Adaptación en VHDL de un microcontrolador genérico para el soporte de algoritmos difusos

Martín Vázquez, Nelson Acosta y Daniel Simonelli
INCA / INTIA - Depto. Computación y Sistemas - Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Provincia de Buenos Aires - TANDIL
http://www.exa.unicen.edu.ar/inca
e-mail: {nacosta, mvazquez, dsimonel}@exa.unicen.edu.ar

1. Introducción

El enfoque con el que se diseñan los circuitos integrados ha tenido que ir evolucionando a través de varios niveles de abstracción. Del diseño geométrico del circuito y resolución de las ecuaciones diferenciales, se pasó al diseño con transistores ayudado con simulación eléctrica, para evolucionar posteriormente hacia el diseño lógico con la ayuda de editores de esquemas y simulación lógica y eléctrica. Finalmente se alcanza la actual metodología, basada en la descripción del comportamiento de los circuitos mediante lenguajes de descripción de hardware, simulación a nivel de comportamiento y utilización de herramientas de síntesis. Estas metodologías, conocidas como metodologías descendentes (*top-down*), han transformado los procedimientos de diseño de sistemas electrónicos, muy especialmente el de circuitos integrados. El lenguaje VHDL [Des95, Cha97] es su más claro exponente.

En cuanto al desarrollo de circuitos integrados, las diferentes alternativas de desarrollo se pueden agrupar cronológicamente en cuatro categorías: diseño totalmente a medida (*full-custom*), matrices de puertas predifundidas (*semi-custom / gate-arrays*), células estándar precaracterizadas (*semi-custom / standard cells*), y Lógica programable (FPGA, CPLD) [Bro94]. En estas categorías se ha bajado el costo de desarrollo y materialización, a expensas de una reducción de la velocidad de proceso.

Para diseñar sistemas electrónicos, la opción de combinar las tecnologías VHDL y FPGA es muy poderosa. El uso de VHDL permite reducir el tiempo de diseño y la utilización de FPGAs posibilita trabajar rápidamente con prototipos.

2. Objetivo Planteado

El objetivo de este trabajo es utilizar VHDL y FPGAs para definir microcontroladores dedicados a medida de la aplicación. Con este propósito, se recurre a la modificación de arquitecturas cores-IP[1] de microprocesadores de propósito general.

En este caso, se parte del microprocesador de propósito general DWARF (descrito en VHDL sintetizable sobre FPGA) y se realiza una modificación de dicha arquitectura para construir el microprocesador DWFUZ, especializado en controlar una familia de aplicaciones difusas [Cos95, Hun95, Rus98].

3. Metodologías descendentes

Las técnicas del diseño descendente se centran en promover el diseño a nivel comportamental, facilitando la evaluación de soluciones alternativas a partir de "diseños de alto nivel". La Fig. 1 muestra el esquema genérico de la metodología descendente apoyado por información ascendente [Ter98].

En la primera fase se definen las especificaciones de diseño, las cuales incluyen información del circuito y del entorno donde éste estará ubicado. A partir de esta información el proceso de descripción HDL contendrá dos modelos: el modelo que contiene las especificaciones funcionales

[1] Core-IP se refiere a un núcleo de diseño (core) que posee registro de propiedad intelectual (IP).

del circuito, normalmente en un nivel de abstracción comportamental y el modelo HDL que contiene el entorno del circuito para validarlo en su contexto (banco de pruebas, *test bench*).

Una vez depurados estos modelos (evaluación, simulación funcional, etc), se inicia el proceso de refinamiento gradual de la descripción del circuito mediante procesos automáticos de síntesis guiados por el diseñador. El refinamiento también puede afectar el banco de pruebas con el objetivo de ajustar su precisión a la del modelo del circuito.

En la primera fase de refinamiento gradual se aplica la síntesis comportamental para pasar de una descripción de nivel funcional a una de nivel RT. A continuación viene la etapa de síntesis RT-lógica que tiene por objeto la obtención de un esquema o lista de componentes y sus interconexiones (*netlist*) basado en la biblioteca escogida de células y módulos. El diseño resultante debe realizar la funcionalidad específica, respetar las prestaciones requeridas (área, consumo, velocidad) y garantizar la testabilidad del circuito. Se acostumbra a realizar varias iteraciones de los procesos automáticos de síntesis para la testabilidad del diseño.

Después de la síntesis RT-lógica, se obtiene un diseño más detallado y se puede abordar a las etapas del diseño físico. Se realiza los procesos de ubicación y conexionado para generar la topografía y descripciones necesarias para la implementación física del circuito. Hecho esto, se puede realizar extracción de información tecnológica (retardos, parámetros eléctricos), retroanotación, simulación *post-layout* , etc. Se itera en el proceso de diseño hasta que éste cumpla con las prestaciones funcionales y temporales requeridas.

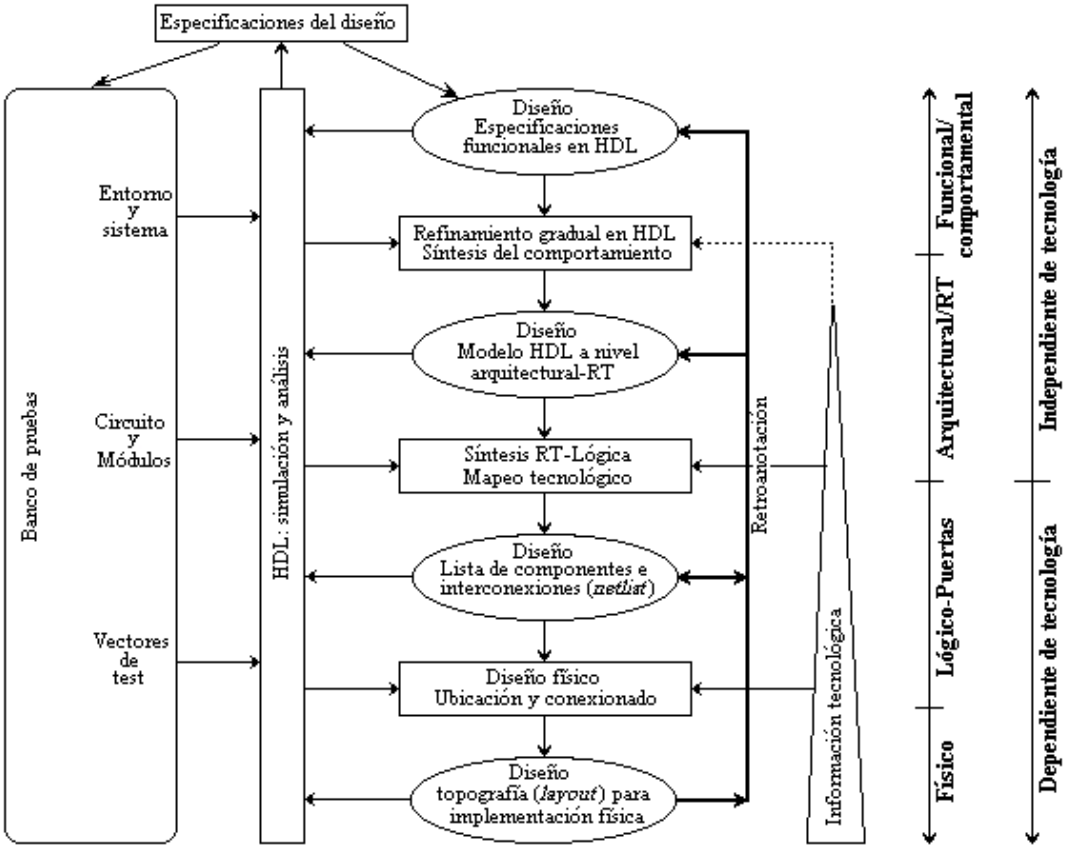

Fig. 1. Esquema genérico de las metodologías descendentes (*top-down*) [Ter98]

Se puede observar en la figura dos tipos de información ascendente, una es la información tecnológica (parámetros geométricos, eléctricos, retardos, células, etc), y la otra es la información que, a través de retroanotación, proviene de un nivel de abstracción inferior. Esta información, como ya se dijo, sirve para depurar en sucesivas iteraciones el diseño.

En las metodologías descendentes (*top-down*) se aplica el principio de abstracción, se trata de independizar el diseño de nivel funcional respecto del arquitectural-RT, y a su vez éste con el nivel lógico-puertas, siendo las herramientas de síntesis quienes marcan esta independencia. Las características de los HDLs facilitan y potencian el desarrollo de las metodologías de diseño descendentes, las cuales se basan en el uso intensivo de estos lenguajes (modelado/descripción de los circuitos y bancos de prueba), soportados durante todo el proceso de diseño por herramientas de simulación, síntesis y análisis-verificación.

La aplicación de VHDL en síntesis es una tecnología fundamental en las metodologías descendentes utilizadas industrialmente en la actualidad. La síntesis a partir de VHDL constituye hoy en día una de las principales aplicaciones del lenguaje. Las herramientas comerciales disponibles en la actualidad utilizan mucho a VHDL en síntesis RT-lógica.

Mnemotécnica	Nro. bytes	Codificación
load Rd	1	00100 d2 d1 d0
load (Rd)	1	00101 d2 d1 d0
load #d	2	01000000 d7 d6 d5 d4 d3 d2 d1 d0
load d	2	00110000 d7 d6 d5 d4 d3 d2 d1 d0
load (d)	2	00111000 d7 d6 d5 d4 d3 d2 d1 d0
store Rd	1	00000 d2 d1 d0
store (Rd)	1	00001 d2 d1 d0
store d	2	00010000 d7 d6 d5 d4 d3 d2 d1 d0
store (d)	2	00011000 d7 d6 d5 d4 d3 d2 d1 d0
in Rd	1	01100 d2 d1 d0
out	1	01101000
xor Rd	1	10000 d2 d1 d0
add Rd	1	10001 d2 d1 d0
test Rd	1	10010 d2 d1 d0
clear_c	1	10100000
set_c	1	10101000
jc #a	2	11000000 a7 a6 a5 a4 a3 a2 a1 a0
jz #a	2	11001000 a7 a6 a5 a4 a3 a2 a1 a0
jump #a	2	11010000 a7 a6 a5 a4 a3 a2 a1 a0
jsr #a	2	11011000 a7 a6 a5 a4 a3 a2 a1 a0
ret	1	11100000
reti	1	11101000

Tabla 1 - Codificación del conjunto de instrucciones del DWARF

4. Microprocesador DWARF

La arquitectura del microprocesador DWARF es la base de este trabajo. Esta arquitectura es conocida en el mundo académico y se considera relativamente simple. El hecho de que sea simple no implica que no posea ciertas características interesantes para modelar y describir. La arquitectura de este microprocesador es del tipo Von Neumann, la cual involucra el manejo de dos conceptos: el programa almacenado y ruptura de secuencia.

El microprocesador DWARF tiene las siguientes propiedades [Bou98]: a) La habilidad de manejar datos de ocho bits. b) Un banco de 8 registros de 8-bit. c) Direccionamiento indexado e indirecto. d) Puertos de entrada y salida. e) Instrucciones de llamada y retorno de subrutinas. f) Manejo de interrupciones.

El microprocesador posee una ruta de datos de ocho bits, de este modo, el *opcode* de la instrucción puede ocupar ocho bits. También pueden ocupar ocho bits los datos en modo inmediato. Varias instrucciones ocuparán dos bytes y por lo tanto el procesador deberá ejecutar dos *fetches* (obtenciones) para leerlas desde la memoria externa de programa. En la Tabla 1 se muestra la codificación de las instrucciones.

4.1. Modelo de la arquitectura

El modelo de la arquitectura del hardware del DWARF debe tener todo lo necesario para poder ejecutar las instrucciones discutidas en la sección anterior. Los componentes requeridos por el hardware son los siguientes: Contador de programa (PC, program counter), Registro de instrucción (IR, instruction register), Carry (C), zero (Z), Unidad aritmético lógica (ALU), Acumulador (ACC), Puntero a pila (SP), Banco de registros (register set, R0-R7), Puertos de entrada y salida, Selector de dirección {el microprocesador divide la memoria en tres regiones: PROGRAM (256 bytes), DATA (256 bytes) y STACK (256 bytes). El selector de dirección "habilita" (*enable*s) una de esas regiones y le envía la dirección a la correspondiente sección de memoria. La región de PROGRAM es direccionada por el PC, la región STACK es direccionada por SP y DATA por el valor que contenga uno de los registros, Detector de interrupción, Buses, Decodificador de instrucción (interpreta el código de la instrucción corriente, junto con el *flag* de carry (C), *flag* de zero (Z) y la información de interrupción).

Este procesador interactúa con una memoria externa, la cual como ya se dijo, está fraccionada en tres partes diferentes: PROGRAM, DATA y STACK. El tamaño de las distintas regiones es de 256x8, esto se debe a que la ruta de datos del procesador es de ocho bits. Al tener esta longitud, los operandos en las instrucciones son de ocho bits, lo que significa que los datos y las direcciones son de ocho bits.

5. Microprocesador DWFUZ

Las dos principales limitaciones que ofrece el DWARF para controlar aplicaciones difusas son el reducido espacio de memoria y el limitado conjunto de instrucciones. La memoria externa es de 1024x8, donde 256 bytes es el área de programa el cual es insuficiente para el controlador difuso. Por otro lado, las operaciones aritméticas *add*, *xor* y *test* limitan la eficiencia de los cálculos. Para solucionar estos inconvenientes, se añadieron nuevas funciones .La tabla 2 muestra la codificación de las instrucciones agregadas.

La funcionalidad agregada en DWFUZ es:

- **Inferencia difusa**. La inferencia difusa se realiza mediante el *método max-min*, para lo cual se agregaron las instrucciones aritméticas *max* y *min*.
- **Fusificación**. La estrategia de fusificación utilizada es la *orientada a memoria*, donde se obtienen los valores de pertenencia almacenados en ROMs externas.
- **Defusificación**. Se emplea la técnica del *centroide*, utilizando información precalculada almacenada en ROMs externas. Por otra parte, el cálculo de la división es muy costoso, para lo cual se agrega una instrucción especial.
- **División**. Utiliza registros especiales (dividendo DENDH, DENDM, y ACC 23-0 bits; divisor DSORH y A 15-0 bits; resto REM 24-0 bits); y además se agregan las instrucciones (`fdiv`) o indirecta (`loaddendm`, `loaddendh` y `loaddsorh`).
- **Memoria externa**. Se extendió la RAM externa de 1024x8 a 2048x8. Un tema a considerar, es que los saltos que en la RAM original se producen en el rango 0..255, ya que son 8 los bits utilizados para direccionar el salto. En la RAM extendida, se alteró al microprocesador para que interprete a la RAM extendida como 8 páginas de 256 bytes cada una, y se agregó el registro PAG (3-bit) para complementar la dirección de la RAM extendida. Debido a extensión de la memoria se deben modificar las instrucciones de salto y de manejo de interrupciones.

5.1. Modelo de la arquitectura del DWFUZ

Por definición, el modelo de esta arquitectura es una extensión del modelo de la arquitectura del DWARF. Los componentes agregados para DWFUZ son: a) Registros para la división

(DENDM, DENDH y DSORH), 8-bit. b) Registro resto (REM) 24-bit. c) Puntero a página (PAG) 3-bit. d) Puerto de salida1 de propósito general 8-bit.

Los componentes alterados respecto del DWARF son: a) ALU de 8 bits, pero maneja operandos de 16 y 24 bits. b) Banco de registros de 16 registros, modificado para posibilitar la carga de un bloque de 6 datos de 8 bits. c) Selector de dirección con manejo automático de la página. d) Bus se modifica el acceso a todos los nuevos registros. e) Decodificador de instrucción para tratar las nuevas instrucciones.

Mnemotécnica	Nro. bytes	Codificación
`min Rd`	1	11110 d2 d1 d0
`max Rd`	1	11111 d2 d1 d0
`inblock`	1	01001000
`loaddendm d`	2	10110000 d7 d6 d5 d4 d3 d2 d1 d0
`loaddendh d`	2	01011000 d7 d6 d5 d4 d3 d2 d1 d0
`loaddsorh d`	2	10111000 d7 d6 d5 d4 d3 d2 d1 d0
`fdiv Rd`	1	01010 d2 d1 d0
`jumpfar #p #a`	3	11010100 p7 p6 p5 p4 p3 p2 p1 p0 a7 a6 a5 a4 a3 a2 a1 a0
`jcfar #p #a`	3	11000100 p7 p6 p5 p4 p3 p2 p1 p0 a7 a6 a5 a4 a3 a2 a1 a0
`jzfar #p #a`	3	11001100 p7 p6 p5 p4 p3 p2 p1 p0 a7 a6 a5 a4 a3 a2 a1 a0
`jsrfar #p #a`	3	11011100 p7 p6 p5 p4 p3 p2 p1 p0 a7 a6 a5 a4 a3 a2 a1 a0
`out1`	1	01110000

Tabla 2 - Codificación de instrucciones agregadas en DWFUZ

5.2. Resultados obtenidos

Para tener información acerca del rendimiento y consumo del microprocesador DWFUZ, se efectuó el diseño físico sobre algunos dispositivos de la familia XC4000 de Xilinx [Xil99a, Xil99b]. En la Tabla 3 se aprecian algunos de los resultados obtenidos de la implementación de DWFUZ sobre diferentes dispositivos de la XC4000 .

Dada la cantidad de ciclos del programa de la aplicación difusa y la frecuencia máxima de funcionamiento, se calcula la velocidad de trabajo en FIPS (inferencias difusas por segundo) y FRPS (reglas difusas por segundo). Las Tablas 4 y 5 muestran las velocidades de ejecución del DWFUZ, sobre el dispositivo XC4020E-2-HQ208 y XC40250XV-7-BG432.

Dispositivo	Nro. CLBs	% CLBs	Nro. IOBs	Nro. IOBs	Máxima Frecuencia
XC4013E-2-PQ208	357 de 576	61	97 de 160	60	6.104 MHz
XC4020E-2-HQ208	357 de 784	45	97 de 160	60	6.458 MHz
XC4025E-2-CB228	357 de 1024	34	97 de 192	50	6.919 MHz
XC4036EX-2-BG352	357 de 1296	27	97 de 288	34	8.214 MHz
XC40250XV-7-BG432	357 de 8564	4	97 de 336	28	8.887 MHz

Tabla 3 - Resultados de la implementación del microprocesador DWFUZ

6. Conclusiones

Se diseñó un microprocesador especializado para controlar aplicaciones difusas, DWFUZ, basado en la modificación de la arquitectura (*core-IP*) de un microprocesador de propósito general conocido como DWARF. La transformación de la descripción del DWARF en DWFUZ fue relativamente sencilla. El diseño físico del microprocesador DWFUZ se realizó automáticamente sobre diferentes FPGAs pertenecientes a la familia XC4000 de Xilinx. Se obtuvieron métricas para cada dispositivo de área ocupada (CLBs, IOBs), frecuencia máxima de reloj, y velocidad real, etc.

Algunas de las ampliaciones más interesantes que pueden hacerse a este trabajo son: a) Implementación física del microcontrolador difuso completo para las distintas aplicaciones difusas. b) Implementar sobre familias de FPGAs Virtex o Spartan. c) Desarrollar una herramienta que genere automáticamente la descripción de los módulos de fusificación, defusificación, registros de E/S y el programa de aplicación dependiente de la aplicación difusa.

Sistema difuso	Cant. reglas	Cantidad de ciclos	FIPS	FRPS
Péndulo Invertido	49	1266	5101	249949
Camión	35	1040	6209	217315
Autoenfoque	17	1059	6098	103666

Tabla 4 - Velocidad de DWFUZ sobre XC4020E-2-HQ208 a 6.458 MHz

Sistema difuso	Cant. reglas	Cantidad de Ciclos	FIPS	FRPS
Péndulo Invertido	49	1266	7019	343931
Camión	35	1040	8545	299075
Autoenfoque	17	1059	8391	142647

Tabla 5 - Velocidad de DWFUZ sobre XC40250XV-7-BG432 a 8.887 MHz

7 - BIBLIOGRAFÍA

[Bou98]. "*The practical Xilinxs designer lab book*", Dave Van den Bout. Prentice-Hall, 1998.

[Bro94] "*Field-Programmable Gate Arrays*", S. Brown, R. Francis, J. Rose, Z. Vranesic. Kluwer Academic Publishers,1994.

[Cha97] "*Digital design and modeling with VHDL and synthesis*", K. C. Chang. IEEE, 1997.

[Cos95] "*Hardware Solutions for Fuzzy Control*", A. Costa, A. Costa, P. Faraboschi, A. Pagni, y G. Rizzotto. Proceeding of the IEEE , vol. 83, pág. 422-434, Marzo 1995.

[Des95] "*Descripción en VHDL y Simulación de Sistemas digitales*". Jean-Pierre Deschamps, ISISTAN, 1995.

[Hun95] "*Dedicated Digital Fuzzy Hardware*", Donald L. Hung. IEEE MICRO, vol. 15, pág. 31-39, Agosto 1995.

[Rus98] "*Fuzzy Hardware Architectures and Applications*", Marco Ruso. Kluwer Academic Publishers,1998. Editores Abraham Kandel y Gideon Langholz.

[Ter98] "*VHDL Lenguaje estándar de diseño electrónico*", L. Terés, Y. Torroja, S. Olcoz, E. Villar. Mcgraw-Hill, 1998.

[Xil99a] "*XC4000E and XC4000X Series Field Programmable Gate Arrays*". Xilinx, Mayo 1999. (Versión 1.6).

[Xil99b] "*Synthesis and Simulation Design Guide*". Xilinx, 1999..