Facultad de Ingeniería Eléctrica

Laboratorio de Electrónica

"Ing. Luís García Reyes"

Materia: "Laboratorio de Electrónica Digital I"

Práctica Número 11

"Dispositivos programables"

Objetivo:

Aplicación de un dispositivo programable (ROM o PLD) para resolver problemas de diseño de circuitos combinacionales.

Introducción:

El contar con dispositivos programables, permite "fabricar" dispositivos personales, los cuales son "hechos" de acuerdo a nuestras necesidades. Dentro de los dispositivos programables que se pueden utilizar para la solución de problemas de diseño, podemos encontrar a dos dispositivos básicos como son las memorias (ROM, EPROM, EEPROM, etc.) o bien los arreglos lógicos programables (PAL, GAL's, etc.)

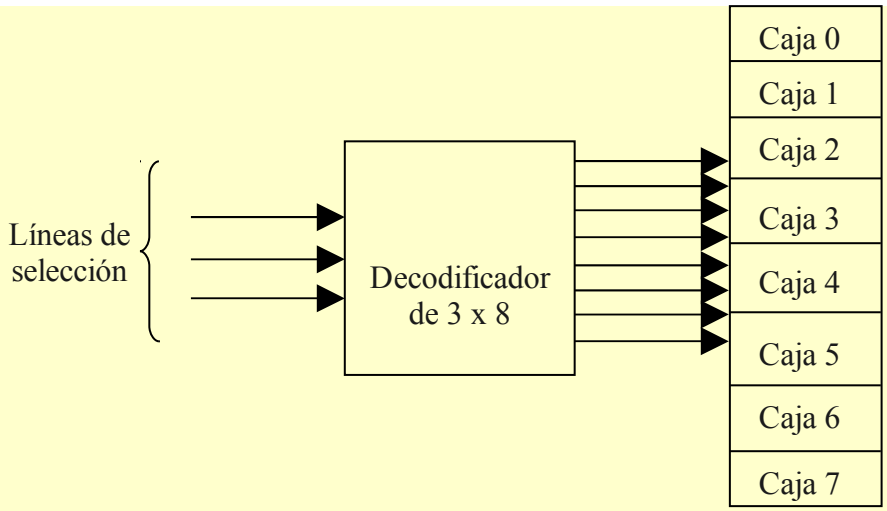
Memorias:

Una memoria digital, es un dispositivo al cual se le transfiere la información binaria que se desea almacenar, y posteriormente se puede leer la información que se coloco previamente en la misma. En muchos de los casos la memoria almacena información en forma de bits.
A una unidad de memoria se le puede ver como una "caja" en donde podemos almacenar información y posteriormente utilizarla.

CajaBit 1	1
CajaBit 2	0
……	..
.....	..
CajaBit n	Dato n

Para seleccionar cada una de estas cajas, se utiliza un circuito, el cual permite seleccionar cada una de las caja, para realizar esta función se requiere de un circuito codificador de direcciones, el cual es básicamente en un mutliplexor.

De manera general, si se tienen N líneas de selección, se pueden tener hasta 2^N Cajas-bits.



En la gráfica anterior se representa como se almacena la información en cada una de las cajas, en este caso en cada una se almacena un solo bit. Sin embargo para propósitos de manejo, el almacenamiento se agrupa en bloques de 1, 4, 8, 16, 32 o 64 bits. Así una de estas agrupaciones se puede ver como una caja con "divisiones" internas en donde cada división interna almacena un bit. A continuación se muestra una representación gráfica de un grupo de bits.

CajaByte 1	1	1	0	0	1	1	1	0
CajaByte 2	0	0	0	1	1	0	1	1

En cada uno de las "cajas" se encuentran bits (con valores de 0 ó 1), estos datos debe de colocarse previamente en las cajas mediante algún método de grabación. Los datos que se encuentran en cada una de estas cajas, pueden ser utilizados cuando sean requeridos. Para leer estos datos, es necesario tener un método para acceder a cada uno de las "cajas", para seleccionar cada una se utilizan circuitos codificadores mas conocidos como Demultiplexores.

Implementación de circuitos combinacionales utilizando memorias.

Si se tiene cuidado se puede ver que un decodificador genera 2^N términos, de las N variables de entrada, agregando compuertas OR, para sumar los minitérminos de funciones booleanas, es posible generar cualquier circuito combinacional. Entonces una memoria programable, es en esencia un dispositivo que incluye tanto el decodificador como las compuertas OR dentro de un solo dispositivo.

Si elegimos conexiones para los minitérminos incluidos en la función, es posible programar las salidas de la memoria de modo que representen las funciones booleanas de las variables de salida de un circuito combinacional.

Lógica programable.

Como su nombre lo indica, es una familia de componentes, los cuales contienen arreglos de elementos lógicos (AND, OR, NOT,LATCH, FLIP-FLOP's) y pueden ser configurados dentro de cualquier función lógica que el usuario requiera. Existen varios tipos de dispositivos programables (ASIC'S, FPGA´s, PLA's, PROM's, PAL's, GAL's, y PLD complejas)

Los ASIC's son circuitos integrados de aplicación especifica, son mencionados aquí debido a que son dispositivos definibles por el usuario. A diferencia de otros dispositivos,

pueden contener funciones analógicas, digitales y combinaciones de ambas. En general estos dispositivos son programables utilizando una mascara y no es programable por el usuario. Esto significa que el fabricante puede configurar con las especificaciones del usuario. Este circuito es utilizado para combinar una gran cantidad de funciones lógicas dentro de un dispositivo. Sin embargo, tiene un costo inicial muy alto, debido a esto, es utilizado solo cuando son necesarias grandes cantidades. Como es posible imaginar, debido a la naturaleza de las ASIC's. Los lenguajes de programación de dispositivos programables comerciales no soportan a este tipo de dispositivos.

Arquitectura básica de un dispositivo programable.

Antes que nada, un dispositivo programable por el usuario, es uno que contienen una arquitectura general predefinida, en la cual un usuario puede programar el diseño dentro de él usando un conjunto de herramientas de desarrollo. La arquitectura general puede variar pero comúnmente consta de uno o mas arreglos de términos AND y OR para la implementación de funciones lógicas. Muchos dispositivos contienen combinaciones de flip-flop's y latches, los cuales pueden ser utilizados como elementos de almacenamiento para las entradas y salidas del dispositivo. Dispositivos mas complejos contienen macroceldas. Las macroceldas permiten al usuario configurar el tipo de entradas y salidas que son necesarias para el diseño.

PROM's

Las PROM's son memorias programables de solo lectura. A pesar que el nombre no implica lógica programable, las PROM's de hecho son elementos lógicos, la arquitectura de la mayoría de las PROM's está constituida típicamente de un número fijo de arreglos de términos AND que proporcionan un arreglo programable. Generalmente es utilizado para decodificar combinaciones específicas de entradas en salidas de funciones como un mapa de memoria en un entorno de microprocesadores.

PAL's

Las PAL's son dispositivos de arreglos lógicos programables. La arquitectura interna consiste en términos AND programables proporcionando términos OR fijos. Todas las entradas al arreglo pueden ser alambradas a compuertas AND, pero términos específicos AND son dedicados a términos específicos OR. Las PAL's tienen una arquitectura muy popular y es probablemente el dispositivo lógico programable más utilizado. Si un dispositivo contiene macroceldas, este usualmente tiene una arquitectura PAL. Típicamente las macroceldas pueden ser programadas como entradas, salidas o entrada/salida utilizando la habilitación del tercer estado. Normalmente tienen registros de salida los cuales pueden o no ser utilizados en conjunto con el pin de entrada/salida asociado. Otras macroceldas tienen más de un registro, varios tipos de retroalimentación en los arreglos y ocasionalmente retroalimentación entre macroceldas. Estos dispositivos son principalmente utilizados para remplazar múltiples funciones lógicas TTL, comúnmente denominadas "lógica alambrada"

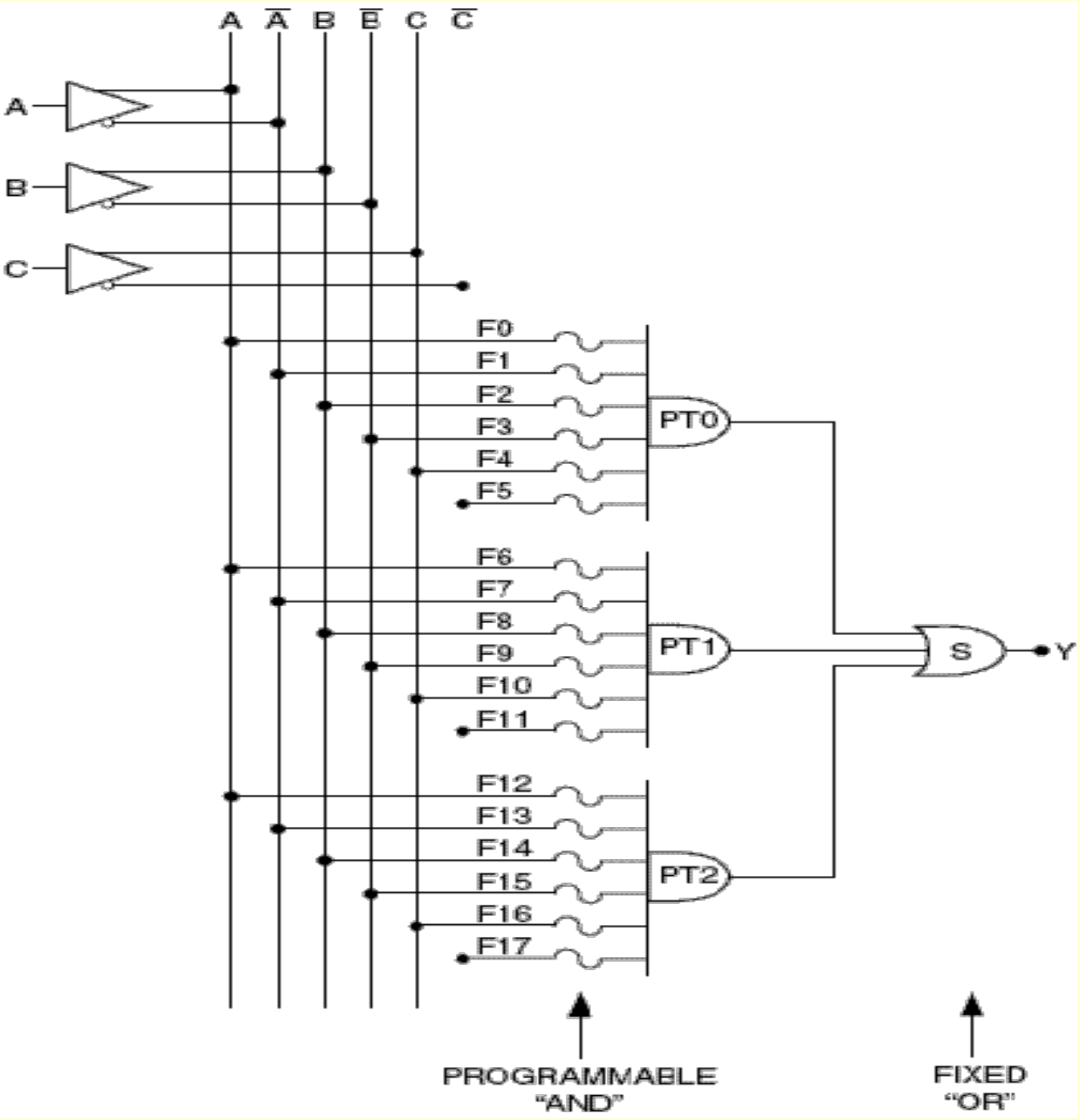
GAL's

Las GAL's son arreglos lógicos genéricos, están diseñados para "emular" muchas PAL's comunes utilizando macroceldas. Si un usuario tienen un diseño utilizando varios tipos de PAL's, comunes, se pueden configurar varias GAL's similares para emular cada uno de los otros dispositivos. Esto puede reducir los diferentes tipos de dispositivos adquiridos y solo mantener un solo tipo de modelo. Permitiendo abaratar costos, ya que de manera general una gran cantidad del mismo dispositivo decrementa los costos de adquisición y almacenamiento. Estos dispositivos son "REPROGRAMABLES" y borrables eléctricamente lo cual los hace muy útiles a los ingenieros de diseño.

Arreglo de lógica programable (PLA)

Los PLA's son arreglos lógicos programables. Estos dispositivos contienen los dos tipos de arreglos programables AND y OR, que permiten a cualquier termino AND proporcionar cualquier termino OR. Los PLA's son probablemente los que tienen la mayor flexibilidad que los dispositivos anteriores, conservando la funcionalidad lógica. Las PLA's típicamente cuentan con retroalimentación del arreglo OR de regreso al arreglo AND que puede ser utilizado para implementar máquinas de estado asíncronas, sin embargo muchas de las máquinas de estados son implementadas como máquinas asíncronas. Con esta idea en mente, los fabricantes crearon un tipo de PLA llamado SEQUENCER, el cual tienen un registro de retroalimentación de salida del arreglo OR dentro del arreglo AND.

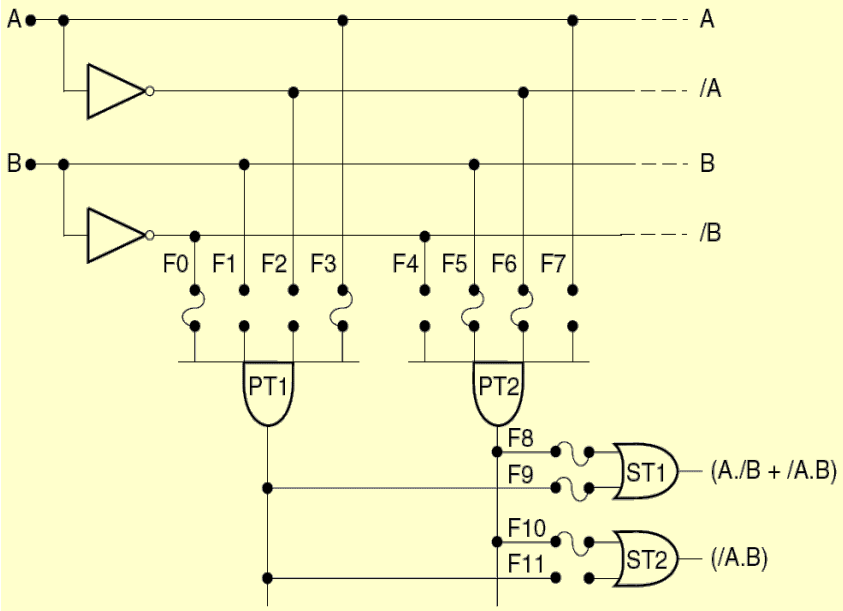
El arreglo de la lógica programable es similar a la memorias es su concepto, solo que el PLA no efectúa la decodificación cabal de las variables **ni genera todos los minitérminos**. El codificador de sustituye por un arreglo de compuertas AND que se programan para generar cualquier término producto de las variables de entrada. Luego, los términos producto se conectan a compuertas OR para formar la suma de productos de las funciones booleanas.

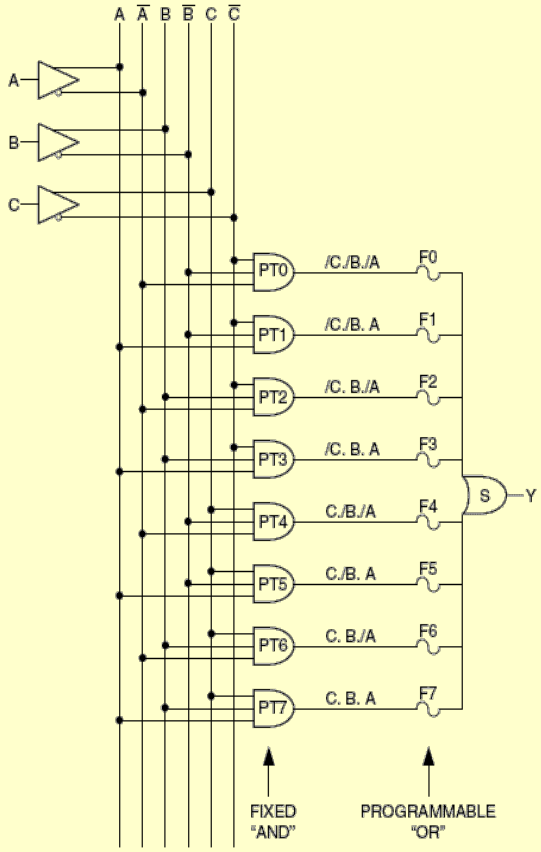


El arreglo lógico programable (PAL) es un dispositivo lógico programable con un arreglo OR fijo y un arreglo AND programable. Dado que solo las compuertas AND son programables, el PAL es más fácil programar, pero no es tan flexible como el PLA

Arreglos lógicos genéricos

Los arreglos lógicos genéricos (GAL). Fueron diseñados para emular muchos de las PAL´s más comunes utilizando macroceldas, Estos circuitos permiten sustituir el uso de PAL´s utilizando GAL´s, ya que se pueden configurar una GAL para emular otro dispositivo. Esto reduce el número de dispositivos que se tienen que utilizar ya que solo se requiere del uso de una Gal.



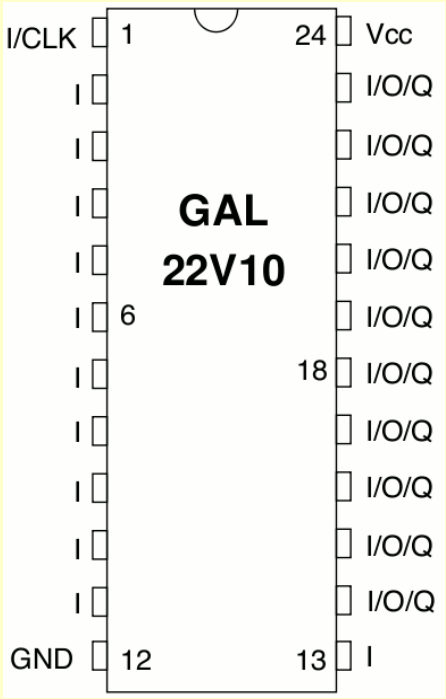


La GAL 22V10

Como se vio anteriormente este circuito es una GAL con las siguientes características:

- ✓ Retardo de propagación máximo de 4 nSeg.
- ✓ Frecuencia máxima 250 Mhz.
- ✓ Entradas con resistencia Pull-Up en todas las entradas
- ✓ Lógica reconfigurable
- ✓ Celdas reprogramables
- ✓ 8 macroceldas de salida
- ✓ Polaridad de salida programable
- ✓ "Firma" electrónica para identificación
- ✓ Voltaje de operación de 5 volts
- ✓ Corriente de salida en alto de hasta -3.2 mA
- ✓ Corriente de salida en bajo de hasta 16 mA.

Posee varias terminales de entrada, salida y entrada/salida, así como entrada de reloj(clock), a continuación se presenta el diagrama de terminales de la GAL 22V10



En este diagrama se puede apreciar que el circuito tienen definidas claramente cuales son terminales de entrada, cuales de entrada/salida y cuales tienen una función especial, así como en que terminales se encuentran las Macroceldas.

Estas características permiten realizar muchas de las principales funciones necesarias al utilizar un circuito lógico, el uso de las Macroceldas permite realizar programación utilizando diferentes esquemas. Así este dispositivo soporta 3 métodos de uso:

- ✓ Simple
- ✓ Compleja
- ✓ De registro

En esta práctica solo se utilizara el modo SIMPLE.

Software **WinCUPL**

Definiciones de PINES/NODOS

La definición de los PINES debe ser declarada al inicio del archivo del código fuente, esta definición es el punto de inicio natural para un diseño. Los nodos y los pines nodos son utilizados para definir registros, y también son declarados al inicio del código fuente. La asignación de los pines necesita ser realizada cuando el diseñador conoce el dispositivo que va a utilizar, sin embargo cuando se esta creando un diseño VIRTUAL solo el nombre de las variables son utilizado y posteriormente será asignado a los pines necesarios para ser llenados.

El área que normalmente contiene los números de los pines se dejara en blanco

Definición de las variables intermedias

Las variables intermedias son variables que están asignadas a una ecuación, pero no están asignadas a un PIN o a un NODO. Estas son utilizadas para definir una ecuación, la cual es usada por muchas variables o para proporcionar un diseño fácil de comprender.

Uso de variables indexadas

Los nombres de las variables que finalizan en un número decimal de 0 a 31 son nombrados como variables indexadas. Estas pueden ser utilizadas para representar un grupo de líneas de dirección, líneas de datos u otras numeradas de manera secuencial. Cunado las variables indexadas son utilizadas en operaciones de campos de bit, las variables con número de índice "0" es siempre el bit menos significativo.

Uso de las diferentes bases

Todas las operaciones realizadas en el compilador CUPL son realizadas con una precisión de 32 bits. Por esto, los números pueden almacenar un valor de 0 a 2^{32} -1 un número puede ser representado en cualquiera de las 4 bases más comunes.

- ✓ Binario
- ✓ Octal
- ✓ Decimal
- ✓ Hexadecimal

La base predeterminada para TODOS los números en el archivo fuente es hexadecimal excepto para las terminales del dispositivo y las variables indexadas, las cuales siempre son decimales. Los números Binarios, Octales y Hexadecimales pueden tener valor sin cuidado "X" mezclados con valores numéricos.

Número	Base	Valor decimal
'b'0	Binario	0
'B'1101	Binario	13
'O'663	Octal	435
'D'92	Decimal	92
'h'BA	Hexadecimal	186
'O'[300..477]	Octal (un rango)	192..314
'H'7FXX	Hexadecimal (un rango)	32512..32767

Uso de la Notación de lista

Una lista es un método de ordenamiento de la definición de grupos de variables. Esto es comúnmente utilizado en:

- ✓ Declaraciones de pines y nodos
- ✓ Declaraciones de campos de bits
- ✓ Ecuaciones lógicas
- ✓ Operaciones de asignación

Los paréntesis son utilizados para delimitar los elementos en la lista

Ejemplo:

La siguiente instrucción: [A7,A6,A5,A4,A3,A2,A1,A0];

Se puede representar por: [A7..A0];

Uso de campos de Bit

Una declaración de campo de bit asigna un nombre simple a un grupo de bits. Después de hacer una asignación de campo de bits utilizando la sentencia FIELD, el nombre puede ser utilizado en una expresión, la operación especificada es aplicada a cada bit en el grupo. Cuando una sentencia FIELD es utilizada, el compilador genera un campo interno sencillo de 32 bits. Esto es usado para representar las variables en el campo bit. Cada bit representa un miembro del campo. El número de bit el cual representa un miembro del campo de bits es el mismo que el número índice si se utilizan variables indexadas. Esto significa que A0, siempre ocupará el bit 0 en el campo. Esto es utilizado principalmente para definir y manipular buses de datos y direcciones.

Ejemplo:
FIELD variable = [A7,A6,A5,A4,A3,A2,A1,A0];

ó bien

FIELD variable = [A7..A0];

Uso de la sintaxis del lenguaje

Uso de los operadores lógicos.

Cuatro operadores lógicos estándar están disponibles para utilizarse NOT, AND, OR, y XOR. La siguiente tabla muestra los operadores y su precedencia de mayor a menor.

Operador	Ejemplo	Descripción	Precedencia
!	!A	NOT	1
&	A&B	AND	2
#	A#B	OR	3
$	A$B	XOR	4

Uso de operadores aritméticos y funciones

Seis operadores aritméticos estándar están disponibles para ser utilizados. La siguiente tabla muestra estás operaciones y su orden de precedencia de mayor a menor.

Operador	Ejemplo	Descripción	Precedencia
**	2**3	Exponencial	1
*	2*1	Multiplicación	2
/	4/2	División	2
%	9%8	Módulo	2
+	2+4	suma	3
-	4-1	Resta	3

Una función aritmética está disponible para ser utilizada en expresiones aritméticas, la siguiente tabla muestra la función aritmética y su base.

Función	Base
log2	Binario
log8	Octal
log16	Hexadecimal
log	Decimal

Uso de las tablas de verdad

Al uso de tablas se le considera como un uso AVANZADO del lenguaje de programación. El uso de tablas de información es algunas veces el camino mas claro para expresar una descripción lógica. Este compilador proporciona la directiva TABLE la cual es capaz de crear tablas de información. La sintaxis de esta directiva indica que primero se deben de proporcionar la lista de las variables de entrada y de salida, y posteriormente especificar una a una la asignación entre los valores de entrada y los valores de las variables de salida. Valores sin cuidado (don't care) son permitidos para los valores de las entradas pero NO está permitido para los valores de salida. Una lista de los valores de entrada puede ser especificada para realizar múltiples asignaciones en una sola directiva.

Ejemplo:

```
FIELD  input = [in3..0] ;
FIELD  output = [out4..0] ;

TABLE  input => output {
        0=>00;      1=>01;      2=>02;      3=>03;
        4=>04;      5=>05;      6=>06;      7=>07;
        8=>08;      9=>09;      A=>10;      B=>11;
        C=>12;      D=>13;      E=>14;      F=>15;
}
```

Uso del Compilador CUPL

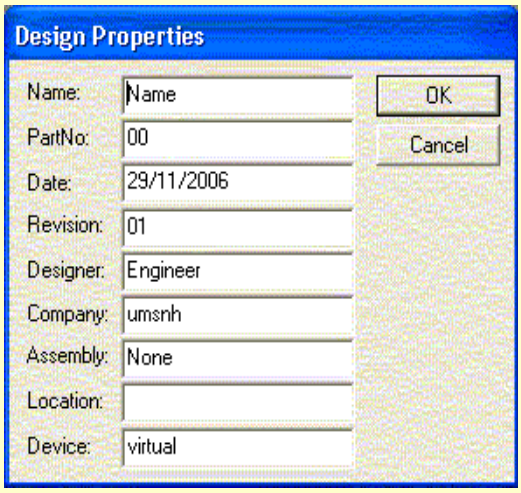
El compilador CUPL es un programa que toma un archivo de texto que esta constituido por directivas de alto nivel y comandos, en base a esto, construye archivos con información más primitiva la cual es reconocida por el programador que se utilizará, o bien por un simulador.

Antes de ejecutar el software es necesario instalar el compilador.

Al ejecutar el software se obtiene un ambiente de desarrollo integrado (IDE), el cual integra a todos los elementos necesarios para el desarrollo del código compatible con del programado, en nuestro caso con extensión .JED, que es el que interpreta ya sea el programador o bien el simulador.

Al iniciar se debe de ir a File → New → Proyect

En la ventana se puede ver algunas de las opciones del proyecto, algunas de estas opciones se deben de modificar y otras se pueden omitir..



Name: Aquí se coloca el nombre del proyecto, no debe de contener mas de 8 caracteres y sin extensión

Device: Aquí se coloca al dispositivo que se va a utilizar, de manera predeterminada se encuentra seleccionada la opción virtual, la cual permite tener un número indeterminado de entradas y salidas, sin embargo nosotros utilizaremos el dispositivo 22v10a.

Al menos se deben de llenar estas 2 opciones.

Posteriormente a esto se piden el número de entradas y el número de salidas, así como el número de pines/nodos. En este momento solo se utilizarán entradas y salidas.

A continuación se genera un archivo predefinido, con el encabezado así como el número de entradas y salidas seleccionadas.

Uno de los primeros pasos antes de desarrollar cualquier software, es asignar las entradas y las salidas de acuerdo al dispositivo utilizado. La determinación de las entradas y salidas depende de la GAL que se utilizará ya que no se pueden realizar asignaciones arbitrarias.

Posteriormente de la asignación de los pines se procede a realizar el software de aplicación.

En el menú opciones → compilación se seleccionan las siguientes:

Output File

Download

- ✓ JEDEC

Doc File Options

- ✓ Fuse Plot
- ✓ Equations

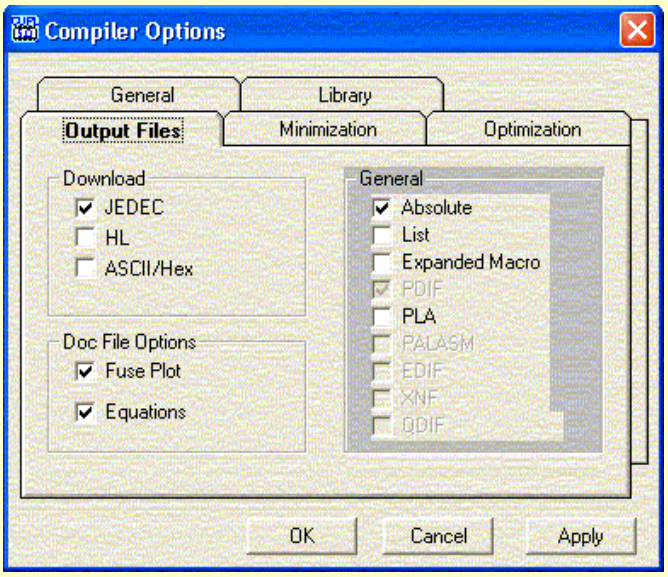
General

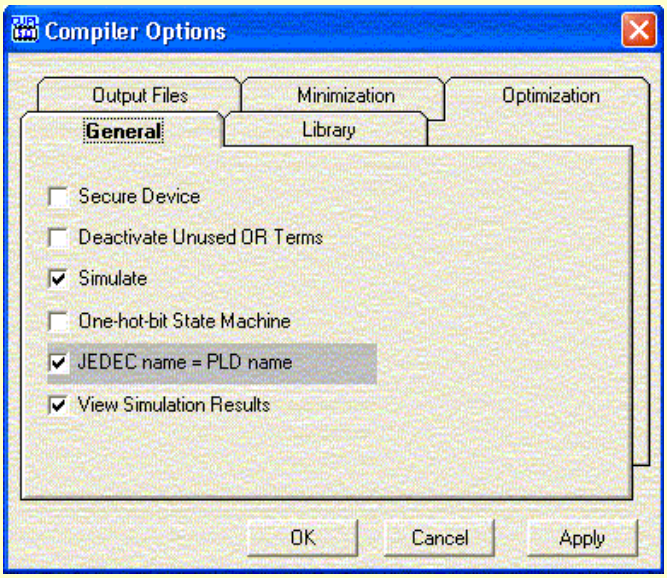
- ✓ Absolute

General

- ✓ Simulate
- ✓ JEDEC name = pld name
- ✓ view simulationresults

al finalizar OK





En el menú options → Device Selection debemos de seleccionar

Package type

- ✓ Dip

Manufacturer

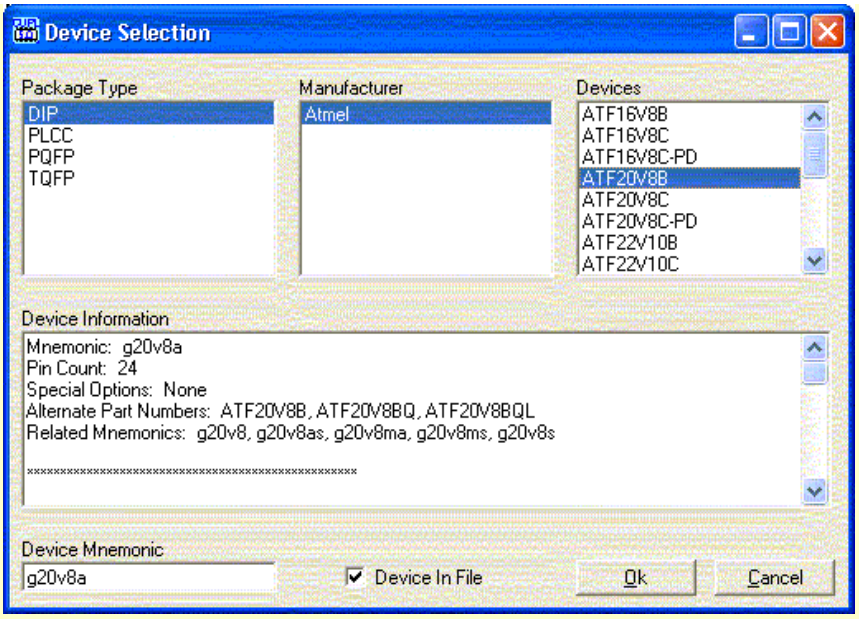
- ✓ Atmel

Devices

- ✓ ATF22V10B o el circuito con el que contamos.

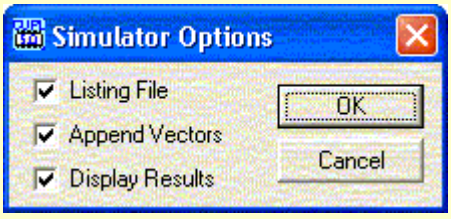
Device Information

En esta área se puede apreciar que los mnemonicos compatibles con este dispositivo tienen a un dispositivo común el **g22v10.** Este será el dispositivo que se utilizará para el desarrollo de las aplicaciones.



Finalmente en el menú options → Simulator

- ✓ Listing File
- ✓ Append Vectors
- ✓ Display Results



Al compilar, se genera un proyecto, el cual debe de constar con los siguientes archivos y las siguientes extensiones:

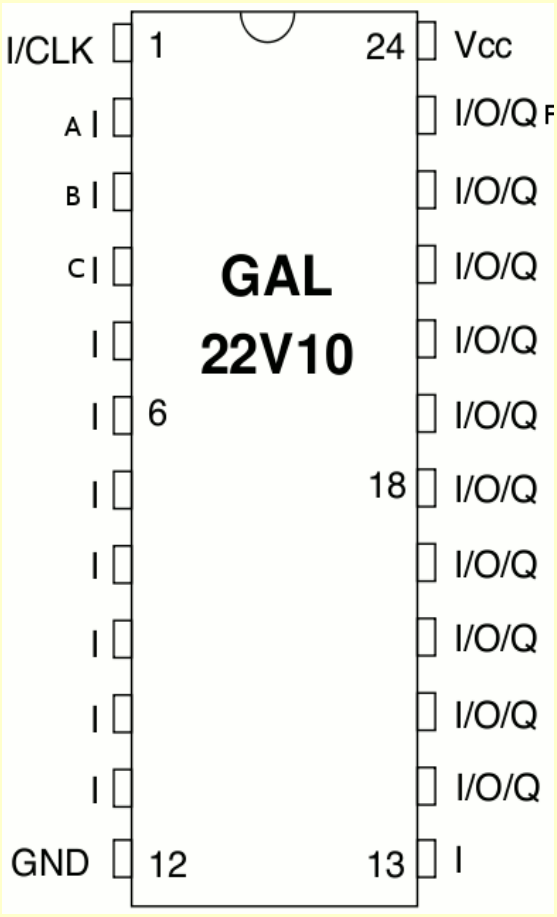
.abs
.doc
.jed
.pdf

El archivo de extensión .JED es el que puede utilizar el programador.
El archivo fuente es el archivo de extensión PLD

Ejemplo:
A continuación se da un ejemplo en el cual se desea resolver la siguiente función:

$$F = AB + A\overline{C}$$

Para realizar esta función es necesario definir primero el número de entradas y salidas, como se puede apreciar en la función existen 3 entradas (A, B y C) y una sola salida (F). Adicionalmente se requiere de definir el tipo de GAL así como su empaquetado que se utilizará, en esta práctica se utilizará la GAL20V8, de la cual se asignaran los siguientes pines:



En este diagrama se puede apreciar que hemos considerado que las entradas A, B y C se encuentran en las terminales 2, 3 y 4, y la salida F en la Terminal 23. Para implementar esta función, se introduce el siguiente código después de definir el número de entradas y su ubicación en el encapsulado.

```
PIN 2 = A;
PIN 3 = B;
PIN 4 = C;

PIN 23 = F;

F = (A & B) # (A & !C);
```

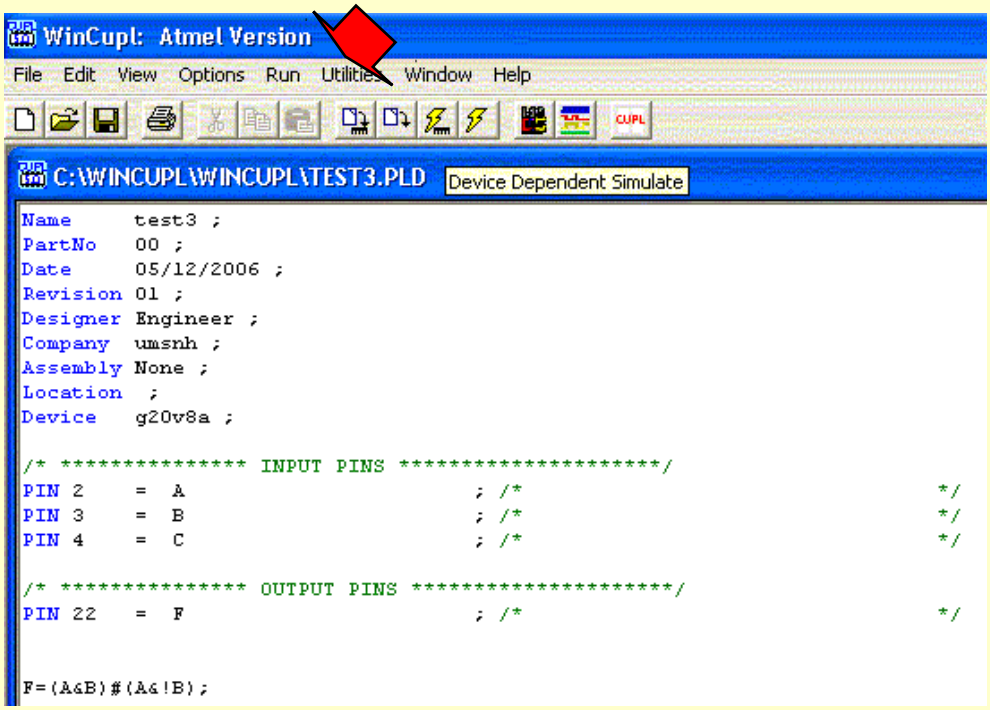
Lo primero que podemos observar, es que todas las instrucciones terminan con ";" permitiendo que finalice la instrucción.

Las tres primeras líneas inician con la directiva "PIN", esta directiva le indica que existe una Terminal asociada a cada uno de las entradas, así las terminales utilizadas son: la 2, la 3 y la 4, también se esta asignando la salida a otra Terminal, la 23, se debe tener cuidado al seleccionar cada una de las terminales, ya que la definición de entrada o salida depende solo de la disposición del encapsulado y no del software implementado. A partir de este momento se procede a compilar el código fuente de acuerdo a las instrucciones antes mencionadas. A partir del cual se genera el código .JED que posteriormente se programará en el CHIP.

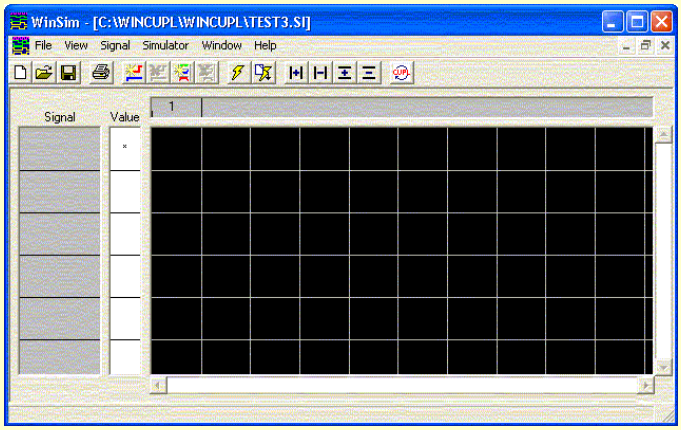
El simulador:

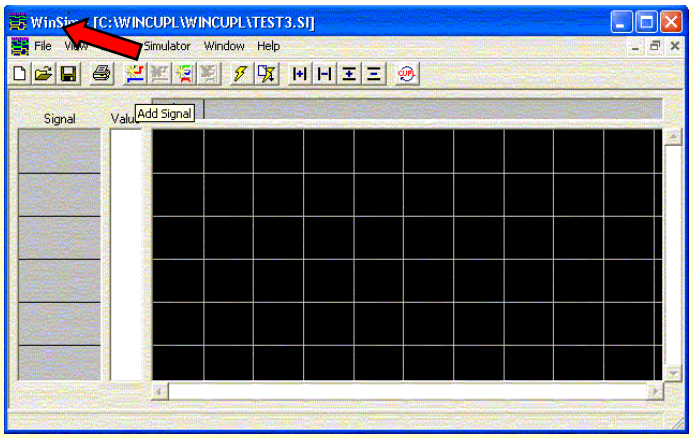
Como en muchas de las aplicaciones, antes colocar el código objeto en el CHIP, es adecuado realizar una simulación del comportamiento, esto permite observar la operación del software antes de vaciarlo al CHIP. Este método permite observar en "baja velocidad" la ejecución del software, que en ocasiones no es posible en la aplicación final.

Para utilizar el simulador solo se requiere de seleccionar el modo simulador, posterior a la compilación del código fuente.

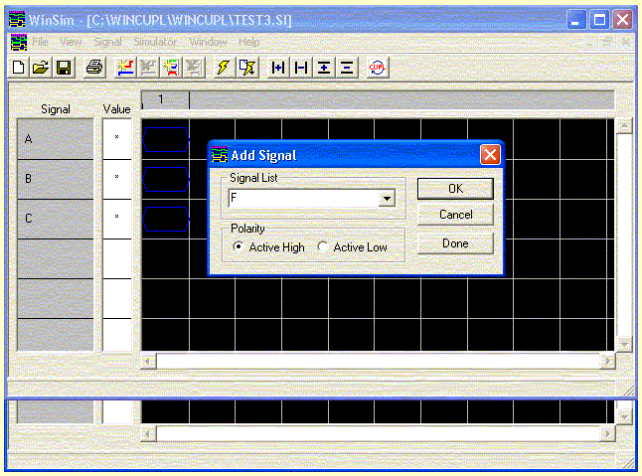


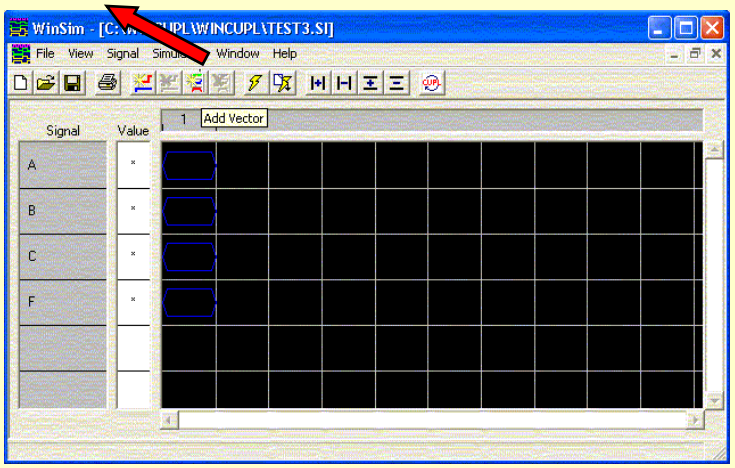
Al presionar este botón se observa la siguiente imagen, en la cual se ve una columna y una pantalla donde se presentará la simulación, en la columna Signal se presentan las señales de entrada o de salida necesarias, en la columna Value se presenta el valor tanto de la entrada como la simulación y en la pantalla negra el estado lógico de las señales. En la siguiente imagen se presenta como se agregan cada una de las señales al presionar el botón ADD signal



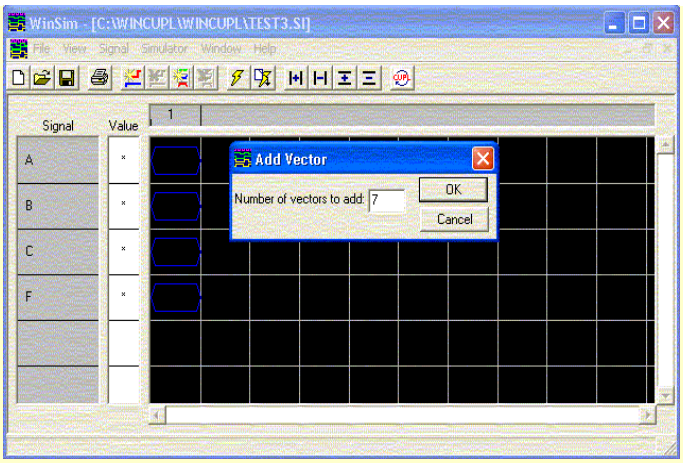


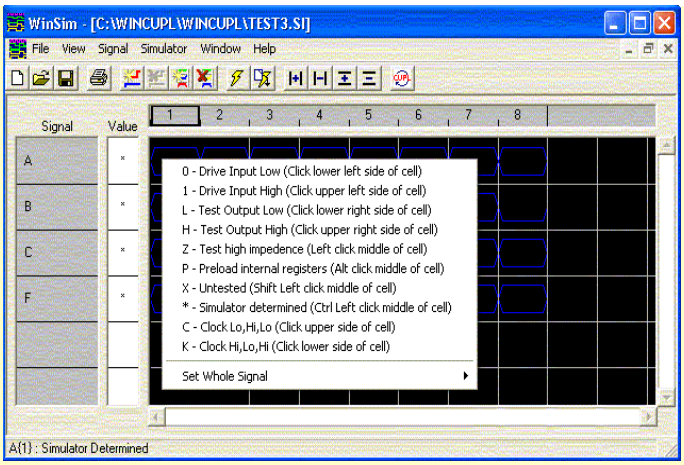
Al ir agregando cada una de las líneas se presenta un cuadro de dialogo como el siguiente, posteriormente de agregar las señales queda así:



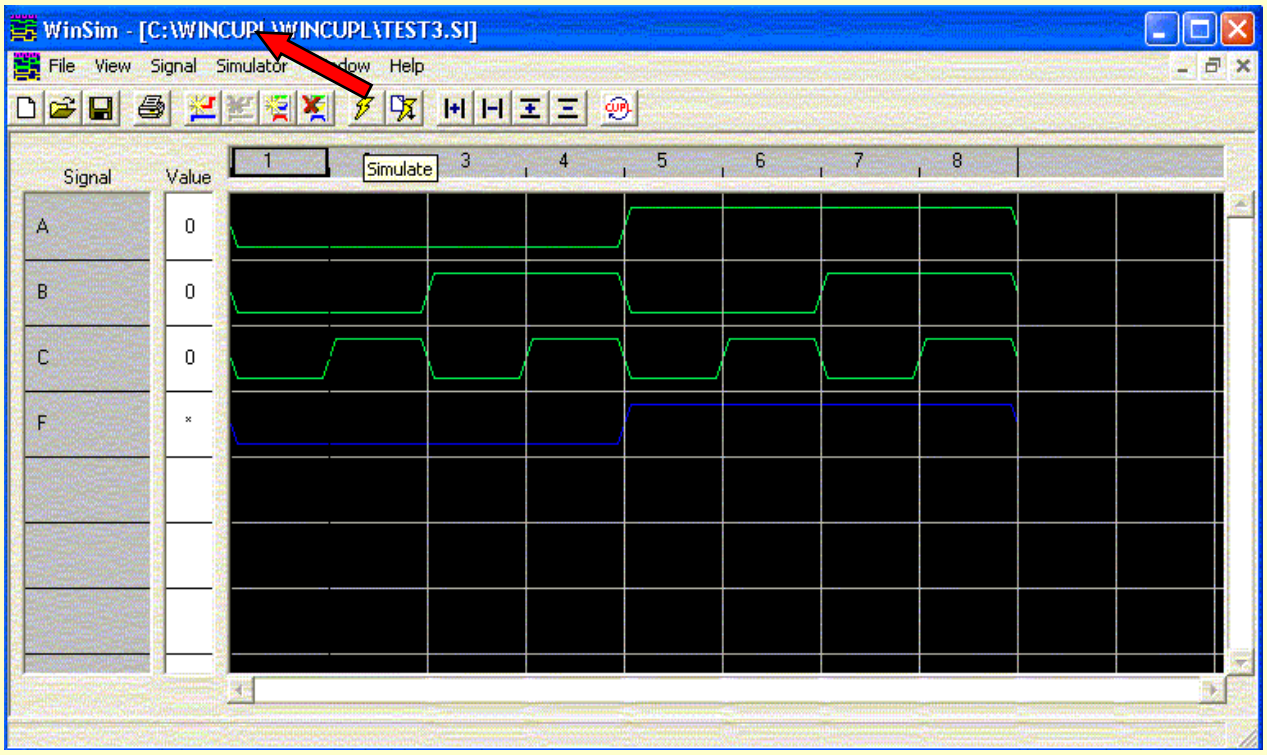


Ahora se agregan los diferentes "periodos" necesarios para poder realizar la simulación de manera adecuada, en este caso tenemos 3 entradas con lo cual tenemos hasta 8 diferentes estados, por lo que se agregan 7 vectores o "periodos". Posteriormente de agregar los vectores, se deben de introducir los diferentes estados "considerados" de las entradas esto se hace colocando el apuntador sobre la señal de simulación. Existen diferentes tipos de señales como son entradas, salidas, tercer estado etc. en nuestro caso A, B y C son entradas y F es salida o bien dependiente del simulador. Todas las posibles opciones se muestran en el recuadro inferior de la imagen





Al activar cada una de las señales, ahora es posible realizar la simulación del circuito, las líneas verdes indican las entradas y las líneas azules la salida la respuesta del código al simulador.



El programador:

Para utilizar el programador se requiere del software GALBLAST, el cual recibe el código del archivo .JED y a partir de este realiza la programación adecuada del dispositivo. O el programador comercial del laboratorio.

Requisitos:

Hoja de datos de la GAL22V10

Leer La práctica

Desarrollo:

Implementar el código desarrollado utilizando la GAL22V10