DISEÑO DE UN FILTRO DIGITAL (IIR) CON MICROPROCESADOR DE ARQUITECTURA MULTICICLO EN FPGA

José Alberto Díaz García, Eugenio Salazar Brenes, Luis Quirós Rojas

Instituto Tecnológico de Costa Rica, Escuela de Ingeniería Electrónica, Cartago, Costa Rica

jdiaz@itcr.ac.cr, esalazar@ietec.org, lquiros@ietec.org

SUMARIO

El presente trabajo muestra el desarrollo de una arquitectura de un procesador multiciclo para propósito general, con una unidad lógico aritmética y otra de punto flotante, utilizado para implementar un filtro digital con respuesta infinita al impulso (IIR), del tipo paso bajo el cual se descargó en un FPGA Spartan™ 3. En el modelado del diseño se utilizó el lenguaje orientado a hardware Verilog. Este microprocesador es utilizado como material de apoyo en el curso EL-4311 Estructura de Microprocesadores, del plan de estudios en la Licenciatura en Ingeniería Electrónica.

1. INTRODUCCION

Este documento describe el diseño y ejecución de un filtro digital IIR, paso bajo, el cual utiliza una unidad de adquisición de datos basada en un microprocesador con una arquitectura multiciclo [1]. El concepto del sistema consiste en ejecutar un programa que simula un filtro digital almacenado en la memoria ROM utilizando el procesador desarrollado. La memoria RAM almacena las variables muestreadas por el convertidor de señales analógicas a digitales, así como las variables intermedias calculadas para generar la salida correspondiente. El proceso de cálculo se realiza mediante un algoritmo que aplica a la señal a filtrar un modelo de una ecuación en diferencias.

Los diferentes sistemas se implementaron utilizando el lenguaje de programación Verilog, el cual permitió realizar modificaciones en forma versátil en la arquitectura y simular algunas de sus características.

El presente corresponde a una parte del proyecto "Desarrollo de Arquitecturas para Microprocesadores", el cual se enmarca dentro de la literatura como Front End [2]; se pretende continuar el proceso hasta la etapa de Back End [2] para así obtener un circuito integrado.

2. ARQUITECTURA DEL SISTEMA COMPUTADOR

La Figura 1 muestra el sistema de adquisición y procesamiento de datos basado en un microprocesador multiciclo con instrucciones CISC en punto flotante. Este cuenta con un microprocesador de 16 bits, una memoria RAM de 32KWords y una memoria ROM de 4KWords, todo contenido en un FPGA Spartan™ 3, además de un convertidor A/D de 8 bits con un tiempo de conversión de 4,5 microsegundos y una frecuencia de muestreo de 10KHz y un convertidor de D/A de 8 bits ambos contenidos en el modulo Digilent™ AIO1.

Las memorias, así como los convertidores se dispusieron externamente al microprocesador. El bus de datos es bidireccional y común entre las unidades, además el procesador es el encargado de generar las señales de control necesarias para el acceso a los diferentes módulos, en ciclos de lectura y escritura.

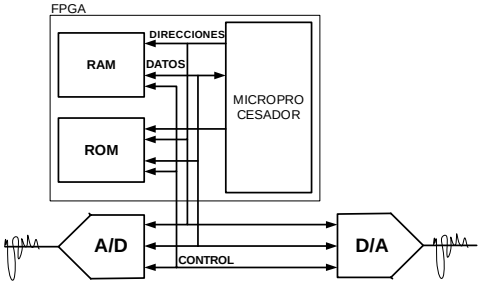

Fig. 1. Diagrama de bloques del sistema

3. ARQUITECTURA DEL SISTEMA MICROPROCESADOR

El microprocesador para esta aplicación se muestra simplificado en la Figura 2. Las unidades con que cuenta la arquitectura son: un banco de 64 registros, una unidad lógico aritmética (ALU) con su correspondiente registro de banderas, un acumulador para almacenar temporalmente los resultados (ACC), dos registros temporales a la

entradas de la ALU (RA y RB), una unidad de punto flotante (UPF), un contador de programa (PC) para el acceso a la memoria principal, y una unidad de control para la decodificación de las instrucciones (CU), el registro DIRR se utiliza con el fin de evitar colisiones entre la información proveniente de la memoria principal (RAM, ROM) y datos que transiten por el bus interno, un buffer ubicado a la salida del PC, es utilizado para establecer una ruta para que el contenido del PC se respalde en el banco de registros sin riesgos de colisión.

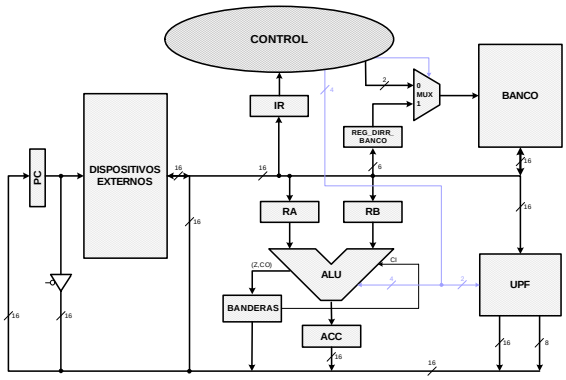

Fig. 2. Arquitectura básica del microprocesador multiciclo

La memoria principal está formada por 32KWords de memoria RAM ubicados a partir de la dirección 7FFFH hasta la FFFFH y 4KWords de memoria ROM, ubicados de la dirección 0000H a la 0FFFH.

3.1. Unidad de Control (CU)

Para el diseño de la CU se utilizó el modelo de Mealy [3] en la decodificación del código de operación de las instrucciones. En la Figura 3 se muestra su diagrama de bloques, este diseño permite manejar eficientemente una gran cantidad de estados y además el desarrollo de nuevas instrucciones.

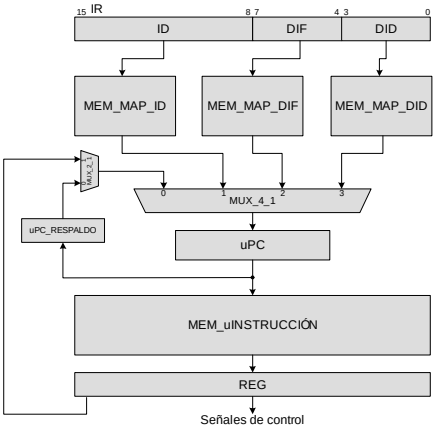

Fig. 3. Unidad de control

3.2. Unidad de Punto Flotante (UPF)

Debido a que el algoritmo a implementar utiliza coeficientes no enteros se hizo necesario el uso de una unidad de punto flotante, para conservar la precisión en todos los cálculos.

Para el diseño de la UPF se utilizó el formato simple del estándar IEEE 754. Esta unidad modifica las banderas necesarias para su operación. La UPF realiza las funciones mostradas en la Tabla 1.

Función	Código	Descripción
Suma	0000	Resultado = $(-1)^{s1}2^{e1-127}(1*f1)$ + $(-1)^{s2}2^{e2-127}(1*f2)$
Resta	0001	Resultado = $(-1)^{s1}2^{e1-127}(1*f1)$ - $(-1)^{s2}2^{e2-127}(1*f2)$
Multiplicación	0010	Resultado = $(-1)^{s1}2^{e1-127}(1*f1)$ * $(-1)^{s2}2^{e2-127}(1*f2)$
División	0011	Resultado = $(-1)^{s1}2^{e1-127}(1*f1)$ / $(-1)^{s2}2^{e2-127}(1*f2)$

Tabla 1. Funciones de la UPF (nomenclatura según estándar)

3.3. Unidad Lógico Aritmética (ALU)

La ALU es de 16 bits con capacidad para realizar las funciones mostradas en la Tabla 2.

Ésta modifica el registro de banderas cada vez que realiza una operación, aritmética o lógica. El manejo de los números negativos se da en complemento a dos. En el registro de banderas hay indicaciones para examinar si el resultado es cero, si se producen acarreos, o si el número es negativo.

Función	Código	Descripción
Suma	0000	Acc = A + B
Resta	0001	Acc = A - B
And	0010	Acc = AB
Or	0011	Acc = A + B
Nand	0100	Acc = !(AB)
Nor	0101	Acc = !(A+B)
Xor	0110	Acc = A ⊕ B
Xnor	0111	Acc = !(A ⊕ B)
Not	1000	Acc = !A
Slr	1001	Acc = A < B
Sll	1010	Acc = A > B
Pass a	1011	Acc = A
Pass b	1100	Acc = B

Tabla 2. Funciones de la ALU

3.4. Banco de Registros (BR)

El banco de registros cuenta con 64 registros de 16 bits cada uno. Los primeros cuatro registros (direcciones del 0 al 3) son utilizados por el procesador en caso de que sea necesario (se acceden únicamente por la CU), los restantes son para propósito general, los cuales se acceden en forma independiente por medio del código de operación.

4. DISEÑO DE INSTRUCCIONES

El largo de las instrucciones se dispuso variable y pueden ser de una, dos o tres palabras de ancho, dependiendo del modo de direccionamiento que se utilice.

La sintaxis de las instrucciones se muestra en la Figura 4, donde el nemotécnico representa el código de operación de la instrucción, la fuente donde se toman y el destino el lugar donde se almacenan los datos, todos los campos son de 16 bits.

Fig. 4. Sintaxis de las instrucciones

Los modos de direccionamiento se muestran en la Tabla 3.

Código	Descripción	Modo
0000	Dato	Inmediato
0001	Registro	Directo
0010	((Dirección))	
0011	(Registro)	Indirecto
0100	(Dirección)	
0101	(Registro + constante)	Indexado y
0110	(Registro + Registro)	Stack
0111	(Registro + (Dirección))	Registro = SP
1000	(Registro + (Registro))	(X)
1001	(PC + Dirección)	Relativo

Tabla 3. Modos de direccionamiento

Cada código de operación se divide en tres partes: ID, DID y DIF, los bits asignados a cada una se muestran en la Figura 5.

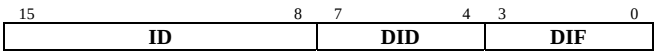

Fig. 5. Partes del código de operación de las instrucciones

El ID (bits del 8 al 15) es el identificador de la instrucción. Las instrucciones disponibles son de los tipos: aritméticas o lógicas, de control o de movimiento de datos. La Tabla 4 muestra el código de algunos identificadores de las instrucciones.

El DID (bits del 4 al 7) y el DIF (bits del 0 al 3) indican los tipos de direccionamiento que tiene el operando destino y fuente, correspondientemente. En la Tabla 3 se muestran los códigos asignados a cada uno de los modos de direccionamiento.

Código	Nemotécnico	Descripción
00H	MOV fuente, destino	Destino ← fuente
01H	JUMP dirr	PC ← dirr
02H	IN puerto, destino	destino←[puerto]
03H	OUT fuente, puerto	[puerto]←fuente
04H	ADDPF	Registro 3←[Parte alta del resultado de la UPF]
05H	SUBPF	
06H	MULTPF	Registro 4←[Parte baja del resultado de la UPF]
07H	DIVPF	
08H	WAIT	Ciclo de espera de 15 ciclos de reloj

Tabla 4. Tipos de identificadores

5. CÁLCULO DEL ALGORITMO PARA EL FILTRO DIGITAL PASO BAJO

Para el diseño del filtro digital se utilizó el modelo de Butterworth de segundo orden, luego se aplicó la transformada Z con el fin de obtener la transformada Z inversa (Z^{-1}) y así contar con un modelo en ecuación de diferencias, el cual corresponde al algoritmo a programar [4].

Las especificaciones de diseño del filtro son: frecuencia de corte (frecuencia de corte digital, ωd) de 2KHz y una frecuencia de muestreo (fc) de 10KHz.

Para la transformación al dominio Z se utiliza la transformación bilineal. El modelo de la salida del filtro se aplica la transformada Z inversa y se resuelve para la salida C(N) tal como se muestra en el modelo 1.

$$C(N) = 0.36953C(N-1) - 0.1958C(N-2) + 0.20657R(N) + 0.4131R(N-1) + 0.20657R(N-2) \quad (\mathbf{1})$$

Donde R(N-T) corresponde al término de entrada muestreado en el tiempo de muestreo T y los C(N-T) al término de salida calculado para el tiempo de muestreo T.

6. PROGRAMACIÓN DEL FILTRO DIGITAL

Para diseñar el algoritmo del filtro (modelo 1) se utilizó el conjunto de instrucciones de la Tabla 5.

Los datos de entrada y salida son almacenados en la memoria RAM, los cuales se manejan como si fueran un arreglo de una pila (stack). En el arreglo se encuentran los valores de entrada pasados y presente así como las salidas pasadas y presente, los cuales son actualizados cada vez que se genere un nuevo valor de salida y de entrada.

El programa escrito en lenguaje ensamblador se compiló utilizando los códigos asignados a las instrucciones (Tabla 4). El programa compilado se almacena en una memoria tipo ROM.

7. IMPLEMENTACIÓN DEL SISTEMA

Para modelar las instrucciones se utilizó el nivel RTL [5].

Para la implementación de las memorias (RAM y ROM) y la CU se utilizaron bloques RAM sincrónicos disponibles en la FPGA Spartan™ 3, como el RAMB16_S36. En la ROM y en CU fue necesario inicializar los bloques RAM utilizados, esto mediante un archivo NCF.

El módulo que realiza las funciones de la ALU es del tipo combinacional. Durante el proceso de síntesis se infiere el uso de los recursos del FPGA como los son sumadores/restadores, desplazadores y multiplexores.

Una de las especificaciones para el diseño de la UPF es que se ejecuten las operaciones en la menor cantidad de ciclos de reloj, por ejemplo se utilizó un bloque multiplicador en forma combinacional (el MULT18X18) empotrado en el FPGA Spartan™ 3.

Para la implementación de los sistemas de la arquitectura desarrollados en Verilog, se utilizó la herramienta ISE™ V7.1i de Xilinx™ para el proceso de síntesis, que en conjunto de la ayuda de ModelSim™ V6.0d de Mentor Graphics™ se depuraron los módulos.

Una vez finalizado el proceso de síntesis se obtuvieron los resultados resumidos, que se muestran Figura 7. De donde se pueden observar los distintos porcentajes de los recursos utilizados de la FPGA, así como información adicional referente a los tiempos de ejecución. Una vez depurado el sistema, este se descargó en el FPGA Spartan™ 3 utilizando la herramienta iMPACT™ V7.1i de Xilinx™.

```
=====================================================================
*                           Final Report                              *
=====================================================================

Device utilization summary:
---------------------------

Selected Device : 3s1000ft256-4

 Number of Slices:                      384  out of   7680     5%
 Number of Slice Flip Flops:            240  out of  15360     1%
 Number of 4 input LUTs:                603  out of  15360     3%
 Number of bonded IOBs:                  37  out of    173    21%
 Number of MULT18X18s:                    1  out of     24     4%
 Number of GCLKs:                         1  out of      8    12%

=====================================================================
TIMING REPORT

Clock Information:
------------------
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
CLK                                | BUFGP                  | 272   |
-----------------------------------+------------------------+-------+

Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 13.821ns (Maximum Frequency: 72.353MHz)
   Minimum input arrival time before clock: 6.527ns
   Maximum output required time after clock: 16.168ns
   Maximum combinational path delay: No path found
=====================================================================
```

Fig. 7. Reporte final de síntesis provisto por Project Navigator™ V7.1i

8. CONCLUSIONES

El procedimiento desarrollado para la implementación de instrucciones para la arquitectura del procesador multiciclo es un recurso didáctico muy valioso que se puede utilizar en los cursos referentes a estructuras de microprocesadores.

El diseño de la unidad de control utilizado permite aumentar la cantidad de instrucciones que se pueden ejecutar en el sistema, modificando los contenidos de los bloques de memoria RAM utilizados en este módulo.

La implementación de una unidad de punto flotante en esta arquitectura permite mantener precisión en el manejo de los coeficientes de la ecuación en diferencias que realiza al filtro digital, lo cual permite realizar instrucciones más complejas.

Debido al uso de los convertidores del módulo Digilent™ AIO1 la precisión de la señal resultante se reduce.

La utilización de bloques invocados propios del Spartan™ 3 (como el MULT18X18 y RAMB16_S36) reduce el tiempo de retardo y las rutas de conexión respecto a los bloques inferidos, lo cual permitió optimizar de los recursos de la FPGA utilizada, esto se refleja en la frecuencia de operación obtenida.

9. REFERENCIAS

[1] Patterson David A. y Hennessy John L., Computer Organization and Design, The Hardware / Software Interfase, Morgan Kaufmann Publishers, San Francisco, California, USA., pp. 284-353, 2005.

[2] Kaeslin, Hubert., Digital Design Flow, Swiss Federal Institute of Technology, Zurich. Accedido Enero 10, 2006, Microelectronics Design Center, http://dz.ee.ethz.ch/support/ic/digitalflow/index.en.html

[3] Wakerly John F., Digital Design: Principles and Practices, Prentice Hall, USA, pp. 23-54, 2000.

[4] Ogata Katsunhiko, Sistemas de Control en Tiempo Discreto, Prentice Hall, México, pp. 23-54, 1996.

[5] Palnitkar Samir, Verilog HDL, A Guide to Digital Design and Synthesis, SunSoft Press, A Prentice Hall Title, USA, pp. 215- 359, 2003.