DISEÑO DE SITEMAS DIGITALES CON PLD'S

M.C. Juan Angel Garza Garza
Facultad de Ingeniería Mecánica y Eléctrica de la Universidad Autónoma de Nuevo León
Pedro de Alba s/n, Cd. Universitaria, San Nicolás de los Garza, Nuevo León
jagarza@uanl.mx

RESUMEN

En esta ponencia se discute el Diseño Digital mediante el uso de Dispositivos Lógicos Programables (PLD's), por medio de Programas de Captura Esquemática y Lenguajes de Descripción de Hardware (HDL's).

La capacidad de los PLD's permiten implementar una gran variedad de diseños y aplicaciones de la lógica combinacional y secuencial utilizando el mismo dispositivo para diferentes aplicaciones ya que es reprogramable, de tal forma que siguiendo un proceso sencillo, y en poco tiempo, se puede obtener un dispositivo a la medida (ASIC *Application Specific Integrated Circuits*) que antes solo era posible fabricarlo en la industria utilizando maquinaria especializada.

Este procedimiento de diseño resulta de gran utilidad en la enseñanza de la Electrónica Digital gracias a que los PLD's son económicos, los lenguajes de descripción de hardware poderosos y fáciles de programar.

1. INTRODUCCIÓN

La Electrónica, a través de un sin número de aplicaciones tecnológicas y con una rapidez arrolladora, ha logrado intervenir prácticamente en todos los ámbitos de la vida cotidiana de la sociedad contemporánea.

Tal vez sorprende que la palabra "Electrónica", que es hoy tan común, entró como vocablo del medio tecnológico en 1930, para abarcar la radio y las aplicaciones industriales de tubos de electrones (bulbos).

Lo que disparó verdaderamente el desarrollo de la electrónica, sin embargo, fue el invento del transistor en 1947. John Bardeen, Walter Brattain y William Shockley de los Laboratorios Bell, descubren el efecto transistor, el cual permite cambios en la conductividad de los materiales mediante el uso de corriente eléctrica.

Figura 1
George Boole George Boole

En la década de 1960 surge otro campo de la electrónica denominado Electrónica Digital, el cual se basa en el Álgebra Booleana creada por George Boole (1815-1864), lógico y matemático británico, que en 1854 escribió *Las leyes del pensamiento*, en el cual describe un sistema algebraico en donde las proposiciones lógicas se indican por símbolos y pueden relacionarse mediante operadores matemáticos abstractos que corresponden a las leyes de la lógica. La principal característica del álgebra booleana es que las variables sólo pueden tomar dos valores: Uno o Cero (Verdadero o Falso).

Heidi Elliot de la revista Electronic Business opina que *lo que empezó como dispositivo para reemplazar el tubo de vacío se puede comparar ahora con el invento de la rueda*.

2. ASICS

Desde los finales de la década de 1970, los equipos electrónicos digitales utilizan Circuitos Integrados (*CI* o *CHIPS*) de función lógica fija, realizados en pequeña o mediana escala de integración (SSI, MSI).

Para la implementación de aplicaciones muy complejas, que requieren de una gran cantidad de circuitos de función fija, resulta más conveniente intégralos en un solo dispositivo fabricado a la medida, los cuales son llamados: ASICS, *Application Specific Integrated Circuits.* (Circuitos Integrados de Aplicación Especifica o circuitos a la medida).

Entre las ventajas que presenta el uso de los ASICs podemos mencionar que: Ahorran espacio, reducen el número de dispositivos, menor costo, reducen el tiempo de ensamble, bajo consumo de potencia, menor calentamiento, facilidad en la verificación (control de calidad), mejor confiabilidad.

Los ASIC se pueden clasificar por su tecnología de fabricación en cuatro categorías: Arreglos de Compuertas, Celdas Estándar, Full Custom y Lógica Programable

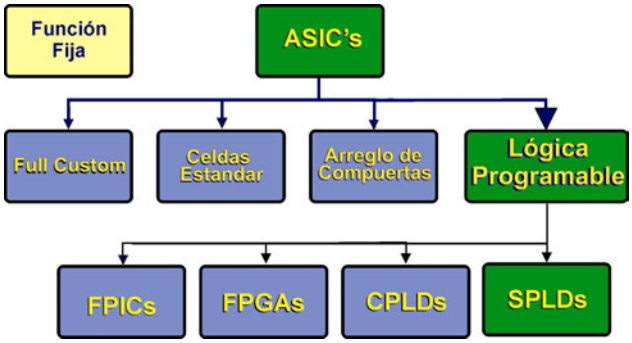


Figura 2 Clasificación de los ASICS

Las tecnologías de Arreglos de Compuertas, Celdas Estándar y Full Custom, están encaminadas a la producción industrial de alto volumen y requieren de equipo especializado para la fabricación del ASIC.

Por otro lado, con la Lógica Programable es posible diseñar e implementar desde un solo circuito con el uso de solamente una computadora, un programador y software de Diseño Electrónico Asistido EDA (Electronic Design Assistant).

3. PLD

Un dispositivo de lógica programable (PLD) es un Circuito Integrado cuya estructura lógica final es directamente configurada por el usuario, sin necesidad de llevar a cabo ningún proceso de fabricación.

Peggy Aycinena de la revista electrónica Integrated System Design asegura que los dispositivos lógicos programables son la ola del futuro porque presentan las siguientes características: 10,000 compuertas en 1 in^2, entradas y salidas configurables, reprogramables, programado remotamente para diferentes funciones.

Los PLDs facilitan el proceso de diseño y reducen el tiempo de desarrollo, cuando se requieren prototipos o producción de baja escala, pues todo el proceso se puede llevar a cabo con la ayuda de una computadora personal, programas de aplicación y el programador los cuales actualmente están disponibles a bajo costo.

Algunos fabricantes de PLD's son: Actel (www.actel.com), Altera Corp. (www.altera.com), Atmel Corp. (www.atmel.com), Chip Express (www.chipexress.com), Cypress Sem.(www.cypress.com) , Lattice Sem. (www.latticesemi.com), Quicklogic Corp. (www.quicklogic.com), Xilinx Inc. (www.xilinx.com).

Los diferentes tipos de dispositivos de lógica programable que existen hoy en día pueden clasificarse por su tecnología, su capacidad (Figura 2):

- Simplex Programmable Logic Device SPLDs.
- Complex Programmable Logic Device CPLDs.
- Field Programmable Gate Arraysevice FPGAs.
- Field Programmable Inter Connect FPICs.

De la clasificación anterior en este documento sólo nos enfocaremos, como ejemplos a los Simplex Programmable Logic Device SPLDs.

Los SPLDs están constituidos por un arreglo de compuertas AND, seguido por otro arreglo de compuertas OR, uno o ambos arreglos programables, algunos incluyen Flip Flops.

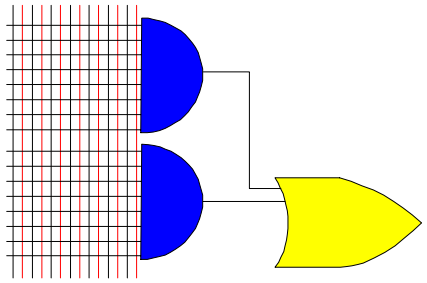


Figura 3 Arreglo And Or de un SPLD.

A su vez los SPLDs se pueden clasificar según su estructura interna en:

- PAL Programmable Array Logic, VANTIS.
- GAL Generic Array Logic, Lattice Semiconductor.
- PLA Programmable Logic Array.
- PLD Programmable Logic Device.

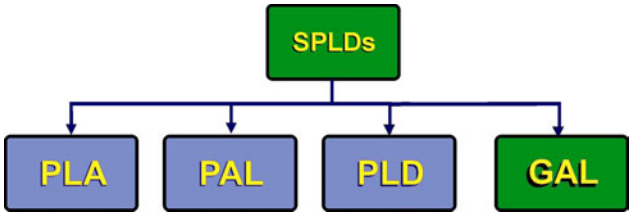


Figura 4 Clasificación de los SPLDs

De estos tipos de SPLDs, el GAL destaca por su bajo precio y versatilidad por lo que lo describiremos en el siguiente punto.

4. GAL

GAL (Generic Array Logic), en español Arreglo Lógico Genérico, son un tipo de circuito integrado, de marca registrada por Lattice Semiconductor, que ha sido diseñados con el proposito de sustituir a la mayoría de las PAL, manteniendo la compatibilidad de sus terminales.

El GAL básicamente está formado por una matriz AND reprogramable y una matriz OR fija con configuración programable de salidas y/o entradas.

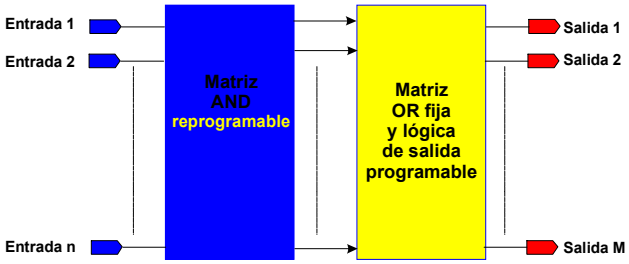


Figura 5 Estructura básica de un GAL

Como un ejemplo de las características ofrecidas por este tipo de dispositivos, a continuación se enlistan las especificaciones más relevantes del circuito GAL16V8 de marca Lattice Semiconductor.

- fmax = 250 Mhz.
- 3.5 ns máximo tiempo de propagación.
- 2.5 ns máximo tiempo de propagación de la entrada de reloj a el dato de salida.
- Celdas Reprogramables.
- Vcc = 5 Volts ± 5%
- Consumo de corriente 90 mA.
- Rapidez en el borrado < 100 ms.
- 20 años de retención de los datos.
- 8 Output Logic MacroCells (OLMC)
- Polaridad de salida Programable.
- Temperatura de operación de 0 a 75 ° C.

En el GAL16V8, las terminales tienen las siguientes funciones: La terminal 1 es la entrada de CLK, las terminales 2 a la 9 son ocho Entradas fijas, la terminal 10 de GND , la terminal 11 como una Entrada de control O/E (Output/Enable), de la 12 a la 19 ocho terminales programables OLMC y terminal 20 de VCC .

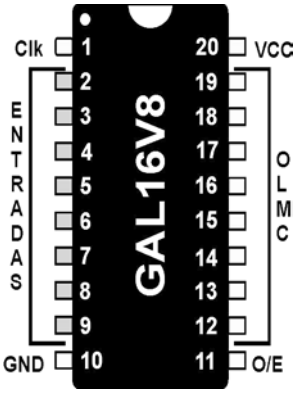


Figura 6 Distribución de terminales del un GAL16V8

Las terminales de la 12 a 19 correspondientes al OLMC (*Output Logic Macrocell)* pueden programarse para trabajar como entradas y/o salidas, y en el caso de ser usadas como salidas estas pueden ser combinacionales o registradas (Flip Flops), lo cual le da la versatilidad de ser programado de diferentes formas y para diferentes requerimientos.

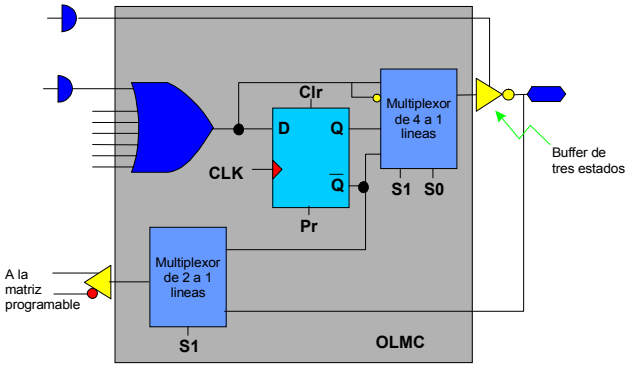


Figura 7 Configuración interna de la OLMC de un GAL16V8.

La programación de los PLDs en general se lleva a cabo por medio de programas de aplicación especializados siendo las dos estrategias de programación mas utilizadas la captura esquemática y la de lenguaje de descripción de hardware (HDL).

La gran ventaja de estas herramientas es el de hacer los diseños en la computadora, donde los errores son fácilmente detectables y corregibles.

5. CAPTURA ESQUEMÁTICA.

Por Captura Esquemática se entiende el proceso de descripción, mediante un dibujo, de un circuito eléctrico, que representan a los diferentes componentes del circuito y solo se efectúan interconexiones entre ellos.

Existen varios programas aplicación de Captura Esquemática como el "Schematic" del IspStarter de Lattice Semiconductor o "Fundation" de XILINX entre otros.

Esta técnica permite simular en la computadora el circuito virtualmente y verificar su funcionamiento antes de su fabricación o implementación en un PLD, reduciendo así el ciclo de diseño y el tiempo de obtención de un producto.

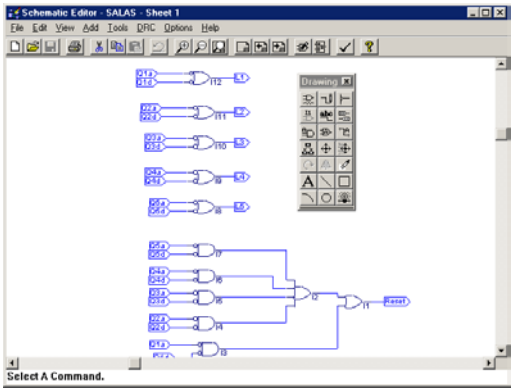
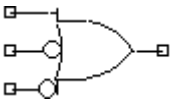


Figura 8 Programa de Captura Esquemática.

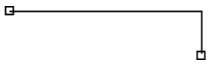
La desventaja es que en el diseño de circuitos grandes no es posible comprenderlos debido a que hay demasiados componentes e interconexiones en la pantalla.

Los cuatro componentes básicos de la captura esquemática son: Símbolos, Conectores, Etiquetas y Puertos de Entrada y/o Salida

Símbolos es una representación gráfica de los componentes.

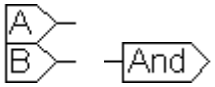


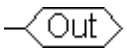
Conectores (alambre), permite la interconexión entre las terminales de los símbolos, o Dispositivos de entrada/salida.



Etiquetas (Variables), Son los nombres para la identificación de las entradas o salidas.

Puertos de entrada/salida es la definición de las características de un Puerto de Entrada, Salida o Puerto Bidirecional.





Utilizando el programa IspStarter de Lattice Semiconductor, el primer paso en el proceso de obtener el circuito es precisamente la captura esquemática en donde se genera un archivo de extensión .SCH, posteriormente se pasa al proceso de enlace (link) en donde se valida si los componentes son permitidos en el dispositivo seleccionado, además si el dispositivo tiene la capacidad de integrar los elementos requeridos (Fit Design). También es posible efectuar la simulación antes de obtener el circuito final con la intención de asegurar que este cumpla con los requerimientos solicitados.

En el proceso de enlace se genera un archivo reporte con extensión .REP que nos indica, entre otras cosas, la asignación de terminales (Pin Out) y el archivo JEDEC con el cual efectuaremos la programación del circuito a través de un programador.

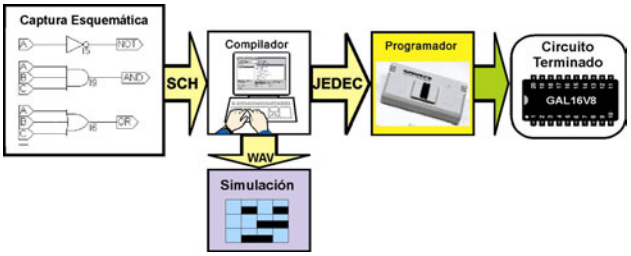


Figura 9 Diagrama del proceso de diseño digital por medio de captura esquemática.

6. LENGUAJES DE DESCRIPCIÓN DE HARDWARE.

Otra forma alternativa de diseño es usar un Lenguaje de Descripción de Hardware" (HDL), en donde se describe el funcionamiento a través de instrucciones de alto nivel en forma similar a un lenguaje de programación.

Los HDL's más usados son:

- ABEL, Advanced Boolean Expression Language.
- VHDL, en donde el acrónimo tiene dos raices la V de VHSIC (Very High Speed Integrated Circuits) y HDL (Hardware Description Language).
- VERILOG idéntico en función al VHDL

El diseño utilizando HDL's posee varias ventajas sobre la metodología tradicional de diseño a nivel compuerta, algunas de estas ventajas son listadas a continuación .

- Es posible verificar el funcionamiento del sistema dentro del proceso de diseño sin necesidad de implementar el circuito.

- La simulación del diseño, antes de que éste sea implementado mediante compuertas, permitendo probar la arquitectura del sistema para tomar decisiones en cuanto a cambios en el diseño.

- Esta metodología elimina el antiguo método tedioso de diseño mediante compuertas, reduce el tiempo de diseño y la cantidad de errores producidos por el armado del circuito.

- Un circuito hecho mediante una descripción en un HDL puede ser utilizado en cualquier tipo de dispositivo programable capaz de soportar la densidad del diseño. Es decir, no es necesario adecuar el circuito a cada dispositivo porque las herramientas de síntesis se encargan de ello.

En este documento se describe la programación usando el lenguaje ABEL-HDL.

7. ABEL-HDL

ABEL, Advanced Boolean Expression Language (Lenguaje Avanzado de Expresiones Booleanas), fue desarrollado por Data I/O Corporation para la implementación de funciones booleanas en dispositivos lógicos programables.
ABEL puede ser usado para describir el comportamiento de un sistema digital partiendo de:

- Ecuaciones Booleanas.
- Descripción del comportamiento usando instrucciones WHEN-THEN
- Tablas de Verdad.
- Tablas de Estado
- Diagramas de Transición

ABEL es un archivo de texto que contiene los siguientes elementos.

1	Documentación incluyendo nombre del programa y comentarios.
2	Declaraciones que identifican las entradas y salidas de las funciones lógicas que serán efectuadas.
3	Instrucciones que especifican las funciones lógicas que serán efectuadas.
4	Declaración del tipo de dispositivo en que las funciones lógicas especificadas se implementaran.
5	Vectores de prueba que especifican las salidas esperadas de las funciones lógicas para ciertas entradas

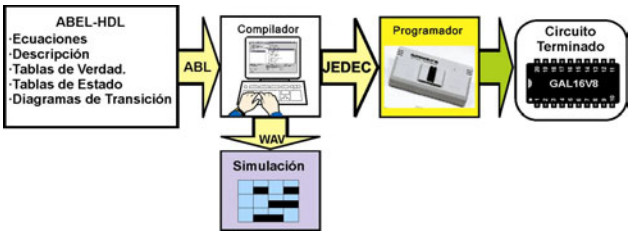


Figura 10 Diagrama del proceso de diseño por medio de HDL.

Las operaciones boolanas pueden ser representadas fácilmente en el lenguaje ABEL-HDL

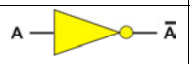
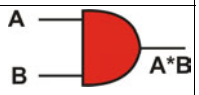
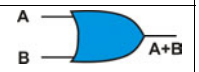
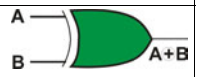
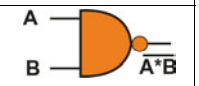
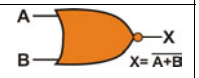
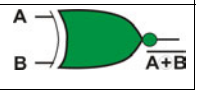
Operador	Descripción	Ecuación	Símbolo
!	NOT	!A	
&	AND	A&B	
#	OR	A#B	
$	EXOR	A$B	
!&	NAND	!(A&B)	
!#	NOR	!(A#B)	
!$	EXNOR	!(A$B)	

Figura 11 Expresiones usadas en ABEL-HDL para las operaciones Booleanas.

La estructura básica del formato de un archivo en ABEL-HDL incluye:

1	**Module**, Inicio del programa
2	**Title** Líneas de titulo y comentarios (opcional)
3	**Declaration** Asignación de las terminales de entrada y salida del dispositivo.
4	**Descripción lógica** (ecuaciones, tablas de verdad, etc.)
5	**Test_vectors** Vectores de prueba (opcional)
6	**End**, Final del programa

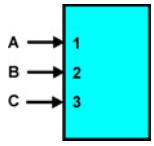
Para la construcción del archivo en ABEL-HDL se requiere que:

1.- Todo programa debe contener al inicio la instrucción MODULE y al final END que indican el principio y el final del programa.

2.- Los comentarios y las líneas de título son opcionales, pero es conveniente utilizarlos para describir el funcionamiento y partes del programa, estos deberán de empezar con comillas (“) ejemplo “Entradas.

3.- Declaraciones (Declarations) por medio de este comando podemos declarar las entradas y salidas del sistema.

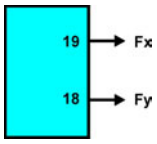
La declaración de las variables de entrada así como la asignación de terminales dentro de un GAL, como se muestra en la figura, se describe en la siguiente línea: **A, B, C PIN 1,2,3;** .



Observe que las variables de entrada están separadas por comas (,), en seguida el comando PIN y la terminal correspondiente a cada variable además se cierra la instrucción con punto y coma (;)

Es necesario que la terminal asignada como entrada cumpla con esa función en el GAL. En el caso del GAL16V8 tiene 8 terminales definidas como entradas exclusivas 2,3,4,5,6,7,8,9 además la terminal 1 puede ser usada como entrada o señal de reloj (Ck) y otras 8 que pueden ser entradas o salidas 11,12,13,14,15,16,17,18 y 19.

Al definir las variables de salida así como su asignación de terminales es necesario incluir el comando **ISTYPE** ‘com’ para indicar que son salidas y combinacionales. Ejemplo: **FX,FY PIN 19,18 ISTYPE ‘COM’**; .



Las terminales designadas como salidas en un GAL16V8 son 12,13,14,15,16,17,18 y 19.

Descripciones Lógicas: en esta sección se pueden usar los comandos:

EQUATIONS: permite expresar las ecuaciones.

TRUTH_TABLE: permite declarar una Tabla de Verdad o Tabla de Estados.

WHEN y THEN: permite referir el comportamiento en algunos casos.

STATE_TABLE: permite describir el comportamiento del diagrama de transición de un sistema secuencial.

Algunas instrucciones como **If, Then, Else, Goto** son recursos importantes en la simplificación de la programación de los sistemas secuenciales.

Test_Vectors: **(**Vectores de Prueba:) esta parte es opcional y podemos efectuar la comprobación o simulación del diseño sin necesidad de implementarlo.

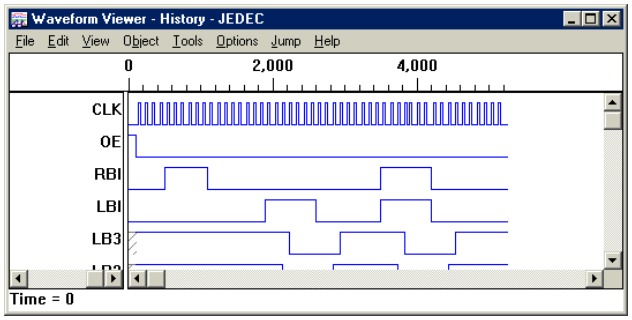


Figura 12 Gráfica de la Simulación de un sistema secuencial.

A continuación detallaremos la estructura básica de un archivo en formato ABEL-HDL utilizando ecuaciones Booleanas.

Encabezado	MODULE EQ
Declaraciones	Declarations “Entradas A,B,C PIN 1,2,3; “Salidas FX,FY PIN 19,18 ISTYPE ‘COM’;
Descripciones Lógicas	EQUATIONS FX= A & ¡B & C # ¡B & C; FY = (A # !B # C) & (A # !C);
Vectores de Prueba	TEST_VECTORS
	([A,B,C]->[FX,FY]) [0,0,0]->[.X.,.X.]; [0,0,1]->[.X.,.X.]; [0,1,0]->[.X. ,.X.]; [0,1,1]->[.X. ,.X.]; [1,0,0]->[.X. ,.X.]; [1,0,1]->[.X. ,.X.]; [1,1,0]->[.X. ,.X.]; [1,1,1]->[.X. ,.X.];
Final	END

A continuación se mostrara la estructura básica de un archivo, en formato ABEL-HDL, utilizando Tabla de verdad incluyendo el Test_vectors

```
MODULE pistas
"Entradas
        A,B,C PIN 1,2,3;
"Salidas
DC9,B747 pin 19,18 istype 'com';
TRUTH_Table
([A,B,C]->[DC9,B747])
[0,0,0]->[0,0];
[0,0,1]->[1,0];
[0,1,0]->[1,0];
[0,1,1]->[0,1];
[1,0,0]->[1,0];
[1,0,1]->[1,0];
[1,1,0]->[0,1];
[1,1,1]->[1,1];
test_vectors
([A,B,C]->[DC9,B747])
[0,0,0]->[0,0];
[0,0,1]->[1,0];
[0,1,0]->[1,0];
[0,1,1]->[0,1];
[1,0,0]->[1,0];
[1,0,1]->[1,0];
[1,1,0]->[0,1];
[1,1,1]->[1,1];
end
```

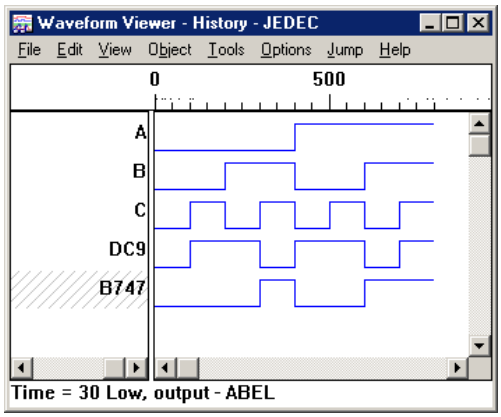


Figura 13 Gráfica de la Simulación del sistema combinacional.

A continuación se muestra un archive en formato ABEL-HDL para un Multiplexor de ocho a una línea usando los comandos WHEN y THEN.

```
MODULE muxeq
" Entradas de datos
        L0..L7 pin 1..8;
"Entradas de control
        A,B,C pin 19,18,17;
"Salida
        Y pin 16 istype 'com';
Equations
        WHEN !A&!B&!C THEN Y=L0;
        WHEN !A&!B&C THEN Y=L1;
        WHEN !A&B&!C THEN Y=L2;
        WHEN !A&B&C THEN Y=L3;
        WHEN A&!B&!C THEN Y=L4;
        WHEN A&!B&C THEN Y=L5;
        WHEN A&B&!C THEN Y=L6;
        WHEN A&B&C THEN Y=L7;
END
```

Para los sistemas secuénciales sincronos en el lenguaje ABEL-HDL se puede utilizar las ecuaciones de los Flip Flops, La tabla de estados o la descripción del problema, a continuación se presenta un archivo de un sistema secuencial:

```
MODULE FER
"ENTRADAS
Clk, A,B,C,R PIN 1..5;
"SALIDAS COMBINACIONALES
        FA,FB,FC PIN 19..17 ISTYPE 'COM';
"SALIDAS REGISTRADAS
        Q1,Q0 PIN 16,15 ISTYPE 'REG';
```

```
"SINCRONIZACION DE LOS FLIP FLOPS
SREG=[Q1,Q0];
EQUATIONS
SREG.CLK=Clk;
```

```
declarations
E0=[0,0];
E1=[0,1];
E2=[1,0];
E3=[1,1];
```

```
STATE_DIAGRAM SREG
STATE E0:
FA=0;FB=0;FC=0;
IF !A&!B&!C&!R THEN E0;
IF A&!B&!C&!R THEN E1;
IF !A&B&!C&!R THEN E2;
IF !A&!B&C&!R THEN E3 ELSE E0;
STATE E1:
FA=1; FB=0; FC=0;
IF R THEN E0 ELSE E1;
STATE E2:
FA=0; FB=1; FC=0;
IF R THEN E0 ELSE E2;
STATE E3:
FA=0; FB=0; FC=1;
IF R THEN E0 ELSE E3;
End
```

8. CONCLUSIÓN

Como hemos comentado a través del documento estas técnicas de diseño digital presentan grandes ventajas sobre las tradicionales, podemos mencionar algunas como: La rapidez del diseño, la facilidad de corregir errores, el de poder implementar en un solo circuito la aplicación además de la facilidad con que se aprenden y dominan estas técnicas entre otras. En caso de que requiera mas información puede consultar nuestra pagina WEB *http://* jagarza.fime.uanl.mx.

9. AGRADECIMIENTOS

Al *M.E.C. Rogelio Garza Rivera*, *M.C. Luis Manuel Martínez Villarreal, Ing. Fernando J. Elizondo Garza y M.C. Julián Eduardo Hernandez Venegas* por su apoyo y colaboración en la elaboración de este documento.

10 REFERENCIAS

Libros:

Diseño Digital Jhon F. Wakerly Prentice-Hall, 2001.

Digital Systems Principles and applicatons Tocci and Widmer Prentice-Hall, 2001.

Digital Fundamentals Floyd Prentice-Hall, 2000

Digital System Design Using VHDL Charles H.Roth International Thomson Publishing Company. 1998

Prácticas de Laboratorio de Electrónica Lógica I Garza, Hernandez, Cantu y Martínez U.A.N.L. F.I.M.E 2002

Paginas WEB

http://www.seas.upenn.edu/~ee200/
http://jagarza.fime.uanl.mx

.