial en su totalidad se procede a las distintas simulaciones tanto en VUnit como en ModelSim, para verificar el funcionamiento y analizar posibles fallas. [2].

- *Iteración:* En caso de que la descripción inicial del circuito no haya superado los ensayos, se procede a realizar una revisión de los componentes del circuito que presentasen dificultades para luego llevar a cabo las modificaciones necesarias
- *Validación:* Este paso no es necesario realizarlo sobre cada circuito descripto a lo largo del proceso sino sobre el circuito final en cuestión. Para llevar a cabo esta etapa es necesario implementar una plataforma de ensayo sobre el circuito previo a ser sintetizado sobre la FPGA. [6]

Elección del microprocesador

En primer lugar, no se debía perder de vista que la finalidad de este proyecto era la de lograr una implementación de un microprocesador para microsistemas, por lo que el mismo tendría que, en algún punto, poder comunicarse con el mundo externo o con otros microprocesadores. La posibilidad de añadir bloques de comunicación sería determinante a la hora de definir qué tipo de procesador desarrollar. A su vez, una posible continuación del proyecto consiste en realizar una implementación en microelectrónica, por lo que la toma de la decisión se veía directamente influenciada por este aspecto. Era necesario que el microprocesador a implementar no fuera de una complejidad superior y sus partes perfectamente determinadas para que se pueda realizar la descripción física (layout) en alguna tecnología CMOS y finalizar con la síntesis del circuito integrado. [5][8]
En base a los condicionantes y características propuestas, se llegó a reducir el número de opciones a tres:

Intel 8051: Un microcontrolador comercial muy popular de los años ’80 con una arquitectura Harvard, y distintos módulos como temporizadores, UART, unidades dedicadas de operaciones booleanas, memorias internas y externas, entre otras. Desde el punto de vista de software al ser un microprocesador comercial muy conocido no presentaba mayores problemas en cuanto a compiladores tanto del lenguaje C como de *assembler*. Como desventaja se puede encontrar que al ser un microprocesador de una complejidad mayor, su implementación en microelectrónica podría acarrear complicaciones y ser muy compleja.

Implementación académica: Este es un microprocesador con fines académicos, de una estructura muy simple de 8 bits de palabra de datos, un banco de registros, acumulador, pero sin ningún tipo de implementación más allá del aspecto teórico. Desde un punto de vista de aplicación de microelectrónica, este microprocesador era ideal, ya que cada bloque se encontraba descripto de forma detallada. En cuanto a los aspectos de aplicabilidad y software, esta opción no era la más conveniente, ya que, al ser un microprocesador académico, no existía implementación alguna ni compiladores de ningún tipo.

ZPU: Esta última opción era un microprocesador de software libre, por lo que existen diferentes implementaciones. Es un microprocesador que trabaja sin registros de propósito general sino en base a una pila, con un ancho de palabra de datos variable, y que se encuentra optimizado para ser utilizado con una FPGA prescindiendo de una cantidad relativamente grande de elementos lógicos. Desde el punto de vista de microelectrónica puede no ser el mejor, pero no por eso deja de ser una buena opción a la hora de pensar en la fabricación del circuito. Por otra parte, teniendo en cuenta el aspecto de software, éste microprocesador tiene la ventaja de la existencia de compiladores que permitirían desarrollar programas para él en lenguaje de programación C. Además, posee la versatilidad suficiente como para poder anidarle otros módulos existentes, realizando algunas pocas modificaciones teniendo en cuenta el aspecto de la comunicación con el mundo externo.
En base a un estudio más detallado de cada una de las opciones, se llegó a la conclusión de que el mejor camino a tomar era el de implementar el ZPU, por las razones ya mencionadas.

ZPU
El ZPU es un microprocesador del tipo de los basados en ¨pila¨, de desarrollo libre. Fue diseñado de forma tal que utilice la menor cantidad de elementos lógicos posible, para

liberar una gran porción de éstos con otros propósitos, razón por la cual es cada vez más demandado en la industria de los dispositivos lógicos programables y por ello, también posee una gran cantidad de implementaciones y tiene compatibilidad con un compilador de GNU, brindado gran cantidad de herramientas para la creación de programas.
La potencia de procesamiento no es su fuerte, ya que no almacena resultados en registros intermedios, sino que trabaja directamente sobre la memoria con una pila. Su set de instrucciones es reducido, y solamente es de implementación obligatoria por hardware aproximadamente un tercio del mismo, ya que el resto pueden ser implementadas mediante emulación de instrucciones por software con el fin de reducir los recursos lógicos de hardware en sistemas restringidos. También cabe destacar que utiliza el PC (*Program Counter* o Contador de Programa) de una forma no convencional, debido a que las posiciones de memoria del ZPU son de 32 bits de ancho y los códigos de operación son de 8 bits de ancho, cada posición de memoria posee 4 códigos de operación. Más allá del tamaño del PC, sus 2 bits menos significativos son siempre utilizados para seleccionar el código de operación dentro de una misma posición de memoria, mientras que con los bits restantes se selecciona la posición de memoria. La arquitectura de este procesador puede ser descripta como de tipo Von Neumann ya que instrucción y dato no pueden ser traídos de la misma en paralelo, pese a tener una memoria RAM de dos puertos.
Se pueden diferenciar dos tipos de implementaciones: una llamada ZPU y otra llamada ZPU *small*. En su descripción son prácticamente iguales haciendo la salvedad que el ZPU *small* implementa una menor cantidad de instrucciones directamente en hardware, ahorrando por lo tanto recursos lógicos. Es por ello que se decidió tomar la implementación del ZPU *small para* utilizarlo como punto de partida para realizar el diseño propio, al que se denominó **ZPU PR**.

ZPU PR

Para lograr la implementación se optó por proponer un circuito en base a la división más clásica de los microprocesadores, esto es, tres grandes bloques: la **CU** (Unidad de Control), el ***datapath***, y la ***memoria***. En una primera aproximación al diseño se realizaron los diseños iniciales de cada uno de los bloques, algunos de los cuales fueron sufriendo modificaciones para lograr el funcionamiento deseado en conjunto.
Partiendo de esta base, el *trabajo* comenzó por el diseño del *data path*. Conociendo el set de instrucciones con el que el microprocesador debe cumplir, y en base a la implementación del ZPU *small*, las exigencias que el *data path* debía satisfacer estaban claramente definidas. El mismo debería ser capaz no solo de realizar el set de instrucciones mínimo en su totalidad, sino que además debía realizar cada instrucción en la cantidad de ciclos correspondientes para poder sacarle provecho en un futuro a las herramientas que llevaron a tomar la decisión de implementar este microprocesador. Como entidad final el *data path* posee 5 entradas y 5 salidas. Sus entradas son: ambos puertos de lectura de la memoria RAM de 32 bits de ancho, la entrada de lectura de la memoria externa también de 32 bits de ancho, los 7 bits menos significativos del código de operación y la palabra de control de 23 bits de ancho. Sus salidas son: un puerto de dos bits de los bits menos significativos del PC, y ambos puertos de direccionamiento y escritura que se conectan directamente con la memoria RAM, de 13 y 32 bits de ancho de palabra respectivamente.

La CU se constituye básicamente como una máquina de estados finitos (MEF). Posee 7 puertos de entrada, los últimos dos bits del PC, el bit 15 del puerto de lectura A de la memoria RAM, la señal de un bit de aviso de ocupación de la memoria externa, la señal de un bit de interrupción externa del procesador, y el puerto de lectura B de la memoria RAM, de un ancho de 32 bits. Sumado a estos puertos se encuentra el puerto de reset externo al ZPU y la señal de reloj. Cada uno de ellos cumple una función específica en la toma de decisiones de la palabra de control que debe generar el bloque. Sumado a la palabra de control de 23 bits, la otra vía de comunicación de la CU con el *data path* es mediante los bits del 6 al 0 del código de operación que se encuentra en ejecución. Muchas veces se utiliza

parte del código de operación como dato de carga inmediata o direccionamiento de saltos ya sean del PC o del SP. La palabra de control no es el único puerto de salida que posee CU, ya que esta palabra solo se encarga del control del *datapath*, y la CU debe llevar la cuenta del control de todo el resto de los módulos presentes. Es por ello que dentro de sus puertos de salida se encuentran también: ambos habilitadores de escritura de la memoria RAM, el habilitador de escritura externa, el habilitador de lectura externa, y la señal llamada Break, que da cuenta el estado del procesador cuando comienza a hacer la depuración del código de programa. En su totalidad los puertos de salida de la CU terminan siendo 7.

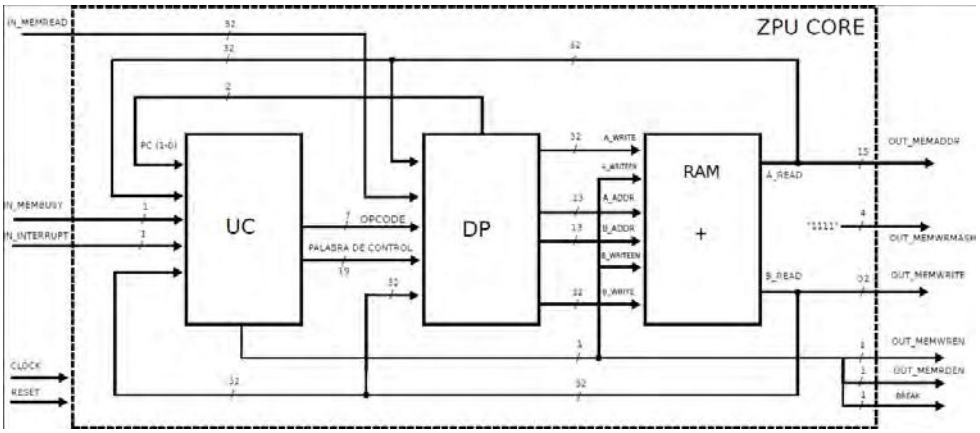

Figura 1. Esquema del Procesador basado en ZPU

La memoria RAM es una memoria de dos puertos de tamaño variable, no solo en las direcciones sino también en el ancho de palabra que almacena. El ancho de palabra puede ser de 16 o 32 bits dependiendo de la estructura elegida por el programador a la hora de implementar ZPU. En este caso la elección fue de un largo de palabra de 32 bits. Se acordó que el tamaño de la palabra de direccionamiento sería de 13 bits, lo que da como resultado 8192 posiciones de memoria. De las cuales las ultimas 32 posiciones pertenecen a la pila, y las primeras 256 a las rutinas de instrucciones emuladas, de reset y de interrupciones. Como resultado esto deja 7904 posiciones para memoria de programa. Pero es importante recordar que por posición se pueden almacenar hasta 4 instrucciones, por lo que en definitiva termina habiendo espacio para almacenar más de 31000 instrucciones.

Verificación

Para realizar las pruebas de los tres grandes bloques se hizo uso de la herramienta VUnit, integrando distintos bancos de prueba dentro de una misma simulación para llevar un control más exigente de los bloques y su funcionamiento. Esto permitió optimizar los tiempos, el control y la manipulación de las simulaciones en comparación con otras herramientas como el ModelSim. El criterio de prueba de cada uno de los bloques no es el mismo dependiendo de su complejidad y su función, en bloques simples como un incrementador o un decrementador, basta solo con realizar una prueba iterativa de todo el rango de valores al que el bloque será sometido, comparando su salida con el valor esperado. En bloques como los MUX (multiplexores), no es de vital importancia qué valores posean a la entrada sino, que se cumpla la selección de este en forma estricta, en todas sus variantes. Es así que las pruebas se llevan a cabo, pero no necesariamente con todo el rango de valores en cada una de las entradas. Bloques como la ALU, con un aumento en su complejidad tanto como en importancia en las distintas instrucciones que debe realizar el *data path*, deben ser sometidos a un banco de prueba de mayor tamaño, ya que no solo importa la operación que se desea realizar sino también el resultado de la misma, y los valores que se ingresan al bloque. Una vez verificado el funcionamiento de los tres bloques principales que componen el ZPU PR, para continuar se realizó la conexión de los bloques. Para ello fue necesario crear una nueva entidad que ya sería la entidad final, es decir, el microprocesador completo. Esta entidad, posee 5 puertos de entrada y 6 puertos de salida.

El esquema de pruebas debía integrar el uso de los tres bloques y revisando las señales internas, hacer un análisis a lo largo del tiempo de la evolución de los valores de las distintas señales y puertos del circuito. Para poder asegurar el funcionamiento del microprocesador en su totalidad, era necesario poder comprobar la correcta realización de cada una de las instrucciones. Para ello el formato de pruebas propuesto fue realizar pequeñas rutinas, utilizando una pequeña cantidad de instrucciones por prueba, cargarlas manualmente en la memoria RAM, y analizar las señales internas y puertos de cada entidad con el paso de los ciclos.

RESULTADOS Y DISCUSIÓN

Problemas y Soluciones

Pese a que los bloques lograron pasar todas y cada una de las pruebas por separado, a la hora de analizarlos en conjunto comenzaron a aparecer errores de distinta naturaleza a lo largo de las diferentes pruebas. En su mayoría estos errores se debieron a problemas de sincronismo, de retardo, de tiempos de establecimiento, es decir, problemas temporales no contemplados por VUnit. Debido a la descripción de los bloques, y la falta de consideración de ciertos retardos, la actualización de ciertos valores se realizaba fuera del tiempo esperado. Se notó durante las pruebas realizadas una cierta similitud en las distintas fallas, llegando a un punto en el que alcanzando las últimas etapas del análisis se pudieron prever algunos errores y solucionarlos con la inclusión de bloques de retardo en los lugares críticos que ocasionaban las fallas. Siguiendo con la metodología de trabajo, se procedió con las pruebas del nuevo circuito con las diferentes herramientas de software nuevamente, tanto VUnit como ModelSim.

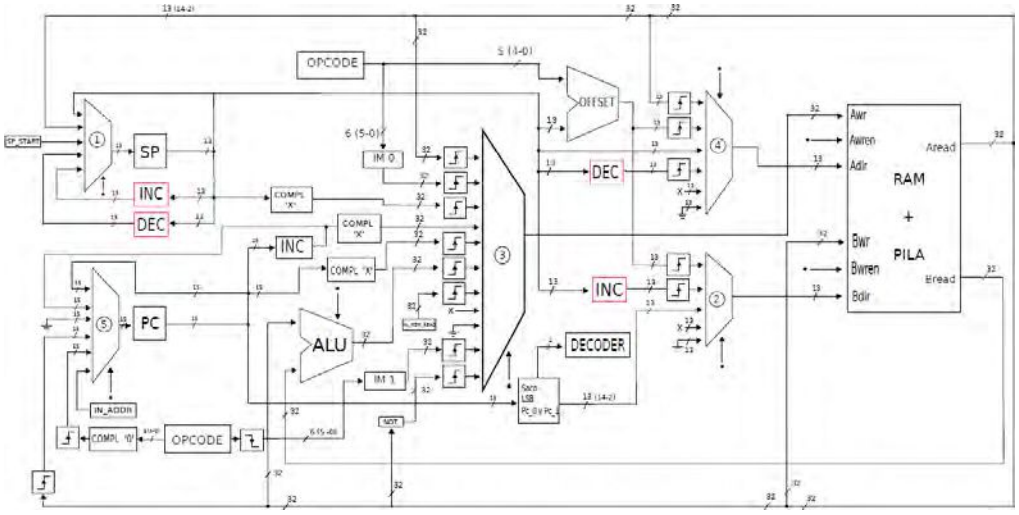

Figura 2. Diagrama de componentes final del diseño.

El *data path*, fue el bloque que más modificaciones sufrió, ya que fue el que más inconvenientes presentó en las simulaciones. No solamente fue necesario modificar las pruebas sobre la entidad total, sino también sobre los bloques internos que sufrieron las distintas variaciones. Estos cambios mencionados anteriormente, influyeron directamente sobre las palabras de control que debían ingresar al bloque a través de una de sus entradas. No solo se modificó el largo de la palabra por descartar el bit de control del bloque de carga inmediata, que fue dividido en dos circuitos concurrentes, sino que las palabras de control de los diferentes estados también sufrieron modificaciones.

Al igual que con el ZPU *Small*, para dar por finalizadas las pruebas sobre el diseño propio se decidió realizar una prueba en placa del procesador en funcionamiento. En base a los resultados obtenidos se observó que existen diferencias entre una descripción en VHDL simulable y una sintetizable. La discrepancia de estas características recae en la forma en la

que se puede describir un circuito y la forma en la que la FPGA lo sintetiza en base a los elementos que tiene a su alcance. La responsabilidad de solucionar estos errores entonces recae en el diseñador y su capacidad de crear un circuito que, utilizando los recursos que ofrece la FPGA, pueda sintetizar el circuito deseado.
Realizadas las correcciones correspondientes, el circuito final se vio drásticamente modificado, no solo el *data path* internamente, sino que sus puertos se vieron reformados al igual que los de la CU. Se puede observar en la Figura 2. los diseños finales de los circuitos.

RESULTADOS

Alcanzada esta etapa de trabajo solo restaba realizar las pruebas finales sobre el último circuito diseñado, como lo indica la metodología de trabajo. Las modificaciones a los bancos de prueba para las simulaciones por software correspondientes fueron realizadas tanto en VUnit como en Modelsim. Dichas simulaciones se llevaron a cabo no solo para cada bloque del circuito y para el circuito en su totalidad, sino que para una plataforma de ensayo creada para realizar las pruebas sobre el kit de desarrollo también. Con la misma metodología que en el caso del ZPU *Small* se decidió realizar la carga manualmente de diferentes programas pequeños, donde se obtuvieron resultados correctos. Sobre el kit de desarrollo se realizaron más pruebas de diferentes programas para verificar el funcionamiento de las distintas instrucciones realizadas, todas también con resultados exitosos.

CONCLUSIONES

En cuanto a la experiencia obtenida en el proceso de desarrollo, se puede resaltar la importancia en el orden de trabajo, así como también en la definición y el uso sostenido de la metodología de trabajo propuesta. Podemos asegurar que hacer uso de la técnica de modularización también facilita mucho este tipo de proyectos y la herramienta VUnit sirve para explotar al máximo esta técnica de trabajo [2][6][9]. La herramienta de Modelsim por su parte tiene mucho más provecho a la hora de hacer el análisis de casos particulares de funcionamiento. Por último es muy importante tomar en cuenta la diferencia entre los proyectos simulables y sintetizables, ya que esta es la clave para lograr la síntesis correcta de cualquier desarrollo en el cual se esté trabajando [4].
A la hora de pensar en el trabajo a futuro existen diferentes ramas por las cuales se puede continuar con el desarrollo de este proyecto y que abarcan básicamente, el lograr el funcionamiento de las diferentes herramientas de simulación y compilación, continuar con el diseño en microelectrónica de los diferentes módulos en base a bibliotecas de compuertas básicas y compatibilizar el diseño con la interfaz de comunicación llamada Wishbone, que es un protocolo de comunicación universal que permite comunicar cualquier desarrollo de hardware con diferentes módulos que posean la misma interfaz.

BIBLIOGRAFÍA

[1] ALTERA, *DE0 User Manual*. 2012.
[2] S. Chacon, B. Straub.*Pro Git*. 2014.
[3] E. O. Hwang, *Digital logic and microprocessor desing in VHDL*. 2015
[4] , R. Jasinski, *Efective code with VHDL. Principles and best practice*. 2016
[5] R.J, Tocci, *Sistemas digitales. Principios y aplicaciones*. 2010
[6] *VUnit documentation*, URL https://vunit.github.io/documentation.html
[7] E. Bozich, *Introducción a los dispositivos FPGA. Análisis y ejemplos de diseño*. 2005
[8] N. Sede, J. Serrangeli, M. Agostini, *Diseño de Circuitos en Microelectrónica*. 2018
[9] *Introducing GitFlow*. URL https://datasift.github.io/gitflow/IntroducingGitFlow.html