IMPLEMENTACIÓN EN HARDWARE DE UN SVPWM EN UN SOFT-CORE NIOS II. PARTE I

HARDWARE IMPLEMENTATION OF A SVPWM ON A NIOS II SOFT-CORE. 1ST PART.

J. J. Raygoza P.[1], Susana. Ortega C.[1], Carlos A. Chirino G. [1], J. Rivera D. [1]
juan.raygoza@cucei.udg.mx / susana.ortega@cucei.udg.mx / carlos.chirino@red.cucei.udg.mx / jorge.rivera@cucei.udg.mx

Recibido: 03 marzo, 2009 / Aceptado: 28 octubre 2009 / Publicado: 31 diciembre, 2009

RESUMEN. Este artículo expone la implementación en hardware de una modulación por ancho de pulsos en un espacio vectorial en un soft-core embebido Nios II. La utilización de dispositivos reconfigurables otorga flexibilidad en el diseño y otras mejoras en términos de consumo de potencia con la ayuda de herramientas de software que permiten hacer más eficientes los algoritmos. Se presenta la configuración de un procesador embebido Nios II en su modo estándar implementado en hardware sobre una FPGA Cyclone II de la familia de Altera. Una de las ventajas del soft-core embebido Nios II es que podemos efectuar cálculos que contengan números del tipo flotante. Esto nos sirve para realizar las operaciones trigonométricas requeridas por el algoritmo de modulación por ancho de pulsos en un espacio vectorial. El algoritmo es descrito en lenguaje C++ mediante una aplicación software también de la familia de Altera. La solución nos brinda buena precisión en los cálculos matemáticos. Entre los resultados obtenidos se muestra la gráfica de ocupación del dispositivo Cyclone II para la implementación del soft-core embebido, así como tabla de tiempos medidos en simulaciones con modelsim y tabla de valores correspondientes a las seis salidas del SVPWM realizado en el Nios II. Dichos valores fueron leídos con ayuda de un analizador lógico en tiempo real.

PALABRAS CLAVE: FPGA, Procesador embebido, Operaciones en punto flotante, Cyclone II, Altera.

ABSTRACT. This paper presents the hardware implementation of a Space Vector Pulse Width Modulation on an embedded Nios II Soft-core. The utilization of reconfigurable devices grants flexibility in the design and other improvements in terms of power consumption with the help of software tools that allow us to do more efficient our algorithms. We expose a configuration of a Nios II embedded processor in standard mode, implemented in hardware on a Cyclone II FPGA of the Altera family. One of the advantages of using the Nios II embedded soft-core is that we can do calculus with floating point numbers. This is useful to do the trigonometric operations needed by the Space Vector Pulse Width Modulation algorithm. The algorithm is written in C++ language using a software application of the Altera family. With this solution we achieve precision in the mathematics calculations. Among the obtained results, we show a graphic of occupation of the Cyclone II device used to implement the embedded soft-core. Also we present a table of times that were measured in simulations by the modelsim tool and a table of values of the six out ports of the SVPWM implemented in the Nios II. This values were obtained with a logic analyzer in real time.

KEYWORDS: FPGA, Embedded processor, Floating point operations, Cyclone II, Altera.

Introducción

La creación de software para desarrollo de sistemas embebidos revolucionó el uso de las FPGAs. La utilización de dispositivos reconfigurables otorga flexibilidad en el diseño y grandes beneficios para el desarrollo de aplicaciones en estos dispositivos. Gracias al aumento en capacidad de integración de las FPGAs se creó software para desarrollar sistemas de procesamiento embebidos. Como resultado existen procesadores soft y hard-core para las FPGAs. Los procesadores soft-core pueden ser descritos en un lenguaje HDL por el diseñador o generados mediante una específica herramienta de diseño embebido como el Xilinx EDK o el SOPC Builder de Altera. El desarrollo de aplicaciones para sistemas embebidos implica abordar bastantes elementos además de la propia aplicación, tal como el procesador y su posible configuración, los periféricos, la memoria, entre otros. En el caso de los procesadores soft-core, es posible hacer la descripción sintetizable del procesador mediante un lenguaje de descripción HDL que define la

[1] Centro Universitario de Ciencias Exactas e Ingenierías (CUCEI), Blvd. Marcelino García Barragán #1421, Guadalajara, Jalisco, 44430, México - www.cucei.udg.mx

lógica que lo implementará. Esto tiene como principal ventaja la flexibilidad, ya que ciertos parámetros pueden ser configurados, como el tamaño de memoria cache o el uso de multiplicadores en HW, tal como ocurre en el soft-core del microblaze de Xilinx [1] o el Nios II de Altera.

El procesador Nios II, basado en un CPU de uso general, es un procesador de arquitectura RISC de 32 bits diseñado específicamente para los dispositivos lógicos programables de Altera [2]. Como se muestra en la **Figura 1,** entre las características fundamentales de este procesador destaca que posee un grupo de registros de propósito general de 32 bits. Otra de sus facilidades es que proporciona el acceso a múltiples periféricos dentro del propio chip e interfaces para memorias y periféricos externos, así como la posibilidad de incorporar la multiplicación y el *barrel shifter* por hardware, lo cual ofrece mayores prestaciones (hasta 200 DMIPS) y mayor eficiencia que otros procesadores.

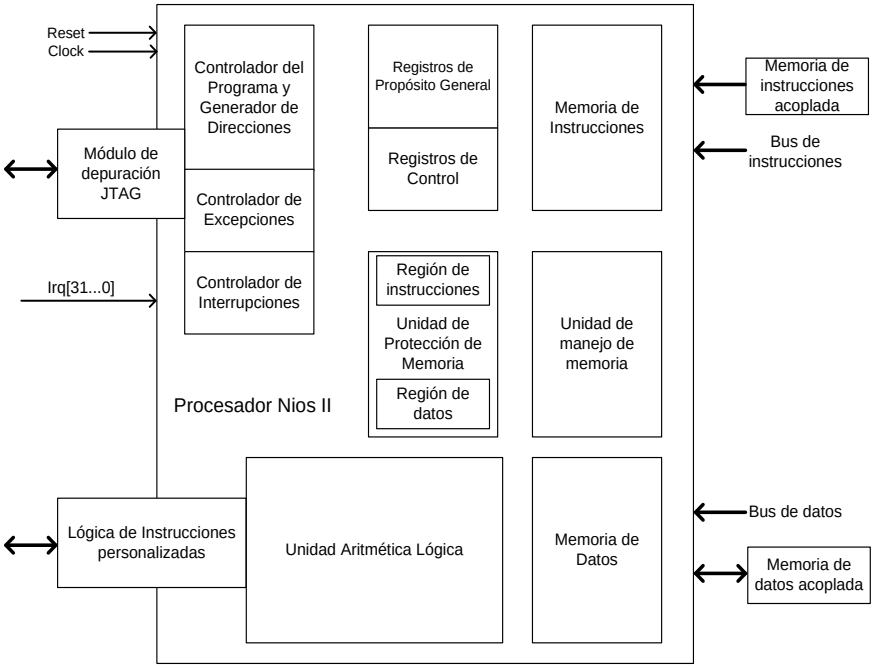


Figura 1. Procesador Nios II

Existen tres versiones del procesador Nios II y todas ellas comparten la misma arquitectura de instrucciones de 32 bits:

- Nios II /f (rápido), con las mayores prestaciones y uso moderado de lógica.
- Nios II /s (estándar), con altas prestaciones y poco uso de lógica.
- Nios II /e (económico), el menor uso de lógica y de bajo costo.

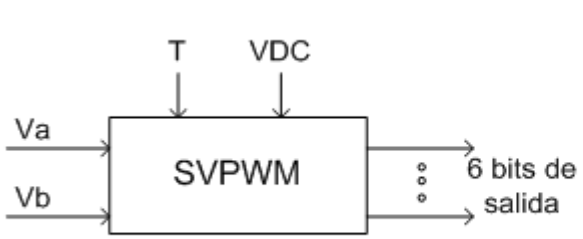
Ya que el procesador Nios II es un procesador soft-core que sólo se implementa en dispositivos FPGAs de Altera, utilizamos la serie Cyclone. El dispositivo FPGA Cyclone II soporta todas las versiones de la familia Nios II. Estos dispositivos ofrecen una solución para aplicaciones que requieren de gran capacidad de integración con características necesarias para aplicaciones como memoria incorporada, interfaces para memoria externa y circuitos de control de reloj. La FPGA Cyclone II amplía la densidad lógica de la serie a más de 68000 elementos lógicos, además de ofrecer mayor funcionalidad con multiplicadores incorporados, soporte para una interfaz de memoria externa o posibilidad de integrar nuevas interfaces de entrada y salida. Además la Cyclone II puede llevar varios procesadores Nios II en un solo dispositivo, para mayor rendimiento del sistema, menor consumo de potencia y mayor ahorro en recursos.

Conjuntamente con el dispositivo FPGA, para la implementación del Nios II es necesario el software que proporciona Altera para configurar el procesador. El Quartus II [3] es un software con una serie muy completa de herramientas de síntesis, optimización y verificación de un entorno en diseños HDL. Entre sus utilidades se encuentra el SOPC Builder, una poderosa herramienta de diseño de sistemas que nos permite definir y generar un sistema completo en un dispositivo programable. Es una herramienta de propósito general para la creación de sistemas que pueden o no contener un procesador. El SOPC Builder automatiza la tarea de integrar componentes de hardware generando la interconexión de estos mediante archivos HDL que definen cada componente del sistema y un archivo HDL top-level que se encarga de conectar todos los componentes.

Implementación.

El Algoritmo

La modulación SVPWM se basa en la sintetización de un vector deseado delimitado por los valores Va y Vb, como se muestra en la **Figura 2.**



$$V = \sqrt{Va^2 + Vb^2}$$
$$teta = \tan^{-1}\left(Vb/Va\right)$$
$$T1 = \sqrt{3}T\,\frac{V}{VDC}\sin\left(1.0471 - teta\right)$$
$$T2 = \sqrt{3}T\,\frac{V}{VDC}\sin\left(teta\right)$$
$$T0 = T - T1 - T2$$

Figura 2. Bloque del algoritmo SVPWM y ecuaciones a implementar.

El Hardware

El diseño del procesador que soportará la aplicación se realizó sobre una plataforma de prototipado DE2 Development and Education Board [4], que incluye una FPGA Cyclone II EP2C35. Sobre esta plataforma se diseñó el procesador Nios II con la configuración denominada estándar caracterizada por su sencillez, ya que no utiliza multiplicadores en HW ni predicción de saltos o memoria caché innecesarios. Esta configuración es óptima para aplicaciones de rendimiento medio y con grandes cantidades de código o procesamiento de datos. El software a utilizar es el Quartus II 8.1 Web Edition de Altera, con el que configuramos el procesador Nios II y sus periféricos a través de la herramienta SOPC Builder [5]. Además, asignamos los pines correspondientes a la FPGA Cyclone II y hacemos síntesis y simulación del hardware embebido.

El reloj del sistema es de 50 MHz, un periférico de entrada interfaz uart (uart1) y un puerto de salida de datos de 8 bits (out_port). La memoria de instrucciones la elegimos de 4Kbytes; no tenemos memoria de datos, por lo que utilizamos una memoria On Chip de 32Kbytes para lo que se pueda necesitar.

Asimismo, en el procesador Nios II configuramos el uso de la unidad de punto flotante [6] para implementar operaciones aritméticas de precisión en punto flotante, ya que podemos utilizar instrucciones definidas [7] para hacer mas rápidas las operaciones aritméticas en el programa de la aplicación en C/C++ del Nios II. De esta manera, el sistema quedó configurado como se muestra en la **Figura 3** para poder utilizar los recursos de hardware que están predefinidos y facilitar el diseño de la aplicación en software.

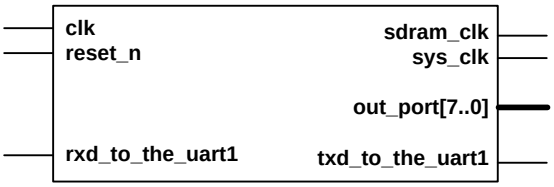


Figura 3. Bloque esquemático del sistema.

Realizada la síntesis del sistema, mediante las herramientas del software Quartus II obtenemos el reporte de ocupaciones mostrado en la **Tabla 1.** La utilización de elementos lógicos de la tarjeta Cyclone II es sobrada, debido a que utilizamos un procesador Nios II en su configuración estándar con pocos periféricos de entrada y salida y una memoria de tamaño medio.

Tabla 1. Ocupación de la FPGA Cyclone II EP2C35F672C6 con un sistema Nios II estándar.

Cyclone II	*Utilizados*	*Disponibles*	*%*
Logic Elements	4752	33,216	14.31%
Total pins	63	475	13.26%
Total memory bits	31768	483,84	6.57%
Embedded Multipliers	11	35	31.43%
PLL's	1	4	25.00%

El software

Se utilizó el software Nios II C/C++ IDE para implementar el software que se cargará al sistema [8]. El algoritmo resuelve ecuaciones de control a partir de 2 voltajes de entrada medidos por el puerto uart del sistema. Con este par de voltajes, a través de las ecuaciones mostradas en la **Figura 2**, calculamos tres intervalos de tiempo (*T0*, *T1* y *T2*) y un ángulo teta. El valor de *T* es de 0.001s. Posteriormente, según el valor del ángulo teta, se realiza un código para que el puerto de salida muestre un dato de 6 bits por periodos acordes con los tres tiempos calculados.

Después de compilar el código en C++, realizamos la simulación del hardware y software mediante la herramienta Modelsim. Se ejecutaron simulaciones con diferentes valores de voltajes de entrada, una de ellas se muestra en la **Figura 4**. Después de varias simulaciones, obtuvimos la **Tabla 2**, en donde podemos observar que, conforme a los datos de entrada de los dos voltajes, las salidas correspondientes en valor y duración son correctas.

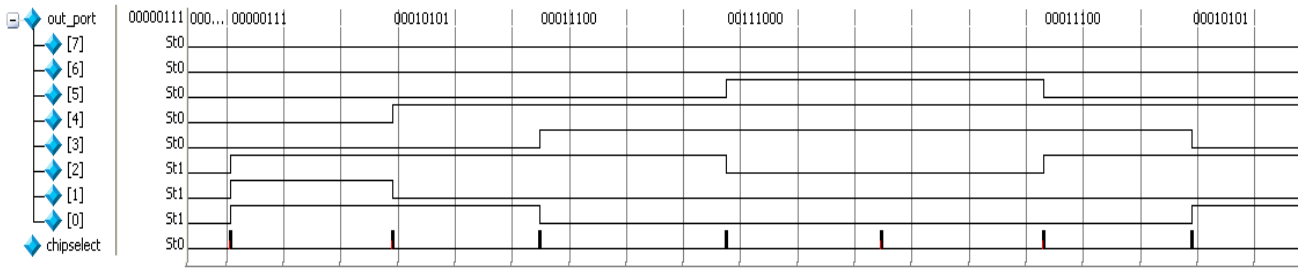


Figura 4 Simulación Modelsim, en el cual los valores de entrada son Va=-55.615V y Vb=171.193V

Resultados

Tabla 2. Resultados recopilados de las simulaciones en Modelsim de la aplicación en software.

Va (V)	*Vb(V)*	*Teta(rad)*	*Puerto de Salida*							
			T0 s	*T1 s*	*T2 s*	*T0 s*	*T0 s*	*T2 s*	*T1 s*	*T0 s*
131.21	123.21	0.7539	000111	100011	110001	111000	111000	110001	100011	000111
11.30	179.64	1.5079	000111	110001	010101	111000	111000	010101	110001	000111
-55.615	171.19	1.8849	000111	110001	010101	111000	111000	010101	110001	000111
-114.72	138.69	2.2618	000111	010101	011100	111000	111000	011100	010101	000111
-178.57	22.57	3.0158	000111	010101	011100	111000	111000	011100	010101	000111
-174.34	-44.74	3.3928	000111	011100	001110	111000	111000	001110	011100	000111
-145.63	-105.78	3.7698	000111	011100	001110	111000	111000	001110	011100	000111
-33.74	-176.80	4.5237	000111	001110	101010	111000	111000	101010	001110	000111
33.70	-176.81	4.9007	000111	001110	101010	111000	111000	101010	001110	000111
145.60	-105.82	5.6547	000111	101010	100011	111000	111000	100011	101010	000111

En el código en C++ se utilizan variables de tipo flotante de 16 bits y se realizan operaciones trigonométricas con precisión de punto flotante con la ayuda de las instrucciones predefinidas. Al realizar las simulaciones mediante las herramientas del software Modelsim, logramos medir los tiempos *T0*, *T1* y *T2*, calculados para los voltajes de entrada elegidos como muestra.

En la **Tabla 3** se presentan los tiempos que se deben obtener según los voltajes de entrada que se miden mediante el uart. Como se observa, los valores de tiempos calculados son suficientemente semejantes con los valores esperados. Con la utilización de las instrucciones de precisión de punto flotante, logramos que la pérdida de información sea insignificante y así obtener los resultados lo más exactos posible. Podemos analizarlo en la **Figura 5** donde se presenta la gráfica de error entre el valor esperado y el valor obtenido.

Tabla 3. Comparativa de tiempos esperados y calculados por el algoritmo realizado en el Nios II

Voltajes de Simulación		*Valores de tiempos esperados*			*Valores de tiempos obtenidos*		
Ua(V)	*Ub(V)*	*T1(s)*	*T2(s)*	*T0(s)*	*T1(s)*	*T2(s)*	*T0(s)*
131.21	123.21	0.000333658	0.000790432	0.000124089	0.000325178	0.000780362	0.000121194
113.08	179.64	0.000513484	0.001152420	0.000361064	0.000496378	0.001128772	0.000349818
-55.61	171.19	0.000858150	0.001098200	0.000759948	0.000859914	0.001109842	0.000755440
-114.72	138.69	0.001082300	0.000889753	0.001192540	0.001035208	0.001086558	0.001142954
-178.57	22.57	0.001064470	0.000144806	0.001919660	0.001119844	0.000154198	0.001920850
-174.34	-44.74	0.000824990	0.000287070	0.002112060	0.000795084	0.000280258	0.002039598
-145.63	-105.78	0.000469653	0.000678630	0.002148280	0.000438928	0.000642904	0.002012424
-33.74	-176.80	0.000379729	0.001134220	0.001754500	0.000391926	0.001185442	0.001814522
33.70	-176.81	0.000754486	0.001134270	0.001379790	0.000740566	0.001127442	0.001357584
145.60	-105.82	0.001148370	0.000678844	0.000530474	0.001177274	0.000704608	0.000544802

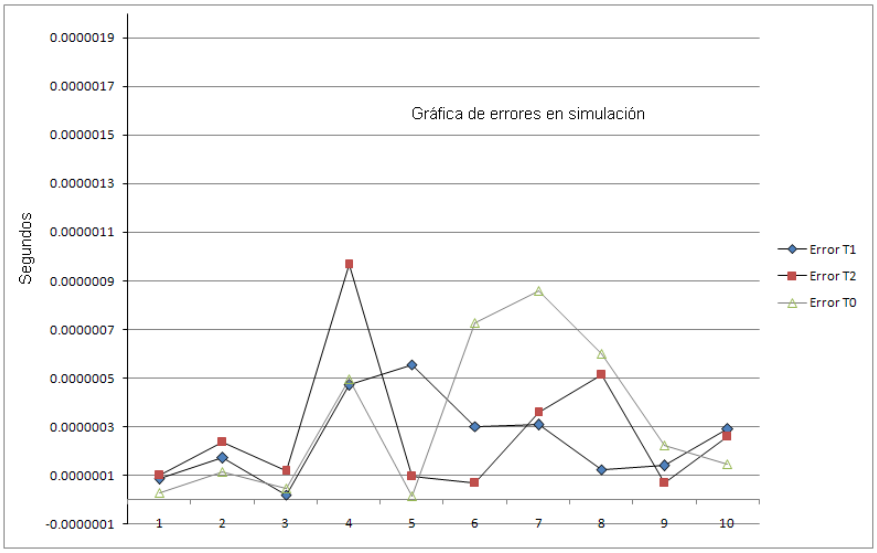


Figura 5. Gráfica de error entre el tiempo esperado y el tiempo obtenido en las simulaciones.

Conclusiones

La configuración del Nios II en su versión estándar para esta aplicación concedió un procesador sencillo pero a la vez útil en rendimiento, necesario para utilizar instrucciones definidas para realizar operaciones aritméticas de punto flotante de forma más rápida. Con esto, se lograron resultados con poca pérdida de información en el procesamiento de las ecuaciones del algoritmo y el cálculo de los tiempos fue muy exacto.

La ocupación de recursos lógicos de la FPGA fue del 14.3% de los elementos lógicos, lo que implica la posibilidad de aumentar las tareas del procesador con otros algoritmos o incluso agregar en el mismo dispositivo más procesadores para elevar el rendimiento del sistema. Se pretende continuar con el trabajo optimizando el código en lenguaje C++ y buscar que el control que se llevó a cabo se pueda implementar bajo un algoritmo de lazo cerrado.

Referencias

1. Microblaze Processor Reference Guide. (2008) http://www.xilinx.com, Xilinx Inc.
2. Nios II Processor Reference Handbook. (2008) http://www.altera.com. Altera Corp.
3. Quartus II Handbook, (2007) http://www.altera.com, Altera Corp.
4. DE2 Development and Education Board User Manual. (2008) http://www.altera.com, Altera Corp.
5. SOPC Builder Applications Altera Corp. (2006), http://www.altera.com/products/software/products/sopc/sopindex.html
6. Using Nios II Floating-Point Custom Instructions. (2008) http://www.altera.com/literature/tt/tt_floating_point_custom_instructions.pdf, Altera Corp.
7. Nios II Custom Instruction User Guide, (2008) http://www.altera.com/literature/ug/ug_nios2_custom_instruction.pdf, Altera Corp.
8. Borja Martínez Huerta, Márius Montón Macián, Jordi Carrabina Bordoll (2008) Síntesis de Unidades Funcionales para Soft-Cores desde un modelo C/C++. *FPGAs: Metodologías, herramientas y aplicaciones.* Juán A. Gómez, Juán M. Sánchez y Miguel A. Vega(Eds.), 112-117. Universidad de Extremadura.
9. David M. Cambre, Eduardo Boemo y Elías Todorovich. (2008). Energy evaluation in the Nios II Processor as a function of cache sizes. *IV Southern Conference on programmable logic.* San Carlos de Bariloche, Argentina.