Implementación en VHDL del Primer Microprocesador de INTEL en una FPGA

Alfredo Granados Ly, Michael Alejandro Díaz Illa

Facultad de Ingeniería Electrónica y Eléctrica, Universidad Nacional Mayor de San Marcos, Lima, Perú

RESUMEN: Hoy en día muchos fabricantes de circuitos integrados utilizan la tecnología de lógica programable para implementar y validar sus nuevos diseños sin recurrir a la fabricación del chip. Con este trabajo demostramos las ventajas que tienen las herramientas de diseño automático con las que cuentan actualmente los ingenieros de diseño y que hace muchos años atrás no se contaba, aumentando la productividad, reduciendo los costos y los tiempos de desarrollo son más cortos. Se ha implementado el primer microprocesador de INTEL 4004 sin tener que desarrollar el esfuerzo realizado por sus creadores.

I. INTRODUCCIÓN

La historia del 4004 comienza en el año 1968 cuando Robert Noyce y Gordon Moore fundaron Intel Co[1]. Un año más tarde recibió el encargo de la empresa japonesa Busicom para diseñar los elementos de una calculadora. El diseño original utilizaba 12 chips distintos, por lo que al ingeniero de Intel Ted Hoff se le ocurrió diseñar un procesador genérico, de tal manera de que se pueda cambiar el comportamiento del chip sin tener que rediseñar un nuevo circuito integrado.

El proyecto comenzó en 1969 y tuvo un costo de $60,000. Lo inicio Ted Hoff y Stanley Mazor definiendo un CPU de 4 bits, una memoria para las instrucciones y otra para los datos con puertos de entrada/salida para manejar el teclado, indicadores e impresora. En 1970 se unió al proyecto Federico Faginn (quien realmente fue el que terminó el diseño) al estar atrasados en los plazos de entrega de los chips.

Así finalmente, en Noviembre de 1971 (casi dos años después) salía al mercado el primer microprocesador denominado 4004. Se trataba de un dispositivo de silicio, compuesto por unos 2.300 transistores y que operaba con una frecuencia de 108 KHz. El costo en esa época de cada chip 4004 era de unos 200 dólares.

Faggin convenció a Noyce que el diseño del 4004 podía ser utilizado en aplicaciones de uso general. Por lo que Intel llegó a un acuerdo con Busicom, devolviendo el dinero del costo del proyecto, a cambio Intel podía utilizar el 4004 en otras aplicaciones que no fueran calculadoras. Desde entonces, Intel se ha convertido en el mayor fabricante de microprocesadores de la historia y en la empresa líder que hoy todos conocemos.

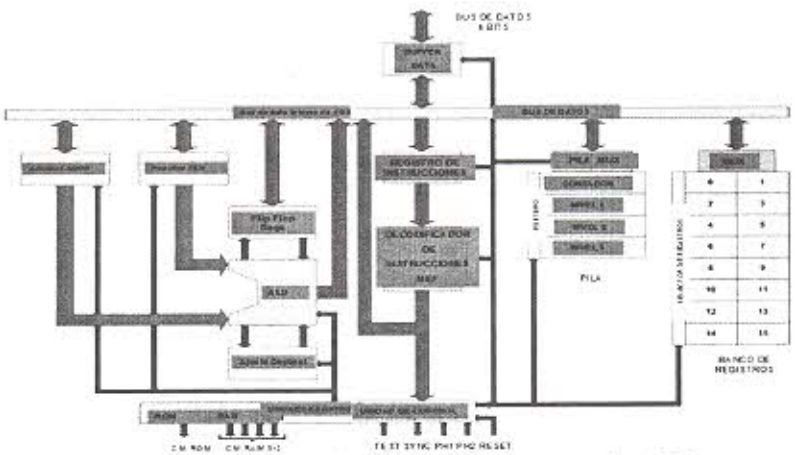

Fig.1. Diagrama de Bloques del microprocesador 4004

A. *DESCRIPCIÓN DEL MICROPROCESADOR 4004*

La figura 1 muestra el diagrama de bloques del microprocesador 4004. Diseñado completamente en base a una ruta de datos de 4 bits. Posee una arquitectura Harvard[2], con una memoria de instrucciones separada de la memoria de datos pero compartiendo el mismo bus. Es capaz de direccionar 32768 bits a una memoria ROM y 5120 bits de RAM. Además se pueden direccionar 16 puertos de entrada y 16 puertos de salida cada uno de 4 bits. Se diseñó con alrededor de 2,300 transistores PMOS utilizando tecnología de 10 micrones y teniendo un ciclo de instrucción de 10,8 microsegundos. Tiene un conjunto de instrucciones que está formado por 46 instrucciones, de las cuales 41 son de 8 bits de ancho y 5 instrucciones de 16 bits de ancho. Su nivel de Pila posee 3 registros para llamadas a subrutinas.

II. IMPLEMENTACIÓN DEL MICROPROCESADOR 4004 EN VHDL.

Para implementar el microprocesador 4004 se utilizó el lenguaje de programación VHDL, empleando el estilo estructural (Diseño BOTTOM - UP) en las últimas etapas [3], para ello cada bloque del procesador se diseñó en base al estilo flujo de datos y algorítmico, llegando a implementar los siguientes bloques:

- Banco de Registros.
- Acumulador.
- Contador de Programa.
- Unidad Aritmética Lógica.
- Registro de Instrucción.
- Unidad de Control.

A. *DISEÑO DEL BANCO DE REGISTROS*

El microprocesador 4004 cuenta con un banco de 16 registros cada uno de 4 bits (R0 a R15). El banco de registros tiene la capacidad de recibir y enviar un dato al bus de datos interno conectando con el ALU o con la memoria externa (ver figura 1). El banco tiene una entrada de datos (DATO) donde el dato de entrada es depositado en uno de los dieciséis registros dependiendo del valor del selector (SELD) y la habilitación del banco (W). Además para sacar el valor de un registro también se utilizo otro selector (SELA). La operación de escritura está sincronizada con la señal de reloj (CLK). El diagrama RTL del diseño del banco de registros se muestra en la figura-2, en el se pueden apreciar los registros con sus correspondientes circuitos de decodificación para la habilitación de escritura. Este diseño se realizó con los estilos flujo de datos y algorítmico.

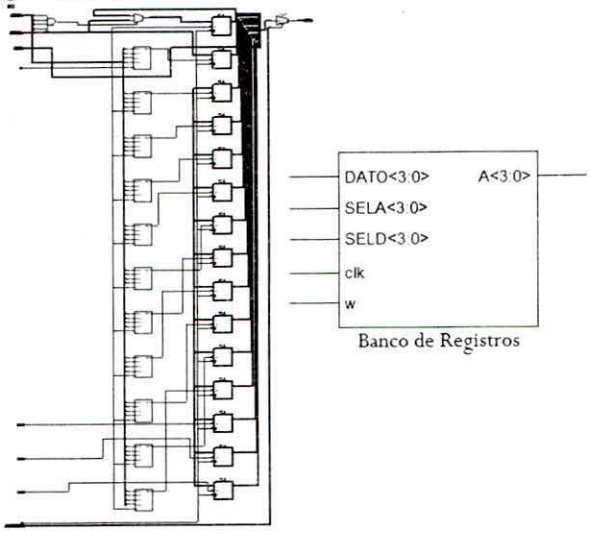

Fig. 2. Banco de Registros.

B. *DISEÑO DEL ACUMULADOR*

El 4004 posee un registro Acumulador el cual concentra la mayor cantidad de operaciones aritméticas, lógicas así como los accesos a la memoria de datos. En la Tabla-1 se muestra el conjunto de instrucciones que se aplican sobre el Acumulador con su correspondiente código de máquina. Cada una de estas instrucciones ocupa dos posiciones en la memoria de programa.

Mnemónico	Descripción	OPR	OPA
CLB	Limpiar el acumulador y el acarreo	1111	0000
IAC	Incrementar el acumulador	1111	0010
CLC	Complementar el acarreo	1111	0011
CMA	Complementar el acumulador	1111	0100
RAL	Rotar acumulador y acarreo hacia la izquierda	1111	0101
RAR	Rotar acumulador y acarreo hacia la derecha	1111	0110
TCC	Sumar acarreo al acumulador y limpiar el acarreo	1111	0111
DAC	Decrementar el acumulador	1111	1000
TCS	Restar acarreo del acumulador y limpiar el acarreo	1111	1001
DAA	Ajuste decimal del acumulador	1111	1011
KBP	Convierte un código 1 de 4 a binario en el acumulador	1111	1100

Tabla-1: Instrucciones que se aplican sobre el Acumulador.

El código de máquina está formado por los campos OPR y OPA siendo importantes porque el diseño de la unidad de control se realizó teniendo en cuenta estos valores.

En la figura-3 se muestra el diagrama a nivel de RTL de Acumulador, es un registro que almacena un dato de entrada y deja pasar sólo cuando su habilitador esta activo.

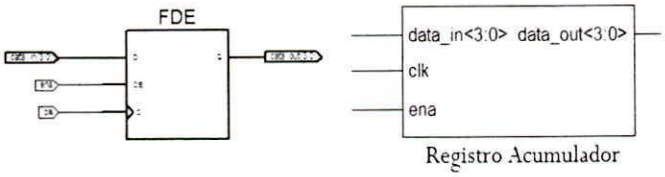

Fig. 3. Acumulador.

C. CONTADOR DE PROGRAMA

El contador de programa es el registro que se encarga de generar la dirección de la siguiente instrucción a ejecutar. Internamente se diseño como un registro de 12 bits para que sea capaz de mantener una dirección de 4192 posiciones de memoria posibles. El contador de programa se incrementa de acuerdo al tamaño de la instrucción y tiene la capacidad de cargar inmediatos para los saltos absolutos. Se diseñó utilizando el estilo algorítmico y el diagrama RTL se muestra en la figura 4.

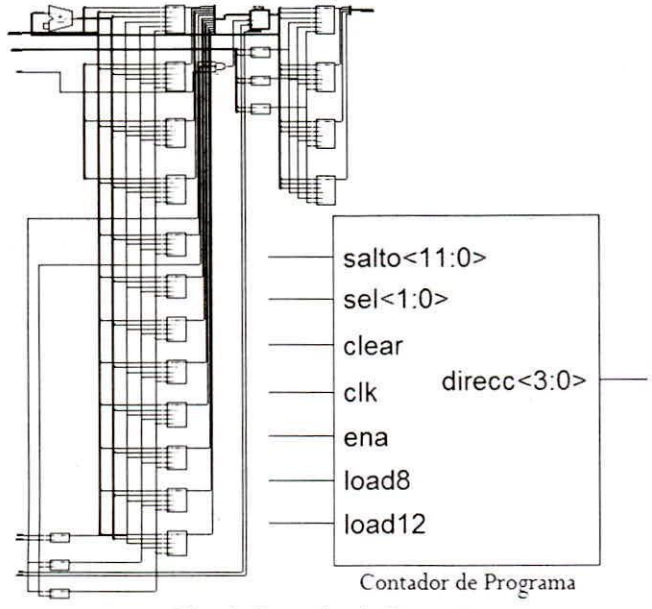

Fig. 4. Contador de Programa.

El contador de programa cuenta con una entrada CLEAR que borra el contenido del acumulador cuando se activa, si LOAD8 se activa toma los 8 bits de la dirección de salto que contiene el registro de instrucción y lo concatena con los bits de mayor peso del valor actual del PC (saltos paginados de 256 posiciones). Si LOAD12 se activa, el PC tomará los 12 bits de la dirección de salto que contiene el registro de instrucción (saltos absolutos de 4192 posiciones). Las direcciones de salto ingresan por el puerto de entrada SALTO.

En Tabla-2 se muestran las instrucciones relacionadas a los saltos. Se puede notar claramente como los inmediatos a cargar al PC pueden ser de 12 bits (instrucciones JUN, JMS) y 8 bits (instrucción JCN).

Mnemónico	Descripción	OPR				OPA			
JUN	Salto incondicional a la dirección de ROM A3, A2, A1	0	1	0	0	A3	A3	A3	A3
		A2	A2	A2	A2	A1	A1	A1	A1
JMS	Salvar el viejo valor del contador de programa y saltar a la dirección de ROM A3, A2, A1	0	1	0	1	A3	A3	A3	A3
		A2	A2	A2	A2	A1	A1	A1	A1
JCN	Salta a la dirección especificada por A2 A2 A2 A2 A1 A1 A1 A1 dentro de la misma ROM que contiene esta instrucción JCN, si se cumple la condición C1 C2 C3 C4, en caso contrario continúa ejecutando la próxima instrucción.	0	0	0	1	C1	C2	C3	C4
	C1=1: Invertir la condición de salto. C2=1: Saltar si el acumulador es cero. C3=1: Saltar si el acarreo vale uno. C4=1: Saltar si la pata TEST está a cer	A2	A2	A2	A2	A1	A1	A1	A1
JIN	Salto indirecto según el par de registros RRR	0	0	1	1	R	R	R	1

Tabla-2: Instrucciones que afectan al Contador de Programa.

D. UNIDAD ARITMÉTICA LÓGICA.

La unidad aritmética lógica (ALU) es la encargada de realizar las operaciones entre el Banco de Registros con el Acumulador. Las operaciones que se implementaron al ALU para el diseño del 4004 son:

- Borrar.
- Borrar acarreo.
- Incrementar.
- Complementar acarreo.
- Complementar.
- Rotar a la izquierda con acarreo.
- Rotar a la derecha con acarreo.
- Sumar acarreo.
- Decrementar.
- Restar acarreo.
- Establecer acarreo.
- Transferir.

La unidad de control es la encargada de definir la operación que va a realizar el ALU ya sea sobre el

Acumulador o sobre el Banco de Registros. Este circuito se diseño con el estilo algorítmico y el diagrama RTL se muestra en la figura-5.

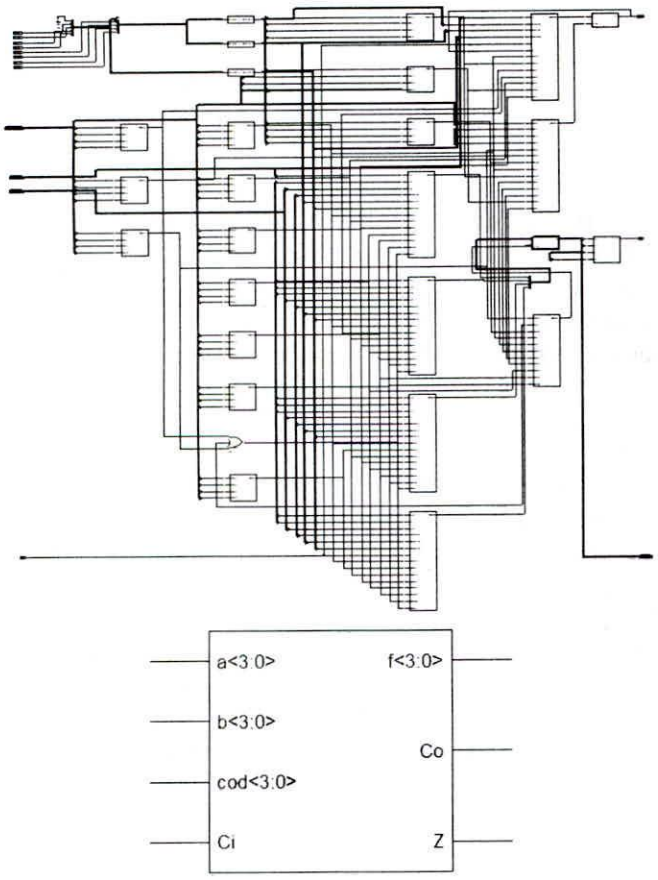

Fig. 5. Unidad Aritmética y Lógica.

Las entradas A y B del ALU reciben los datos a operar y obteniéndose el resultado en la salida F. Se puede apreciar en la figura-5 las salidas del acarreo (Co) y detección de cero (Z).

E. REGISTRO DE INSTRUCCIÓN

El registro de instrucción es el circuito que recibe la instrucción de la memoria de programa, formando parte de la interfaz con el exterior (ver figura-1). Es un registro de 16 bits capaz de cargarse en porciones de 4 bits, debido al tamaño máximo de las instrucciones. Se cargar de acuerdo a los bits de control que reciba de la unidad de control. Para instrucciones de tamaño de 8 bits sólo se utilizarán los 8 bits más significativos del registro de instrucción.

La entrada ENA habilita al registro de instrucción en el instante en que la unidad de control está en las etapas de búsqueda de una instrucción (FETCH) para enganchar las instrucciones que vienen de la memoria de programas. La entrada I selecciona en que porción del registro de instrucciones se van a cargar los códigos de operación (OPR) y operandos (OPA). Las entradas D y DATA son las entradas por donde se carga la instrucción.

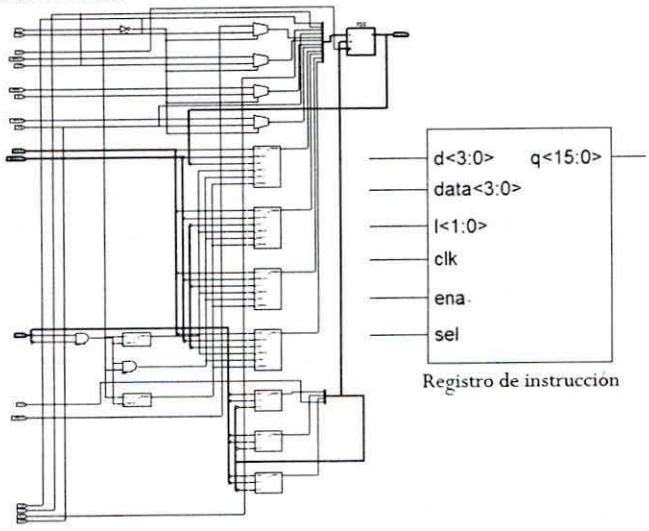

Fig. 6. Registro de Instrucción.

F. UNIDAD DE CONTROL

La unidad de control es el circuito mas importante dentro de los bloques del microprocesador, básicamente es una máquina de estado finita [4]. Es la parte del control del microprocesador que se encarga de gobernar todo el sistema digital. La unidad de control esta conectada con todos los demás subsistemas del microprocesador y se muestra en la figura 7.

La unidad de control se implementó con una maquina de estado de MOORE y consta de once estados. Estos son:

RESETE: reinicia el estado del microprocesador. Se llega a este estado con la activación del pin RESET.

FETCH1: inicia la lectura de una instrucción activando la ROM y la salida SYNC, también envía los 4 bits de mayor peso del PC.

FETCH2: Envía los siguientes 4 bits del PC. Se desactiva la señal SYNC.

FETCH3: Envía los 4 bits de menor peso del PC.

FETCH4: Ciclo de espera para que responda la memoria.

FETCH5: Se atrapa los primeros 4 bits de la instrucción.
FETCH6: Se atrapan los siguientes 4 bits de la instrucción.

DECODE1: Se deshabilita la ROM, el contador de programa apunta a la siguiente instrucción.

DECODE2: Se analizan los 4 bits del código de operación (OPR) de la instrucción y se define si la instrucción está completa (8bits) o falta extraer la otra parte de la instrucción (16 bits) saltando al ciclo EXECUTE1 ó FETCH1 respectivamente.

EXECUTE1: Se envían de los bits de la operación al ALU así como la selección de los operandos.

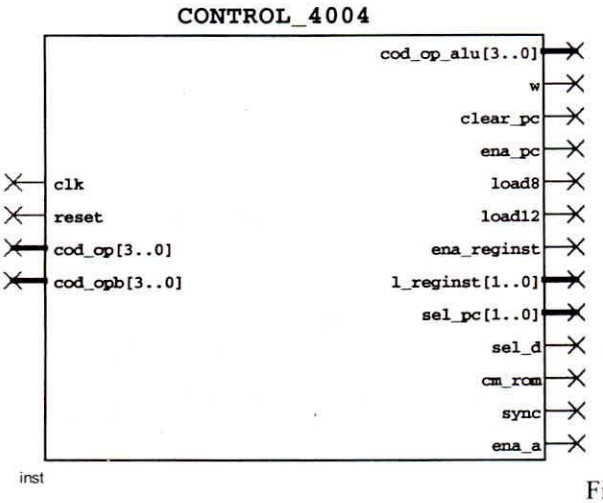

Fig. 7.

Unidad de Control.

EXECUTE2: Ciclo de espera para desactivar los operandos de destino y se regresa al ciclo FETCH1.

En la figura-7 se muestran las entradas y salidas de la unidad de control.

III. IMPLEMENTACION DEL MICROPROCESADOR 4004

Cada bloque fue simulado para verificar su correcto funcionamiento, para ello se utilizó la herramienta de síntesis QUARTUS II de la compañía ALTERA[5]. Se procedió a implementar el microprocesador 4004 realizando la interconexión de los bloques anteriormente diseñados utilizando el estilo estructural.

En la figura-8 se muestra el diagrama a nivel RTL de la interconexión de todos los bloques para formar el microprocesador 4004. En la figura-9 se indican las entradas y salidas del 4004. La entrada INST por donde ingresarán no sólo instrucciones de la memoria de programa, sino también los datos de la memoria de datos, la salida ADDRESS para generar las direcciones de las posiciones a acceder, la señal de RESET que reinicia la máquina de estado y borra los registros internos, la salida CM_ROM para activar a la memoria de programa de prueba y la señal SYNC que indica el inicio de un nuevo ciclo de instrucción.

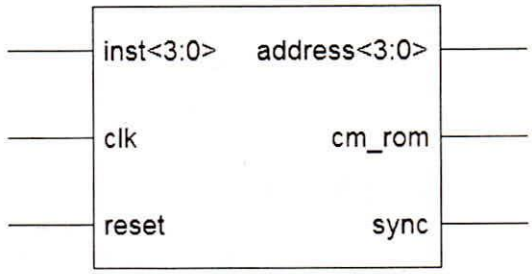

Fig. 9. Microprocesador 4004 implementado con VHDL.

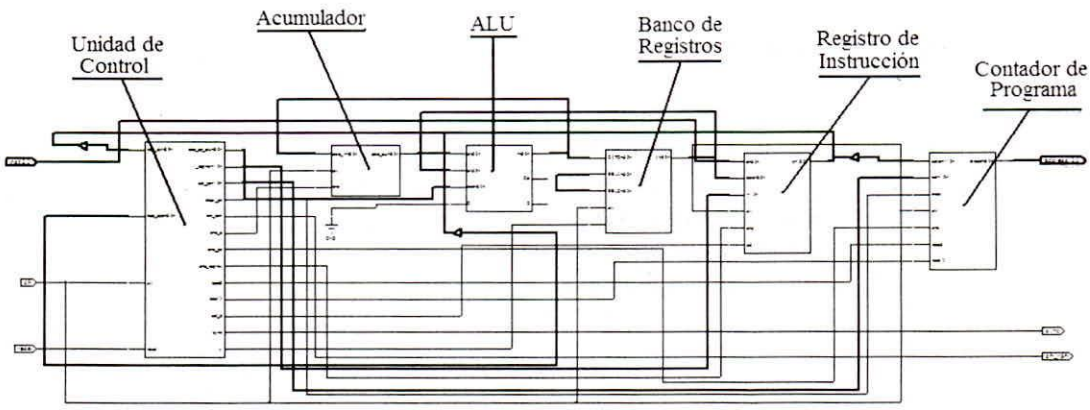

Fig. 8. Diagrama RTL mostrado por la herramienta de síntesis Project Navigator de XILINX donde se muestra la interconexión realizada en el estilo estructural para implementar el 4004.

Para verificar el funcionamiento del 4004 se diseñó una memoria de programa que tenga la capacidad de comunicación con el 4004, debía ser capaz de enviar instrucciones de 8 y 16 bits y recibir direcciones de 12 bits utilizando un sólo un camino de datos de 4 bits.

Para la implementación de la memoria ROM se utilizó un componente llamado: LPM_ROM de ALTERA, se le añadió el hardware necesario para realizar la comunicación descrita en el párrafo anterior.

El listado del programa de prueba cargado en la ROM se muestra en la tabla-3.

Se prueban algunas instrucciones que trabajan con como el acumulador y el banco de registros.

La interconexión del microprocesador 4004 y la memoria de instrucciones de prueba (llamada ROM_4004_CODE) se realizo en entorno gráfico del QUARTUS II y se muestra en la figura-10.

IV. SIMULACIÓN DEL MICROPROCESADOR 4004

La simulación se realizó por 12us, teniendo como periodo de reloj 100ns. En la figura-11 se muestra el diagrama de tiempos de las principales señales de procesador. Se puede apreciar como la señal SYNC se activa para indicar el inicio de una nueva búsqueda de instrucciones. Se puede apreciar también como el Acumulador se va actualizando a medida como se ejecutan las instrucciones de IAC (incremento), CMA (complemento), CLB (borrar), DEC (decrementar), RAL (rotar a izquierda), RAR (rotar a derecha); de la misma manera se puede apreciar como incrementaron los registros R1 y R2 con las dos últimas instrucciones del listado de la Tabla-3 (INC R1 y INC R2).

Direcc	Nmem.	Comentario	HEX	OPR	OPA
0000	IAC	Incrementa Acumulador	F2	1111	0010
0001	CMA	Complementa Acumulador	F4	1111	0100
0002	CLB	Borra Acumulador	F0	1111	0000
0003	DAC	Decrementa Acumulador	F8	1111	1000
0004	IAC	Incrementa Acumulador	F2	1111	0010
0005	IAC	Incrementa Acumulador	F2	1111	0010
0006	RAL	Rotar a la izq. Acumulador	F5	1111	0101
0007	RAR	Rotar a la der. Acumulador	F6	1111	0110
0008	INC R1	Incrementar Regitro1	61	0110	0001
0009	INC R2	Incrmentar Registro2	62	0110	0010

Tabla-3: listado del programa cargado en la ROM

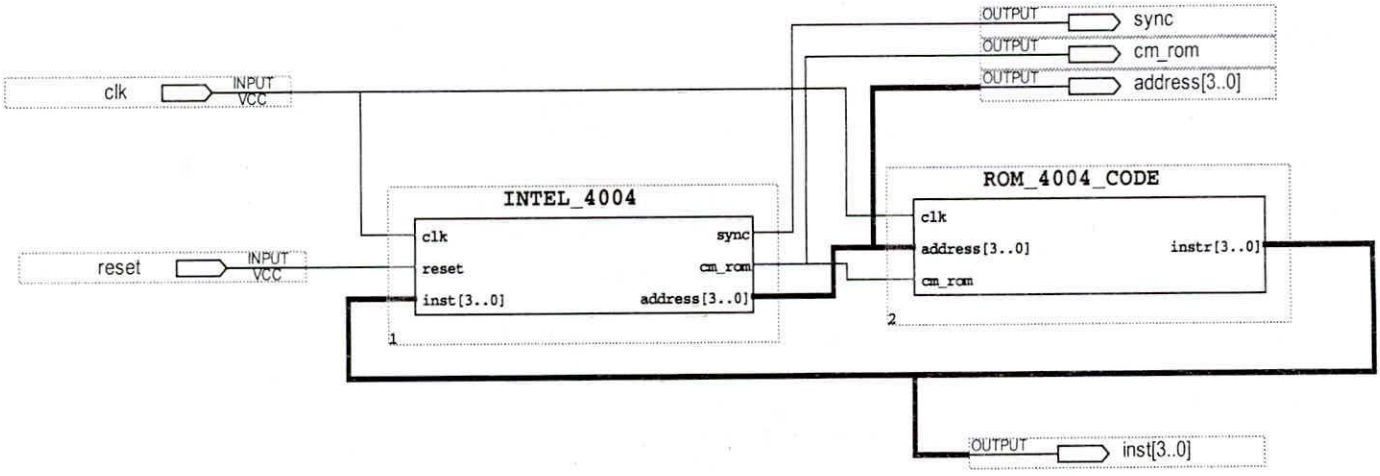

Fig.10. Microprocesador 4004 conectado a la memoria ROM de instrucciones.

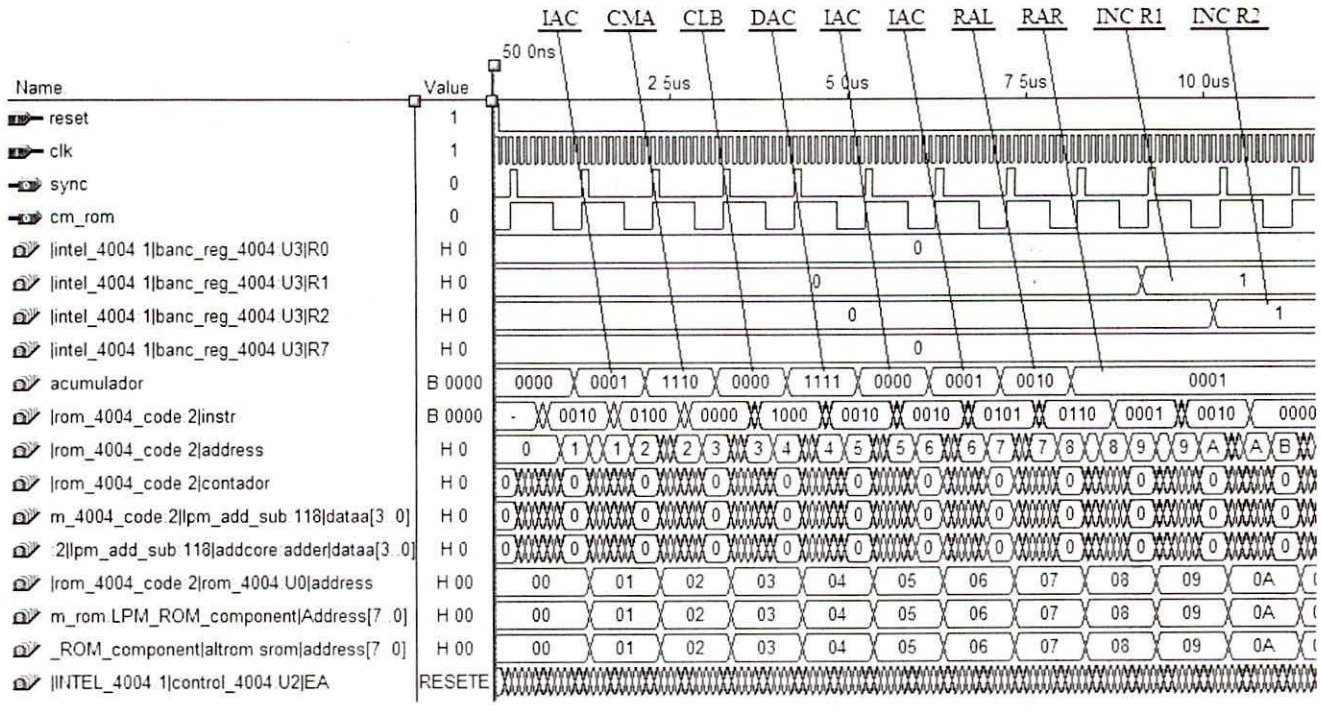

Fig.11. Simulación en el programa QUARTUS II del Microprocesador 4004 junto a una memoria ROM.

V. CONCLUSIONES

Se puede demostrar que con la ayuda de las herramientas de síntesis se logra una gran productividad en el diseño de sistemas digitales, ahorrando costos y tiempo de desarrollo. Si bien el diseño no llega a nivel de máscaras para el proceso de fabricación de chips, se logra tenerlo implementado a nivel de hardware ya sea en un CPLD ó como en un FPGA.

En esta oportunidad se utilizó el FLEX10K10 de la compañía ALTERA, que es un FPGA que contiene 576 celdas lógicas (equivalente a 10,000 compuertas), de los cuales el microprocesador diseñado ocupa 377 celdas lógicas (65%) conteniendo a un total de 126 flip flops.

El tiempo de diseño e implementación fue aproximadamente de 8 horas gracias a la experiencia en el manejo del lenguaje de descripción de hardware VHDL y de las herramientas de síntesis QUARTUS II de la compañía ALTERA y Project Navigator de la compañía XILINX. Cabe resaltar que el diseño se pudo llega a ser en otro lenguaje de descripción de hardware como es el VERILOG[7]

REFERENCIAS

[1] www.intel.com
[2] Arquitectura de computadoras, Morris Mano , Prentice Hall Hispanoamericana 1997.
[3] Circuit Design with VHDL, Volnei Predroni, Massachusetts Institute of Technology, 2004.
[4] Diseno Digital, Morris Mano, Prentice Hall Hispanoamericana 2003.
[5] www.altera.com
[6] www.xilinx.com
[7] Verilog HDL: A Guide to Digital Design and Synthesis, Samir Palnitkar, Prentice Hall PTR, 2003.