Implementación del núcleo de un microcontrolador compatible con los PIC de la gama media en FPGA de Altera

D. Criado[1], P. Montejo[2], V. Escartín[3]

[1] *CIME,Facultad de Eléctrica, CUJAE dilaila.criado@electrica.cujae.edu.cu*

[2] *CIME,Facultad de Eléctrica, CUJAE pablo.montejo@electrica.cuaje.edu.cu*

[3] *CIME,Facultad de Eléctrica, CUJAE victor.escartin@electrica.cujae.edu.cu*

RESUMEN / ***ABSTRACT***

En este trabajo se describe, el diseño del núcleo de un procesador compatible con el juego de instrucciones de microcontroladores PIC de la gama media. La estructura utilizada posee un pipeline de dos etapas. Se utiliza el lenguaje de descripción de hardware VHDL con dispositivos programables de las familias Cyclone de Altera. La metodología de diseño empleada es un proceso recurrente que se inicia con la definición de la arquitectura a nivel de bloques funcionales del sistema a diseñar, se realiza la modelación estructural y la implementación de cada bloque. Finalizando con la simulación funcional y temporal. Todo este proceso tiene como objetivo lograr un dispositivo con la mayor velocidad posible en su funcionamiento y sintetizarlo con el menor número de celdas.

Palabras claves: FPGA, microcontrolador, Módulo IP, VHDL

Mid range Microcontroller PIC core implementation in Altera FPGA.

In this work it is described, the design of a microcontrolador half range PIC core. The language of hardware description is VHDL with programmable devices of Altera Cyclone familiess. The methodology of design employee is a recurrent process that begins with the definition of the architecture with a functional blocks level, then is carried out the structural modelation and the implementation of each block. Concluding with the functional and timming simulation. This whole process has as objective to achieve a device with the biggest speed in its operation and to synthesize it with the smallest number of cells.

Key words: FPGA , Microcontrollers, Module IP, VHDL

INTRODUCCIÓN

Los dispositivos lógicos programables en adelante FPL (Field Programmable Logic), desarrollados a finales de los 70 hoy son muy populares. En la actualidad representan a uno de los sectores que más rápido crecimiento experimenta en la industria de los semiconductores [1]. Estos dispositivos permiten a los diseñadores contar con un alto grado de flexibilidad, ventajas en la puesta hacia el mercado, integración del diseño, además de ser fáciles de utilizar y poder ser reprogramados varias veces.

En el área de los microcontroladores existen ofertas IP de varios dispositivos estándar que permiten aprovechar la gran cantidad de herramientas existentes (compiladores, linkers, simuladores), así como la existencia de bibliotecas de funciones de software probadas y eficientes, que pueden ser usadas para la generación de código

En este trabajo se selecciona la familia de microcontroladores PIC dado que de la misma se fabrica una gran variedad de dispositivos, lo que permite encontrar el más adecuado para una aplicación dada. Además de disponer de las herramientas de desarrollo para la realización de cualquier tipo de aplicación. Se seleccionó el núcleo de un microcontrolador compatible con el PIC de gama media de la Microchip, para sintetizarlo sobre un FPGA de Altera partiendo de la experiencia del diseño de un núcleo de microcontrolador PIC de la gama baja [2]. La herramienta de software utilizada en este trabajo es el programa Quartus II V5.0 de Altera.

Metodología de diseño.

El “soft core” diseñado desde el punto de vista de sus terminales posee hasta cinco puertos bidireccionales, además de la señal de reloj y de reset (MCLR).

Para contrarrestar la complejidad del sistema a diseñar, se utiliza una metodología de diseño basada en la descripción RTL (Register Transfer Logic), además el código VHDL se realiza pensando en un nivel de abstracción medio – bajo y con descripciones orientadas para síntesis en pro de optimizar la utilización de los recursos del circuito programable.

En este diseño se cumplieron las siguientes etapas:

1. Se establecieron todas las especificaciones del circuito a partir del manual del fabricante y el propósito de hacerlo lo más rápido posible con el menor consumo de recursos.
2. Proceso de refinamiento gradual hasta lograr una arquitectura a nivel de transferencia de registro que sea sintetizable eficientemente.
3. Síntesis lógica que garantice los requerimientos funcionales con el menor consumo de recursos posible y las restricciones de tiempo requeridas.

Para cada unidad diseñada se valoraron diversas descripciones y se selecciona la que presenta mejores desempeños en cuanto a área y velocidad.

Arquitectura del microcontrolador.

Del estudio del manual del fabricante de este microcontrolador [3], y después de analizar varias variantes se llegó a la arquitectura general que se muestra en la figura 1. La cual está compuesta por dos bloques funcionales, la unidad de procesamiento (datapath) y la unidad de control (controlpath).

La unidad de procesamiento está formada por los registros donde se almacenan datos, la unidad de cálculo (ALU) y los recursos de interconexión entre registros. La unidad de control genera las señales que controlan la transferencia de información entre los diferentes componentes de la unidad de procesamiento. Está integrada por el decodificador y una máquina de estados.

En la figura 1, el ALU (arithmetic and logic unit), es la unidad en donde se realizan todas las operaciones sobre los datos, según sea ordenado por la unidad de control. Port a, b y c, representan los posibles puertos bidireccionales, dependiendo del microcontrolador, en particular en esta familia pueden estar presentes hasta 5 puertos o solo alguno de ellos. La unidad de búsqueda de instrucciones contiene la memoria en donde se almacena el programa a ejecutar y se encarga de la búsqueda de cada instrucción.

Para poder intercambiar información entre los registros se requiere de buses que los interconecten, se utilizan multiplexores (MUX_A y MUX_B). A los multiplexores se conectan los registros de funciones especificas (W, OPTION, INTCON, Status, FSR etc), el FILE representa a los registros de propósitos generales realizados en RAM. Periféricos representa a los registros asociados a cada periférico según sea el miembro de esta familia utilizado.

Ciclos de las instrucciones y segmentación.

La ejecución de una instrucción básicamente consiste en el movimiento de los datos entre los registros. Algunos datos son transferidos sin modificarlos y otros son manipulados antes de almacenarlos. En la figura 2, se muestra la estructura seleccionada para este diseño del ciclo de ejecución de una instrucción. Este ciclo se divide en dos etapas. En la primera (Fetch) se busca el código de la instrucción y se decodifica. En la segunda etapa (Execute), se busca el operando, se realiza la ejecución de la instrucción y al final se guarda el resultado.

Cada una de las etapas utiliza circuitos distintos por lo que solo una pequeña parte de los circuitos del microcontrolador es utilizada en cada momento. Un modo de mejorar la utilización de los circuitos disponibles y también la velocidad en la ejecución de las instrucciones es comenzar una instrucción antes de que la anterior termine. Esta técnica se conoce como segmentación (pipeline), y es un modo efectivo de utilizar la concurrencia en un microcontrolador. En este diseño cada etapa del ciclo de instrucción demora un período de reloj. De esta forma la duración efectiva de la ejecución de cada instrucción solo demora un período del reloj. Lo que hace que este diseño solo por este concepto sea cuatro veces más rápido que los microcontroladores equivalentes comerciales.

En la etapa de ejecución de una instrucción N, el registro contador de programa tiene la dirección de la instrucción N + 1 a buscar. Las señales de control obtenidas de la decodificación de la instrucción N + 1, se almacenan en un registro para garantizar que las mismas sean estables durante el período de reloj siguiente, es decir para la etapa de ejecución.

Esta estructura presenta una dificultad al ejecutar las instrucciones de saltos que dependen de una condición. En este caso se emplea un ciclo de reloj adicional cuando se cumple la condición ya que necesita buscar la nueva instrucción dada por el salto, por lo que tiene que esperar que termine la ejecución de la instrucción actual para poder decidir la dirección de la siguiente instrucción.

Al estar la unidad de registro de propósitos generales implementada en una RAM sincrónica y para que esta pueda tener listo el dato en el ciclo de ejecución, la dirección del dato hay que suministrarla en el ciclo de búsqueda (adelantada).

En este diseño la decodificación y ejecución de las instrucciones de saltos incondicionales (GOTO, CALL, RETFIE y RETURN), se implementaron de modo tal que su ejecución se redujo a un solo período del reloj.

Diseño de los bloques funcionales.

Diseño del ALU.

Este bloque posee todos los operadores necesarios para realizar las funciones que requiere esta familia de microcontroladores, las cuales fueron obtenidas del análisis de su manual [3].

En la figura 3, se muestra la arquitectura desarrollada para el ALU, este bloque está formado por tres unidades, una para

realizar las funciones lógicas de los operandos (AND, OR, XOR), otra para las rotaciones y para dejar pasar hacia la salida a uno de los dos buses de entrada y la tercera para realizar la operación aritmética de la suma ya que la resta se realiza en complemento a dos. Para el sumador, la variante evaluada que ocupa menos celdas lógicas y con un tiempo de demora menor fue utilizando el sumador de la biblioteca de Altera lpm_add_sub [4]. También componen esta unidad dos multiplexores para seleccionar al bus de entrada o al Bit_Patern y sus complementos. La señal Bit_Patern se utiliza en las operaciones sobre bits y se obtiene de la decodificación del vector que permite seleccionar el bit con el cual se debe operar.

La síntesis resulto en 92 celdas, lo que da como promedio que en este diseño se consume 11.5 celdas por bit.

Diseño de la unidad de búsqueda de instrucciones.

En la figura 4, se muestra la arquitectura desarrollada para la unidad de búsqueda de instrucciones. La integra el registro contador de programa (PC) parametrizable, hasta 13 bits, que contiene la copia de la dirección de la instrucción que se está buscando en ese momento. Como la memoria es sincrónica posee internamente un registro en donde se almacena la dirección de la instrucción. También posee un Stack parametrizable para almacenar al contador de programa en el caso de instrucciones CALL o de interrupción. La unidad de generación de direcciones, en donde se determina la dirección de la próxima instrucción a ejecutar y la memoria ROM donde se almacena el programa.

El generador de direcciones está formado por el registro PCLATH, un circuito que incrementa, un multiplexor que permite seleccionar la información a cargar en el registro PC según sea la instrucción que se esté ejecutando. El resultado de la síntesis son 175 celdas

Diseño de los puertos de entradas y salidas.

Los puertos de propósito general no se realizaron según la estructura del diseño original de Microchip sino que cada puerto bidireccional se dividió en dos, uno de entrada y otro de salida. Esta familia de dispositivo puede tener hasta cinco tipos puertos bidireccionales, dos de 4 bits y los restantes de 8 bits cada uno. En el presente diseño también se implementaron los 5 tipos de puertos pero separando las entradas de las salidas.

Diseño del controlador de Interrupciones.

El circuito utilizado para atender las interrupciones es similar al utilizado por Microchip [3], el cual se utiliza para validar las potenciales líneas de solicitud de interrupción en función de la máscara establecida en el registro INTCON. Las simulaciones del diseño realizadas mostraron que la latencia máxima, medida desde que ocurrió el evento (interrupción) hasta que comienza la ejecución de la instrucción en la dirección 004H es de 2 periodos del reloj,

Diseño de la memoria de programa.

Se implementa una memoria ROM de 512 x 14 bits, que se utilizara como memoria de programa. La capacidad de esta memoria se selecciona según lo requiera la aplicación. El contador de programa es de 13 bits, por lo que memoria máxima con la que se puede trabajar es de 8K palabras de 14 bits cada una, aunque hay miembros de esta familia con 1K palabra. El contenido de la memoria se escribe en un fichero con extensión ".mif".

Diseño de la unidad de registros de propósito general.

Con el objetivo de disponer de hasta 512 registros de 8 bits se realizó este diseño utilizando una memoria RAM sincrónica que poseen estos dispositivos, pudiéndose seleccionar su capacidad según la aplicación. Para su implementación se decidió utilizar la *megafunction* de Altera: ALTSYNCRAM estructurándola como una RAM sincrónica de dos puertos, uno para permitir la escritura y el otro la lectura, esta estructura es para poder leer y escribir la memoria en un solo período de reloj.

Diseño del stack.

Se realizo el diseño de un stack parametrizable para la salva del contador de programas en la instrucción CALL y en la atención a las interrupciones.

Diseño del decodificador de instrucciones.

Esta unidad es combinacional formada por el decodificador de las instrucciones y un registro que almacena todas las señales generadas por esta unidad.

Este microcontrolador tiene una codificación en sus instrucciones relativamente compacta, con una longitud fija. Del análisis de las instrucciones, se obtiene que hay cuatro formatos de instrucciones, que están organizadas por campos en donde los dos bits más significativos identifican cada formato.

Diseño de la Unidad de control.

Esta unidad junto con el decodificador de instrucciones generan todas las señales necesarias para que el resto del sistema funcione. Está integrada por una máquina de estados del tipo Mealy con salidas con registros, para lograr mayor velocidad de operación. Posee 4 estados, en el estado Fetch_EXE, se busca la instrucción y se decodifica, si la instrucción demora un solo ciclo de reloj se permanece en este mismo estado debido al uso de una estructura segmentada, mientras se busca una instrucción se ejecuta la anterior. Los estados Skip_Z (Salto si es cero) y Skip_NZ (salto si no es cero) son para las instrucciones de saltos condicionados si se cumple la condición del salto. Y el estado Sleep relacionado con la ejecución de la instrucción con este mismo nombre que a diferencia de los otros estados solo se sale por la señal de Reset o por interrupción

Resultados.

El resultado de la síntesis del núcleo del microcontrolador conformado por las unidades anteriormente descritas con una memoria de programa de 512 palabras, 512 bytes de RAM, El stack con 16 palabras, tres puertos de 8 bits cada uno, el timer0, el watchdog y se resume en la tabla 1.

La determinación de la frecuencia máxima es empleando el modo de estimado pesimista. Es importante observar en la tabla anterior que el núcleo diseñado consume 509 celdas y puede trabajar hasta una frecuencia de 107.90 MHz (Cyclone

III), resultado que convierte esta variante de núcleo en una propuesta muy económica desde el punto de vista de consumo de recurso y además muy rápida, desarrolla una velocidad de hasta 107.90 MIPS utilizando la familia Cyclone III. Cada instrucción demora un periodo de la señal del reloj, a no ser las de saltos condicionados que demoran un periodo del reloj adicional. Estas velocidades se obtienen considerando que no hay instrucciones de salto condicionados, las cuales disminuyen la velocidad de operación.

En la tabla 2, se muestran diferencias en cuanto a la duración de la ejecución de algunas instrucciones en la implementación presentada en este trabajo y el dispositivo comercial. Los resultados de la implementación en VHDL son mejores debido a la arquitectura adoptada.

En la tabla 3 se muestra algunas características del diseño realizado en el presente trabajo y un dispositivo comercial

La tabla 4 refleja el consumo de módulos de memoria M4K [5, 6] que ocupa el presente diseño en el FPGA Cyclone. Cada módulo posee 4096 bits y pueden ser organizados en varios formatos (4096 x 1, 512 x 8, 256 x 16).

La validación del diseño se realizó, utilizando el programa MPLAB para generar el fichero en lenguaje ensamblador del microcontrolador. El fichero resultante de este programa se convirtió a formato .MIF, el cual es el que lee el programa en lenguaje VHDL y con el simulador del programa Quartus II versión 5.0 se verificó el correcto funcionamiento de todas y cada una de las instrucciones en el microcontrolador diseñado.

Si se comparan los resultados obtenidos en este trabajo con el PIC de la familia 16C6X para dispositivos programables de Altera ofertado comercialmente por Digital Core Design [7] los resultados son similares en cuanto a la velocidad de operación.

CONCLUSIONES

El núcleo desarrollado trabaja a una velocidad que al menos es 4 veces superior a la del dispositivo comercial ya que la ejecución de una instrucción que no sea de salto consume un solo ciclo del reloj en lugar de 4 como en el circuito comercial. Por otro lado la frecuencia de operación dada la familia Cyclone empleada es al menos 15 veces superior a la de los dispositivos comerciales.

El procesador desarrollado a diferencia del original realiza las operaciones de saltos incondicionales en un solo periodo del reloj y la latencia en las interrupciones es de solo de 2 ciclos de reloj. Además permite la salva automática en un solo período de reloj de 4 registros para aumentar la velocidad en el cambio de contexto entre tareas.

Otro resultado de este trabajo es resaltar la viabilidad de diseñar sistemas complejos utilizando una metodología de diseño RTL para la especificación de la arquitectura, VHDL para la descripción y los dispositivos programables para la realización de sistemas complejos

REFERENCIAS

1. Jason G. Tong and mohammed A. S. Kalid, “Profiling tools for FPGA based embedded sustems”, Journal of Computer, Vol 3, No 6, June 2008.
2. **LESLIE GONZÁLEZ GONZÁLEZ–ALFONSO, DILAILA CRIADO CRUZ, VÍCTOR ESCARTÍN FERNANDEZ**, “Diseño del núcleo de un microprocesador tipo PIC utilizando lógica programable Flex10k de Altera”, presentado al *Taller Nacional “Técnicas Avanzadas de Diseño de Hardware e Instrumentación Virtual”, TADHIV 2006* La Habana, Cuba, 2006
3. **DS33023A**, “PICmicro Mid-range MCU family, Reference manual”, Microchip, 1997
4. **MARCELO S. PORTOS, ANDRÉ M. SILVA, ROGER E. C. PORTO, JOSÉ LUÍS A. GÜNTZEL, LUCIANO V. AGOSTINI**, “Impactos do uso de diferentes arquitecturas de somadores em FPGAS Altera”,presentado al Iberchip, 2005.
5. **QUARTUS II**, Handbook, volume 1, Design and Synthesis, Altera, 2004
6. Quartus II, V5.0 Introduction using VHDL design, Altera, 2005
7. DRPIC166X – Digital Core Design, December 2004.

AUTORES

Pablo Montejo Valdés. Profesor asistente y Master en electrónica, Graduado en XXX de Ingeniería en Telecomunicación y Electrónica

Dilaila Criado Cruz. Investigador Agregado y Master en Electrónica, Graduado en 1992 de Ingeniería Fisica Nuclear.

Víctor Escartin Fernández. Profesor Consultante y Doctor en Ciencias Técnicas, Graduado en 1970 de Ingeniería en Telecomunicación y Electrónica

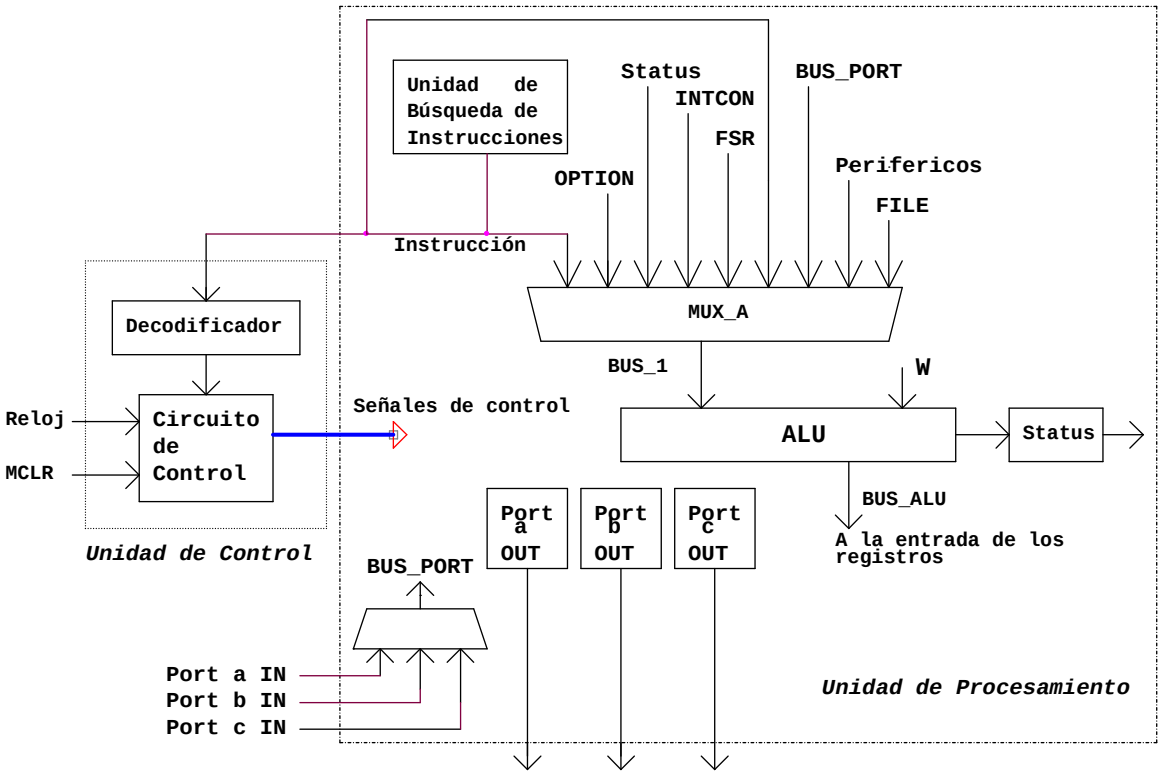

Figura 1. Arquitectura general del sistema.

FETCH		EXECUTE		
Búsqueda de la intrucción	Decodificación	Búsqueda del operando	Ejecución de la instrucción	Escritura del resultado

Figura 2, Ciclo de instrucción.

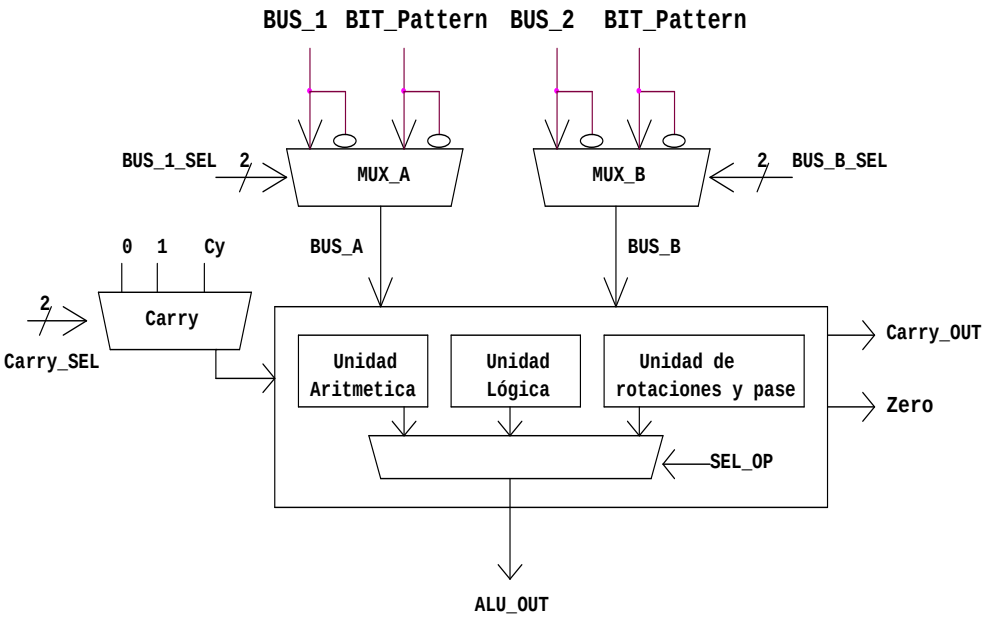

Figura 3, Arquitectura del ALU

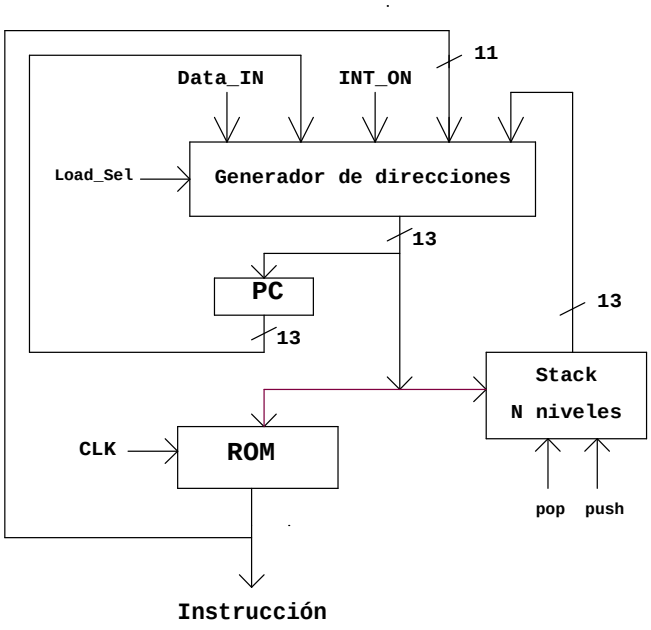

Figura 4, Arquitectura de la unidad de búsqueda de instrucciones

Tabla 1. Resultados de la síntesis

Familia	Celdas	Bits de Memoria	Módulos de memoria	Frecuencia Máxima del reloj (MHz)
Cyclone II (-6)	538	11336	4 x M4K	90.55
Cyclone III (-6)	515	11336	3 x M9K	102.73
Cyclone IV (-6)		11336		
Stratix II (-3)		11336	4 x M4K	119.96
Stratix II Gx (-3)		11336	4 x M4K	123.73
Stratix III (-2)		11336	3 x M9K	131.32

Tabla 2, Ciclos de reloj en la ejecución de las instrucciones de salto.

	PIC (comercial)	CIME
GO TO	8	2
CALL	8	2
RETURN	8	2
RETFIEe	8	2
RETLW	8	2
BTFSC	4 / 8	2
BTFSS	4 / 8	2
DECFSZ	4 / 8	2
INCFSZ	4 / 8	2

Tabla 3, Comparación con el dispositivo comercial.

	PIC (comercial)	CIME
Instrucciones	35	35
ROM	Hasta 8 K	parametrizable
RAM	Hasta 368	parametrizable
Stack	8	parametrizable
Puertos (Bits)	33	parametrizable
Reloj (MHz)	20	78
Ciclo de Instr / Ciclo de Reloj	4	1
Timers	Hasta 3	parametrizable
Watch dog	1	1
Modo sleep	1	1
Implementación	Full custom	FPGA

Tabla 4, Consumo de módulos de memoria M4K.

	Palabras	Bits	M4K
ROM	512	16	2
RAM	512	8	1
LIFO	16	13	1
Total			4