Diseño del núcleo de un procesador AVR de 8 bits utilizando lógica programable de Altera

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

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

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

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

RESUMEN / *ABSTRACT*

En este trabajo se describe, el diseño del núcleo de un microcontrolador AVR, en específico uno similar al AT90S8515. Se utiliza el lenguaje de descripción de hardware VHDL con dispositivos programables de las familias Cyclone y Cyclone II de Altera. Los objetivos que se persiguen con este trabajo son; valorar alternativas de diseño de sistemas digitales en circuitos programables, contar con el diseño de un núcleo AVR para futuras aplicaciones como modulo IP y realizar diseños de sistemas con varios procesadores (SoPC).

Palabras claves: FPGA, Microcontrolador, Módulo IP, núcleo AVR, VHDL,

Design of an AVR microcontroller core using programmable logic from Altera.

This paper describes the design of the nucleus of an AVR microcontroller, in particular a similar AT90S8515. It uses the hardware description language VHDL programmable devices with Cyclone and Cyclone II from Altera. The goals of this work are assessing alternative design of digital systems in programmable circuits, with the design of an AVR core module for future applications such as IP and to perform system designs with multiple processors (SoPC).

Key words: AVR core, FPGA, IP Module, Microcontroller,VHDL

INTRODUCCIÓN

La disponibilidad actual de dispositivos lógicos programables [3] en adelante FPL (Field Programmable Logic), de alta escala de integración y elevadas frecuencias de trabajo permiten abordar diseños de sistemas complejos constituidos por procesadores embebidos de propósito específico o no y otros subsistemas dedicados en un único circuito integrado. 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

MATERIALES Y MÉTODOS

***Selección del microcontrolador*.**

Se seleccionó el núcleo de un microcontrolador compatible con el AT90S8515 de Atmel [4], para sintetizarlo sobre un FPGA de Altera partiendo de la experiencia del diseño de un núcleo de la Familia PIC16C5X [5]. Para la selección del microcontrolador [6] a sintetizar hay que tomar en cuenta aspectos tales como, que no sea muy complejo y que exista comercialmente para poder utilizar todos los recursos de software desarrollados para él. El AVR es un microcontrolador con un conjunto reducido de instrucciones (RISC) pero lo suficientemente amplio que permite reducir el tamaño del código para una aplicación dada y aumentar su velocidad. Posee 32 registros de trabajo que permiten lograr códigos muy eficientes sobre todo si se programa utilizando lenguaje C. También posee capacidad para manejar zonas de memoria de datos relativamente grandes.

Diseño del núcleo de un procesador AVR

La herramienta de software utilizada en este trabajo es el programa Quartus II V5.0 de Altera [7]. El microcontrolador comercial desde el punto de vista de sus terminales posee hasta cuatro puertos bidireccionales, además de la señal de reloj y de reset (MCLR). El “soft core” diseñado utiliza un terminal para el reloj del sistema otro para el terminal del reset, otro para el reloj del watch dog y terminales hasta para 4 puertos de 8 bits cada uno en dependencia de la aplicación.

Para la implementación de este diseño se comenzó estableciendo todas las especificaciones del circuito y la arquitectura general del sistema a nivel RTL. Un proceso de refinamiento gradual se llevó a cabo hasta lograr una

arquitectura a nivel de transferencia de registro que fuera eficientemente sintetizable. La síntesis lógica garantiza los requerimientos funcionales y las restricciones de tiempo requeridas, seleccionando la variante que presenta los mejores desempeños en cuanto a área y velocidad.

En la figura 1 se muestra la arquitectura general obtenida. Esta se puede dividir en dos bloques funcionales: la unidad de procesamiento (datapath) y la unidad de control (controlpath).

La unidad de control que genera las señales para governar la transferencia de información entre los diferentes componentes de la unidad de procesamiento, está integrada por el decodificador, una máquina de estados y el registro de segmentación.

La unidad de cálculo (ALU), los Port a, b, c y d, que representan los posibles puertos bidireccionales y la unidad de búsqueda de instrucciones que contiene la memoria donde se almacena el programa a ejecutar y se encarga de la búsqueda de cada instrucción, forman parte de la unidad de procesamiento.

Para el intercambio de información entre los registros se requiere de buses que los interconecten, estos se implementaron mediante multiplexores (MUX_A y MUX_B). A estos multiplexores se conectan los registros de funciones específicas, el bloque identificado como RAM que representa a los 32 registros de propósito general y la zona de RAM interna. El bloque de constantes es el encargado de generar valores fijos como 0, 1 o detectar el valor de un bit en especifico

Ciclos de las instrucciones y segmentación

La mayoría de las instrucciones se realizan en un solo ciclo de instrucción el cual es idéntico al ciclo del reloj

El ciclo de ejecución de una instrucción fue dividido en dos etapas. En la primera etapa (Fetch) se busca el código de la instrucción y se decodifica y en la segunda etapa (Execute), se busca el operando, se realiza la ejecución de la instrucción y al final se guarda el resultado.

Con el objetivo de mejorar la velocidad en la ejecución de las instrucciones se implementó un pipeline de dos etapas. La estructura presenta una dificultad al ejecutar las instrucciones de saltos. En este caso se emplea uno o más ciclos de reloj adicionales ya que se necesita buscar la nueva instrucción dada por el salto.

Por estar realizada en una RAM sincrónica la unidad de registro de propósitos generales tiene que tener listo el dato en el ciclo de ejecución, por lo que la dirección del dato hay que suministrarla en el ciclo de búsqueda (adelantada).

Diseño del ALU.

En la figura 2, se muestra la arquitectura desarrollada para el ALU, este bloque está formado por la unidad aritmética, la unidad lógica y la unidad de rotación. Para el sumador de la unidad aritmética, la variante evaluada que ocupa menos celdas lógicas con un tiempo de demora menor, fue utilizando el sumador de la biblioteca de Altera lpm_add_sub [7].

También componen esta unidad dos multiplexores para seleccionar al bus de entrada o su complemento. Además de una sección que se encarga de evaluar el cumplimiento de la condición en las instrucciones de saltos condicionados.

La síntesis resulto en 87 celdas, lo que da como promedio considerando que es de 8 bits, un consumo de 10.88 celdas por bit.

Diseño de la unidad de búsqueda de instrucciones.

En la figura 3, se muestra la arquitectura desarrollada para la unidad de búsqueda de instrucciones. La integra el registro contador de programas (PC) de N bits (parametrizable según la aplicación), el cual contiene la copia de la dirección de la próxima instrucción ya que realmente como la memoria es sincrónica el verdadero contador de programa es interno en la memoria. También posee tantos niveles de Stack como se necesiten (parametrizable). La unidad de generación de direcciones, donde se determina la dirección de la próxima instrucción a ejecutar y la memoria ROM donde se almacena el programa. El generador de direcciones esta formado por un multiplexor que permite seleccionar la información a cargar en el registro PC según sea la instrucción que se este ejecutando.

La memoria de programa es sincrónica (es la única variante que soporta la familia Cyclone). Con esta arquitectura se logra que cada instrucción demore un periodo de la señal del reloj, a no ser las de saltos que demoran más. El resultado de la síntesis de esta unidad son 56 celdas

Diseño de los Puertos bidireccionales.

Los puertos bidireccionales de propósito general se realizaron según la estructura del diseño original de Atmel. Este dispositivo puede tener hasta cuatro puertos de 8 bits cada uno.

Diseño del controlador de interrupciones.

El circuito para atender las interrupciones es similar al utilizado por Atmel [2], el cual valida las potenciales líneas de solicitud de interrupción en función de la máscara establecida en los registros que controlan la interrupción. La latencia mínima, medida desde que ocurrió el evento (interrupción) hasta que comienza la ejecución de la instrucción es de 4 periodos del reloj,

Diseño de la memoria de programa

Se implementa una memoria ROM sincrónica con palabras de 16 bits, cuya capacidad se selecciona según lo requiera la aplicación. El contenido de la memoria se escribe en un fichero con extensión ".mif".

Diseño de la unidad de RAM.

Con el objetivo de disponer de los 32 registros de trabajo y de localizaciones para guardar información se utilizó una memoria RAM sincrónica de 8 bits, pudiéndose seleccionar su capacidad según la aplicación. Para su implementación se decidió utilizar la *megafunction* de Altera: ALTSYNCRAM [7, 8] estructurándola como una RAM de tres puertos, uno

para permitir la escritura y dos para la lectura, esta estructura permite leer y escribir la memoria en un solo periodo de reloj.

Diseño del Stack.

Esta unidad fue diseña utilizando una RAM organizada como LIFO. La estructura seleccionada permite almacenar tanto al contador de programa como cualquier registro.

Diseño del decodificador de instrucciones

Esta unidad la forman el decodificador de las instrucciones y un registro que almacena algunas de las señales generadas.

El microcontrolador tiene una longitud fija para la codificación de sus instrucciones, encontrándose diferentes formatos identificados por los cuatros bits más significativos.

Las señales de control obtenidas en la decodificación de la instrucción se almacenan en un registro para garantizar que las mismas sean estables durante el ciclo de reloj siguiente, es decir el ciclo de ejecución.

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 mostrada en la figura 4. esta máquina es tipo Mearly, con salidas registradas para lograr mayor velocidad de operación. Posee 7 estados, en el estado Fetch_EXE, se busca la instrucción y se decodifica, si la instrucción demora un solo ciclo de reloj se vuelve a este mismo estado.

Posee un estado asociado a las interrupciones, dos estados mas para las instrucciones de 2 o 3 estados. El estado Sleep relacionado con la ejecución de la instrucción con este mismo nombre, a diferencia de los otros estados de aquí solo se sale: por la señal de Reset o por interrupción. Además posee otros dos estados para las instrucciones que trabajan con words en lugar de bytes

Diseño de la unidad de 16 bits.

Esta unidad es para atender a los registros índices de 16 bits y la posibilidad de post incremento o pre decremento. Esta formada por seis registros de 8 bits que pueden trabajar como registros de propósito general o como tres registros de 16 bits que utilizan varias instrucciones.

Resultados y discusión

En la tabla I, se resumen los resultados de la síntesis del núcleo del microcontrolador conformado por las unidades anteriormente descritas con una memoria de programa de 1024 palabras, los registros básicos y los puertos A, B, C y D de 8 bits cada uno. Además del tratamiento a la interrupciones y al estado de *sleep*, 1 kbytes de memoria RAM, un Stack de 256 palabras y un watchdog.

Cada instrucción demora un período de la señal del reloj, a no ser las de saltos que requieren más ciclos de la señal del reloj. Esto permite alcanzar velocidades de hasta 50 MIPS según la variante que se utilice como se muestra en la tabla anterior. Estas velocidades se obtienen considerando que no hay instrucciones de salto, las cuales disminuyen la velocidad de operación.

En la tabla II, se muestran diferencias en cuanto a la duración de la ejecución de algunas instrucciones entre la implementación presentada en este trabajo (CIME) y el dispositivo comercial. Los resultados de la implementación en VHDL son iguales o mejores esto es debido a la arquitectura adoptada.

En la tabla III se muestra algunas características del diseño realizado en el presente trabajo (Cyclone), el dispositivo comercial y otro diseño sobre FPGA.

La tabla 4 refleja el consumo de módulos de memoria M4K [7, 8] 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). Al estar organizada la RAM con tres puertos: dos de lectura y uno de escritura, ocupa 4 módulos.

La validación del diseño se realizó, utilizando el programa Wavrasm V1.11, 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.

El núcleo de procesador aquí presentado (AT90S8515) ocupa menos celdas y trabaja a una frecuencia superior que el del trabajo reportado en [9] en el cual diseñan un núcleo de AT90S1200 que ocupó 1060 celdas con una frecuencia de trabajo máxima de 12 MHz

CONCLUSIONES

Con la metodología utilizada para el diseño del microcontrolador, aunque no se trabaja directamente sobre los elementos lógicos, se le suministra al compilador un código VHDL en un nivel de abstracción orientado para síntesis, que permite alcanzar mejores resultados en cuanto a área y velocidad.

El núcleo desarrollado trabaja a una velocidad que al menos es 2.5 veces superior y ejecuta las operaciones de saltos en igual o menor tiempo debido a la arquitectura adoptada en comparación con el dispositivo comercial.

El *softcore* obtenido permite resolver diversidad de problemas de forma secuencial suministrando versatilidad en la integración de sistemas digitales complejos, economizando recursos de hardware, esfuerzo de ingeniería y tiempo de desarrollo.

REFERENCIAS

1. Altera Corp, Cyclone Device Handbook. Volume 1, 2003.
2. Altera Corp, CycloneII Device Handbook. Volume 1, 2007.
3. Jason G. Tong and mohammed A. S. Kalid, "Profiling tools for FPGA based embedded sustems", Journal of Computer, Vol 3, No 6, June 2008.
4. Atmel Corp., AT90S8515, 8 bit AVR Microcontroller with 8 KBytes downloadable Flash.,2001, ATMEL Data sheet
5. **CRIADO D**., "*Diseño del núcleo de un microcontrolador tipo PIC16C5X con arquitectura segmentada utilizando lógica programable"*, presentado al *FIE2008*, Stgo de Cuba, Cuba, ISBN978-84-00-08680-0.
6. Altera Corp, Design and Synthesis Quartus II Handbook, volume 1, 2004
7. Altera Corp, Introduction using VHDL design Quartus II V5.0, 2005

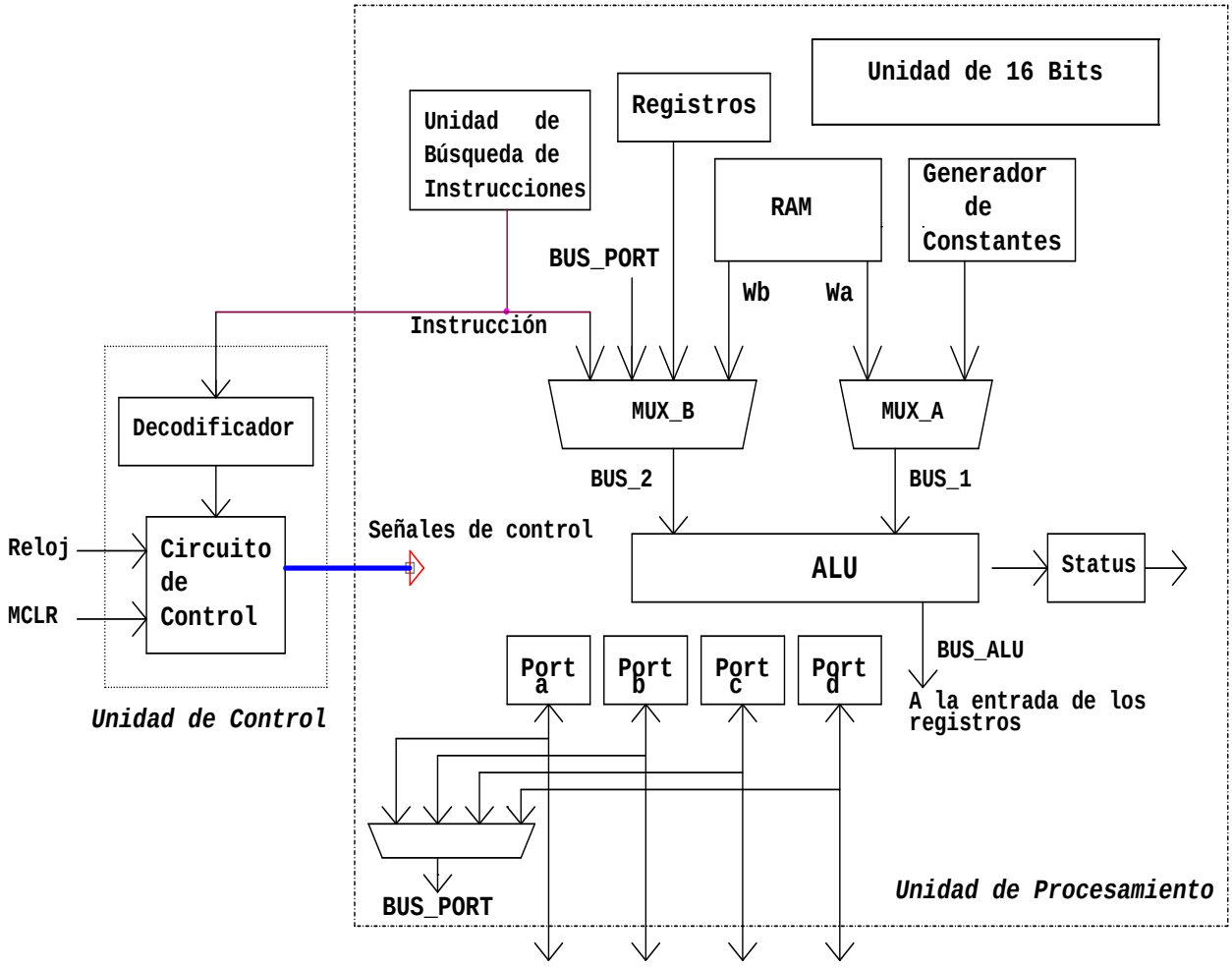

Figura 1. Arquitectura general obtenida.

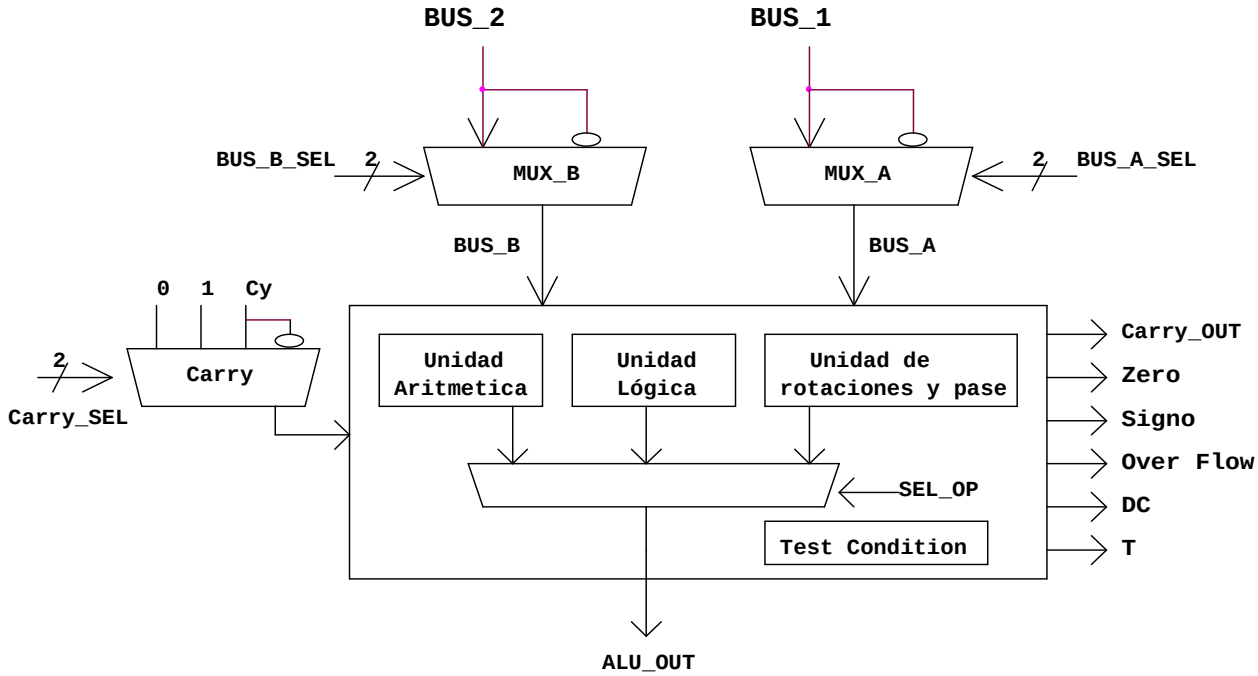

Figura 2. Arquitectura de al ALU.

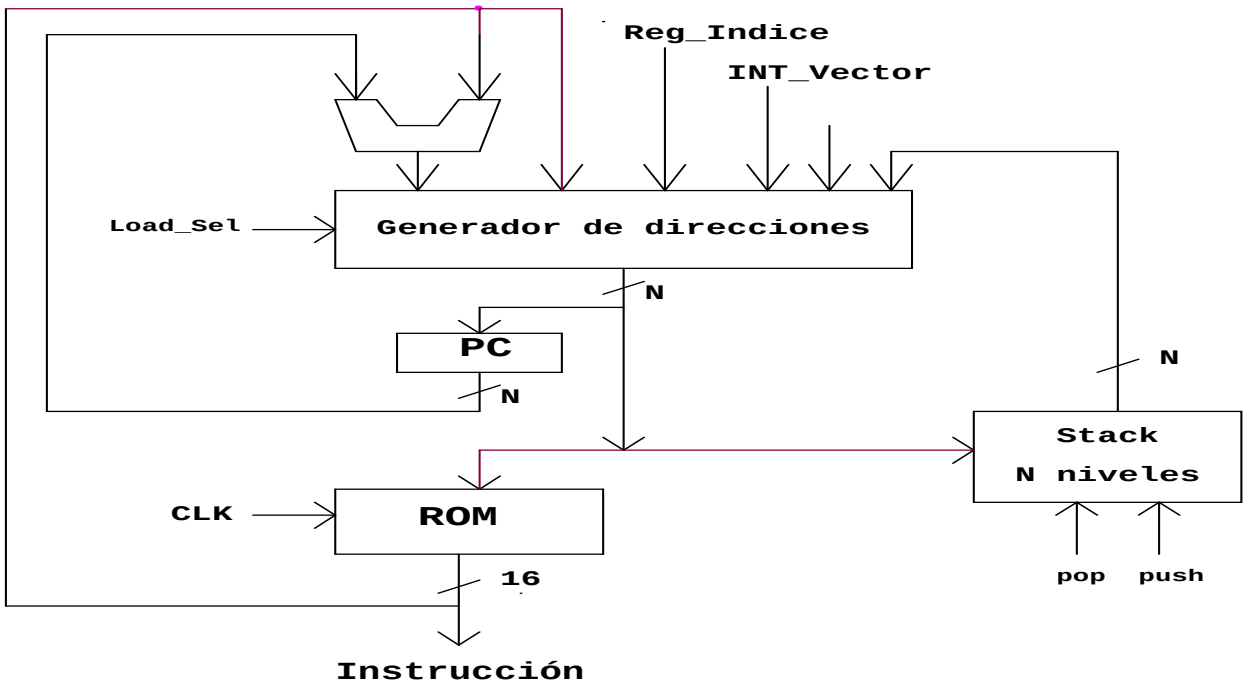

Figura 3. Arquitectura de la unidad de búsqueda de las instrucciones

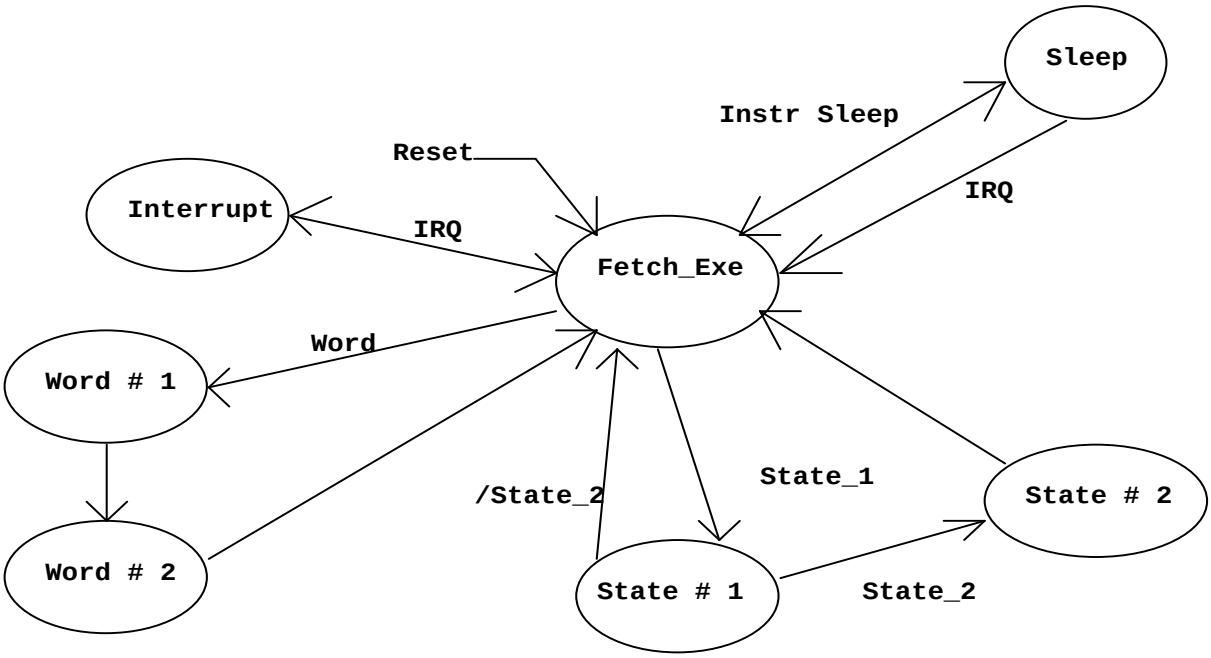

Figura 4. Diagrama de estados de la unidad de control.

Tabla I Resultados de la síntesis

Familia	Velocidad	Celdas	Bits de Memoria	Frecuencia Máxima del reloj (MHz)
Cyclone	-6	1085	36 864	45.85
Cyclone II	-6	1113	36 864	56.15
Cyclone III	-6	1090	36 864	57.15

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

	90S8515	CIME
Jmp	3	3
IJmp	2	2
RJmp	2	1
Branch	2 / 1	2
Call	4	3
ICall	3	2
RCall	3	3
Ret	4	2
Skip	2 / 1	2
Push	2	1
Pop	2	2
Store	2	2
Load	2	2

Tabla III Características de diferentes diseños.

	90S8515	90S1200	CIME
Instrucciones	120	92	120
Registros	32	16	32
ROM	8 K	0.5 K	Parametrizable
RAM	0.5 K	-	Parametrizable
Stack	usuario	4	Parametrizable
Puertos	4	3	4
Reloj (MHz)	20	12	50
Timers	2	-	-
Implementación	CMOS	FPGA	FPGA
Watch dog	1	-	1
Modo sleep	1	-	1
Otras			