Práctica 5

Dispositivos Lógicos Programables

Objetivo
Diseñar circuitos combinacionales y secuenciales en Dispositivos Lógicos Programables.

Equipo
Computadora personal con el software WinCUPL y WinSIM u otro software para desarrollo de código para PLDs.

Fundamento teórico

Dispositivos lógicos programables

Los dispositivos lógicos programables (Programmable Logic Device, PLD) permiten al desarrollador especificar la lógica de operación del dispositivo por medio de programación. De forma que los pasos de simplificación de relaciones algebraicas de las entradas y salidas del sistema y selección de los circuitos integrados para la implementación del circuito puede ser automatizadas por un software de desarrollo para PLDs. Usar lógica programable mejora la eficiencia del diseño y proceso de desarrollo. De esta manera, el trabajo del diseñador es identificar las entradas y salidas, especificar su relación lógica y seleccionar un dispositivo programable donde se pueda implementar el circuito al menor costo. El concepto detrás de los dispositivos programables es simple: están compuestos de muchas compuertas lógicas en un solo circuito integrado y controlan electrónicamente las interconexiones de estas compuertas. En la Fig. 1 se ilustra este concepto con un circuito combinacional sencillo.

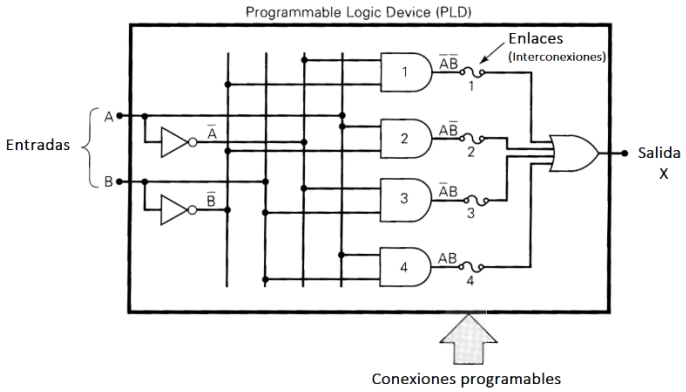


Figura 1. Ejemplo simplificado de un PLD.

La lógica en este PLD simplificado está en la forma de suma de productos (Sum of Products, **SOP**) con compuertas AND alimentando una OR final. La salida X es la función SOP de las entradas A y B. La función de salida depende de cuáles salidas de la AND están conectadas a las entradas de la OR. En el ejemplo, todas las salidas de la AND están conectadas a la OR por medio de conectar los enlaces 1 a 4. Al programar el dispositivo, todos los enlaces pueden ser dejados conectados como en este ejemplo, o podrían desconectarse algunos, de forma que algunas salidas de la AND estén desconectadas de la OR. Por ejemplo, si los enlaces 1 a 3 son abiertos, solo la compuerta 4 de la AND va a estar conectada y salida sería $\mathrm{X} = \mathrm{AB}$; si los enlaces 1 y 4 están abiertos, la salida es $\mathrm{X} = \mathrm{A}\overline{\mathrm{B}} + \overline{\mathrm{A}}\mathrm{B}$. La circuitería de un PLD es tal que un enlace abierto produce un Bajo en su entrada en la OR.

El chip PLD inicialmente tiene todos los enlaces intactos. Para acceder a estos enlaces, el chip debe ser puesto en un modo especial de operación y voltajes especiales deben ser aplicados a ciertos pines. Esto es llamado p*rogramar* el PLD. El chip es colocado en una fixtura llamada programador conectado a la computadora, un software de desarrollo traduce un archivo con el diseño a otro archivo llamado "fuse plot". Este es una especie de mapa que indica qué enlaces en el dispositivo van a ser desconectados y cuáles van a ser dejados intactos. El formato para transmitir los datos de programación al PLD sigue el estándar 3 **JEDEC** (Joint Electronic Device Engineering Council), donde se estandariza también la asignación de pines para varios empaquetados de circuitos integrados independientemente del fabricante, haciendo los programadores universales menos complicados. En la Fig. 2 se muestran los componentes para diseño de circuitos con PLDs. En la Fig. 3 se muestran los métodos en que se puede especificar un circuito en un compilador para desarrollo con PLDs.

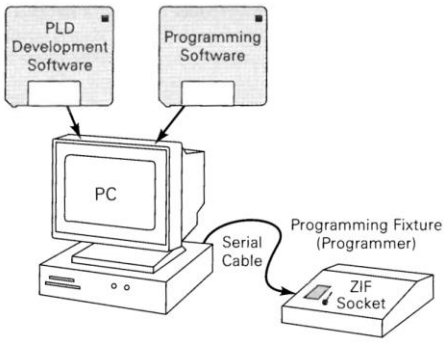


Figura 2. Sistema para diseño de circuitos con PLDs.

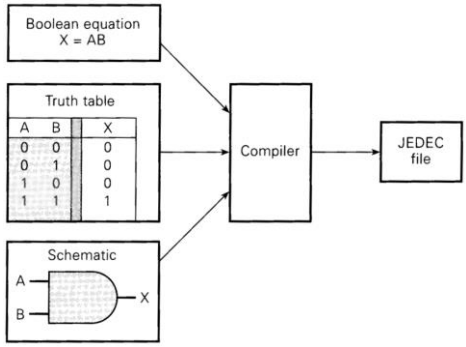


Figura 3. Métodos para describir un circuito en un compilador para PLDs.

La sintaxis para describir la operación de un circuito es llamada Lenguaje de Descripción de Hardware (Hardware Description Language, **HDL**). Un compilador común para diseño con PLDs es **CUPL**. En este compilador, el archivo de entrada es dividido en varias secciones. La *Cabecera* documenta detalles e información que el compilador puede usar para programar el dispositivo. La sección de *Especificación de Entradas y Salidas* es usada para asignar nombres a las señales para los pines del PLD. La sección de *Descripción de Hardware* permite indicar el diseño del circuito en uno de los modos ilustrados en la Fig. 3. A continuación, se muestra un ejemplo de un circuito combinacional descrito en lenguaje CUPL. En la Tabla 1 se presenta la sintaxis para las operaciones lógicas, en el Listado 1 está el código que describe la expresión SOP $X = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$ a implementarse en un PLD GAL16V8 y en la Fig. 4 el diagrama del dispositivo con las entradas y salidas.

Función	Operador	Formato CUPL
AND	&	A & B
OR	#	A # B
NOT	!	!A
XOR	$	A $ B

Tabla 1. Sintaxis de CUPL para operaciones lógicas.

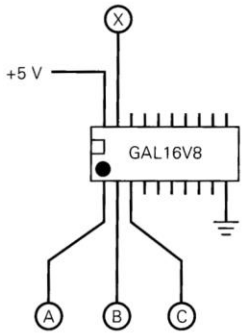
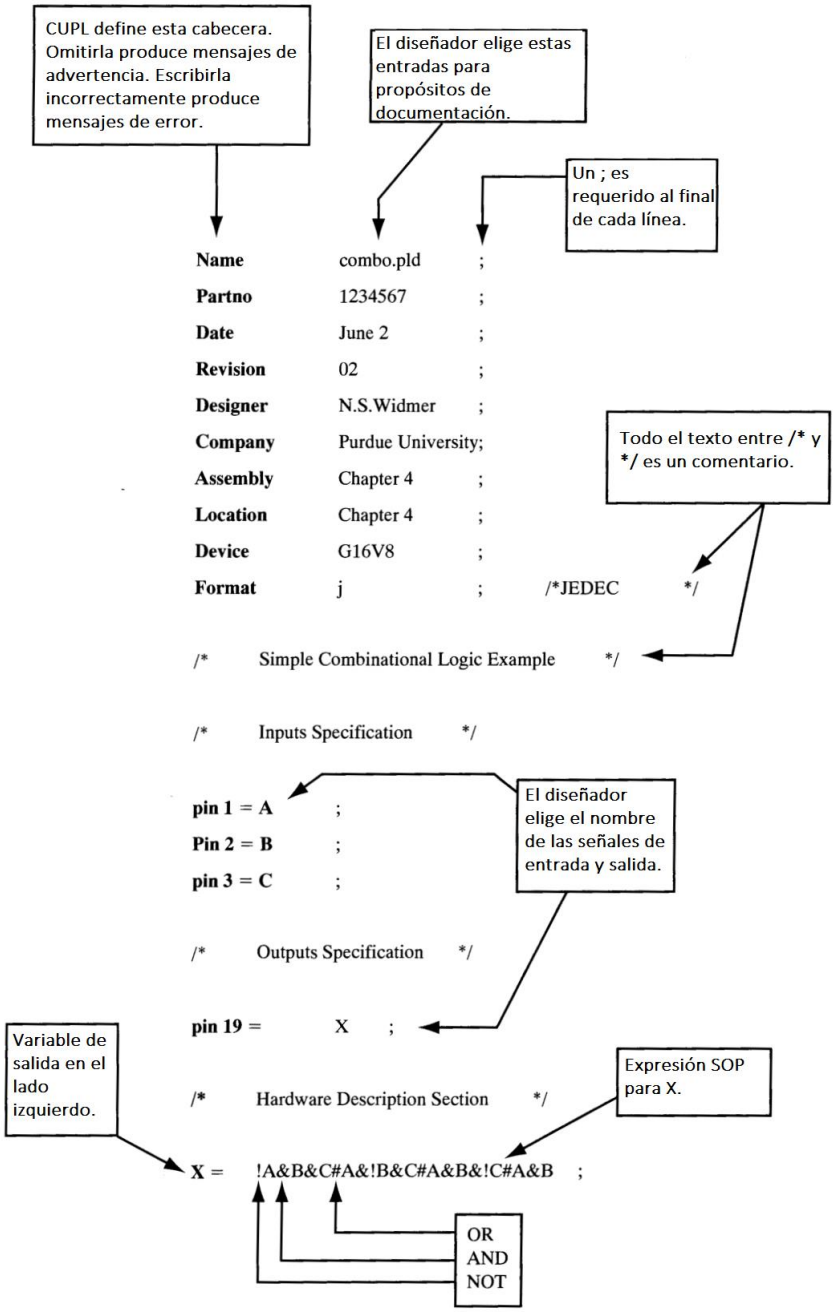


Figura 4. Diagrama de conexión del dispositivo GAL16V8.



Listado 1. Descripción en CUPL de un circuito combinacional.

El compilador reduce la ecuación a su forma más simple antes de generar el archivo de salida. En este punto, el diseño puede ser probado en un simulador, el cual es un programa que calcula los estados de salida basándose en la descripción del circuito y las entradas actuales. Los *vectores de prueba* son un conjunto de entradas hipotéticas y sus correspondientes salidas, las cuales prueban que el diseño opera correctamente. Si los vectores de prueba son comprensivos, el diseño puede ser probado incluso antes de programar el dispositivo.

GAL16V8 y GAL22V10

Los dispositivos Arreglo Lógico Genérico (Generic Logic Array, **GAL**) usan un arreglo de EEPROMs donde ciertas localidades de memoria son designadas para controlar conexiones entre renglones y columnas de una matriz de términos de entrada. El uso de EEPROMs permite borrar y reprogramar el dispositivo. No es necesario que el desarrollador determine los datos en la EEPROM para las conexiones, el software de programación se encarga de ello.

El diagrama lógico del **GAL16V8** es mostrado en la Fig. 5. Este dispositivo tiene ocho pines de entrada dedicados, dos pines de selección de función y ocho pines que pueden ser usados ya sea como entrada o salida. Los componentes principales de un GAL son la matriz de términos de entrada, compuertas AND que generan los productos de los términos de entrada y las macro celdas de lógica de salida (**OLMC** en la figura). La flexibilidad del GAL16V8 recae en sus OLMCs programables. Ocho diferentes productos (salidas de las compuertas AND) son aplicados como entradas a cada una de las ocho OLMC. Dentro de ellas, los productos son ingresados a una OR para generar la suma de productos (SOP). También dentro de la OLMC, la salida SOP puede ser ruteada a un pin de salida para implementar un circuito combinacional, o puede ser ingresada a un flip flop D para implementar un circuito con salida registrada (sincronizada). En las OLMC es donde se encuentran los bits programables de la EEPROM.

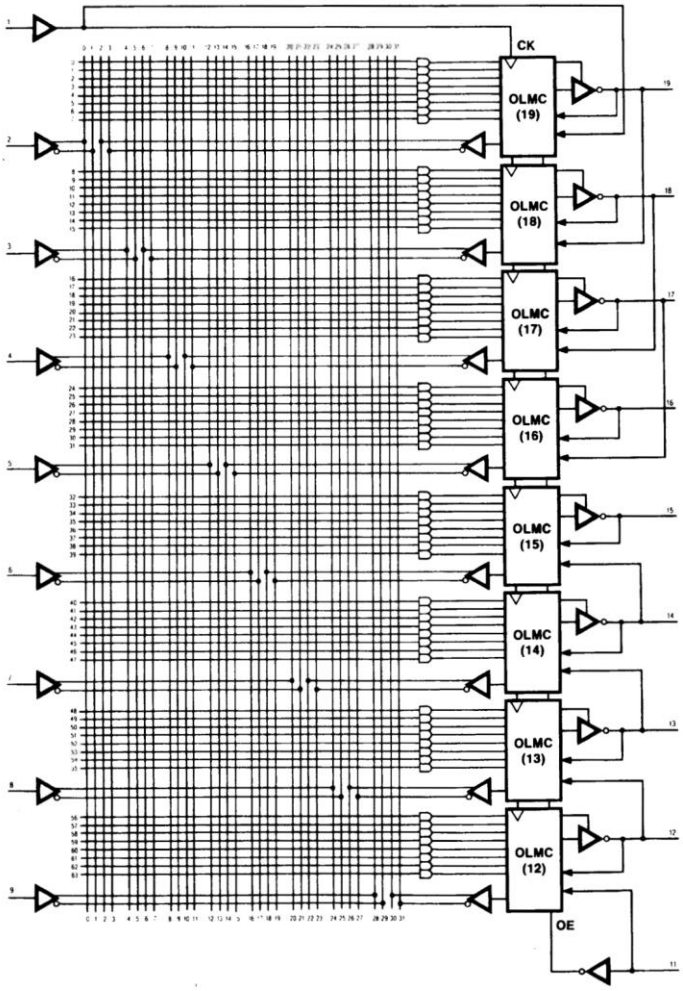


Figura 5. Diagrama Lógico de un GAL16V8.

Un GAL popular es el **GAL22V10**, el cual tiene doce entradas dedicadas y diez entradas o salidas. Su diagrama lógico se muestra en la Fig. 6, el cual es una arquitectura similar, mas no idéntica, al GAL16V8. A diferencia del 16V8, cada OR en el 22V10 no combina el mismo número de términos de producto. El número de términos va desde ocho a 16. Para tomar ventaja de los términos extra, se asignan las expresiones más grandes a los pines de salida correctos.

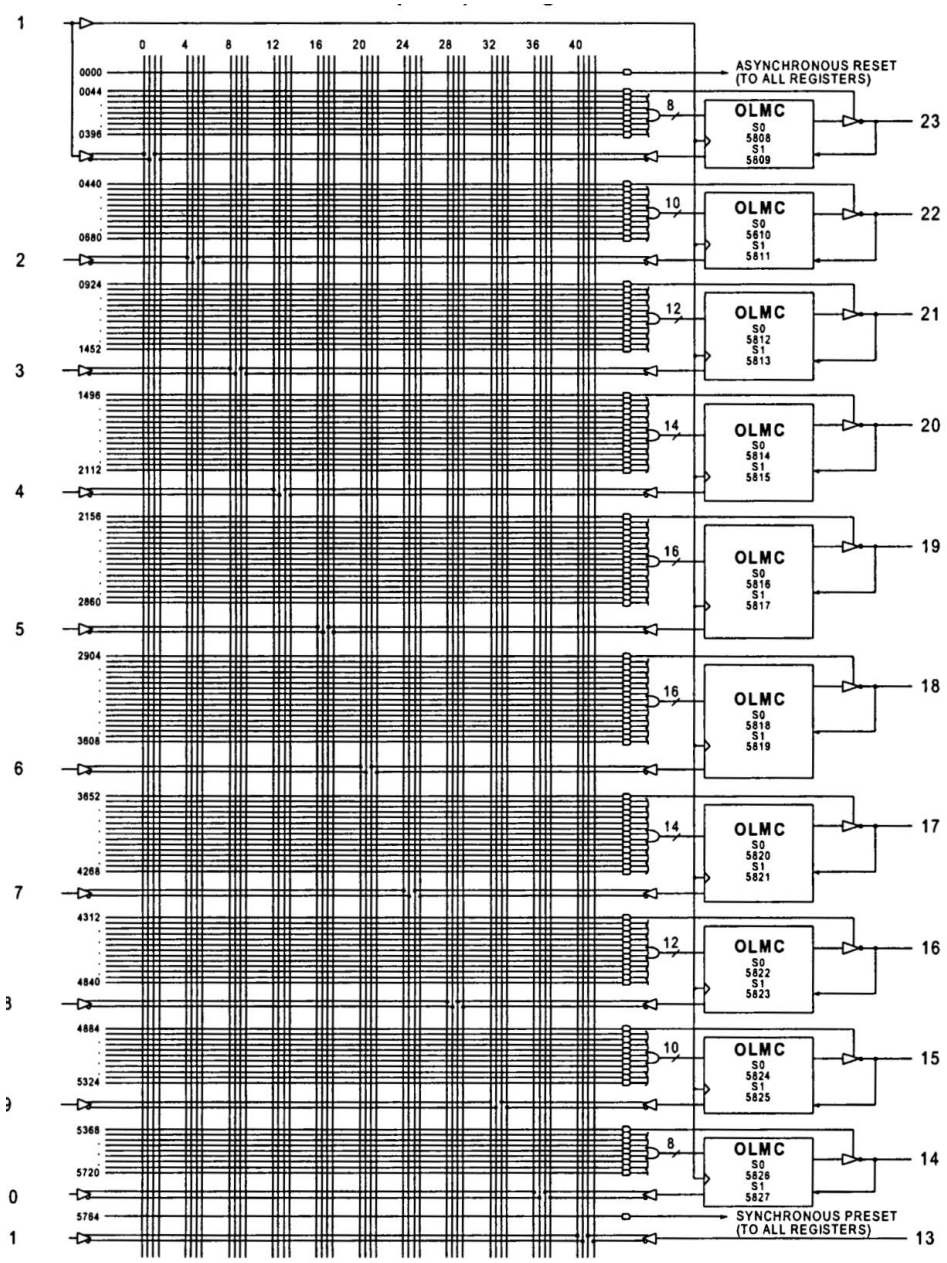


Figura 6. Diagrama Lógico de un GAL22V10.

Desarrollo

0. Descargue e instale el software de diseño para PLDs **WinCUPL** del sitio del fabricante Microchip. El software fue desarrollado por Atmel, esta compañía fue posteriormente adquirida por Microchip.

   https://www.microchip.com/design-centers/fpgas-and-plds/splds-cplds/pld-design-resources

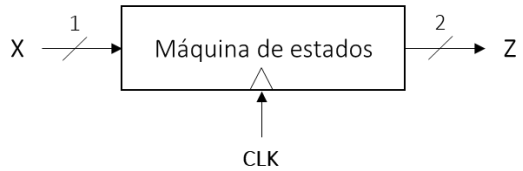
   En el sitio se indica que es necesario usar el número de serie 60008009.

   Es válido usar otro software para desarrollar código para dispositivos GAL.

1. **Circuito combinacional**
   a) Copie y compile el código del *Listado 1* que describe un circuito combinacional.
   b) Simule en **WinSim** (se instala junto con WinCUPL) el circuito combinacional para distintos valores de entrada.
   c) Modifique el código del *Listado 1* para describir la siguiente ecuación en el dispositivo GAL22V10:

$$F = A\overline{B} + ABD + AB\overline{D} + \overline{A}\overline{C}\overline{D} + \overline{A}B\overline{C}$$

   d) Compile y simule el código desarrollado.
   e) Realice un video donde describa la simulación. Indique los distintos valores de entrada que está aplicando al circuito y explique las salidas obtenidas.
2. **Circuito secuencial**
   a) Diseñe un circuito detector de secuencia con una entrada **X** y dos salidas, $\mathbf{Z_1}$ y $\mathbf{Z_2}$, que detecte la aparición de las secuencias 1011 y 1101 en la entrada. La salida $\mathbf{Z_1}$ es 1 cada vez que se recibe la secuencia 1011, mientras que $\mathbf{Z_2}$ es 1 cada vez que 1101 es recibida. El detector debe ser con traslape. El dispositivo a usar es un GAL22V10.



   Básese en el código del *Listado 2* para describir el circuito.

   Procedimiento:
   1. Realice el diagrama de estados y su tabla de transición de estados.
   2. Si aplica, reduzca la cantidad de estados al eliminar estados redundantes.
   3. Describa en CUPL su máquina de estados:
      - Escriba la información de la cabecera del archivo.

- Establezca los pines de entrada y salida.
- Defina variables intermedias y ecuaciones lógicas que sean necesarias.
- Describa la máquina secuencial de estados.

4. Simule el código en WinSim.

b) Realice un video donde describa **detalladamente** su detector de secuencia en WinCUPL, muestre la simulación del circuito y describa el valor de entrada que le esté activando, los estados que va recorriendo el circuito y la salida obtenida. Haga esto para las secuencias 1011 y 1101.

En su reporte incluya:

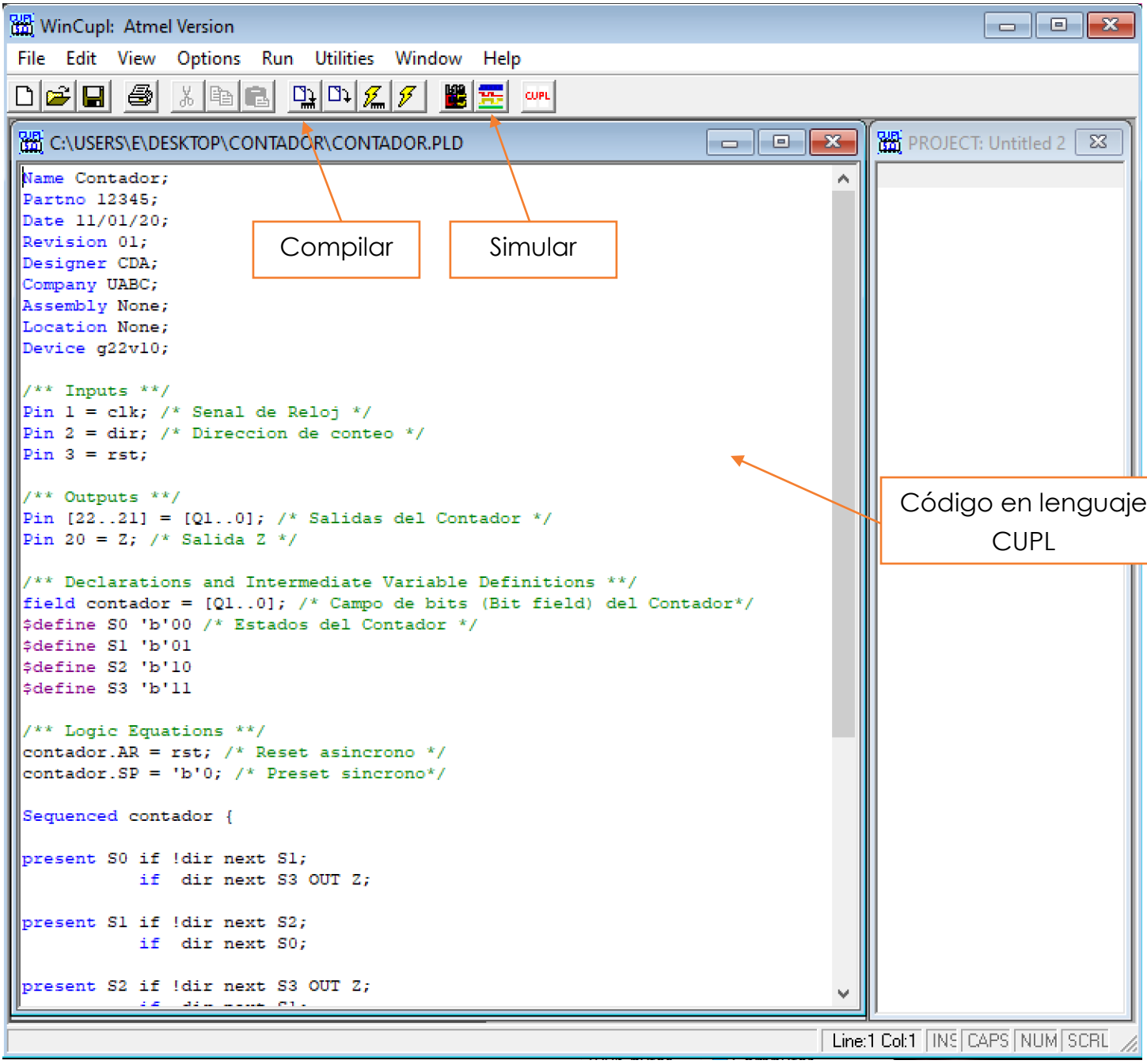
- Diagrama de estados.
- Tabla de transición.

Al entregar su práctica, adjunte los archivos de código de WinCUPL.

Conclusiones y comentarios
Dificultades en el desarrollo
Referencias

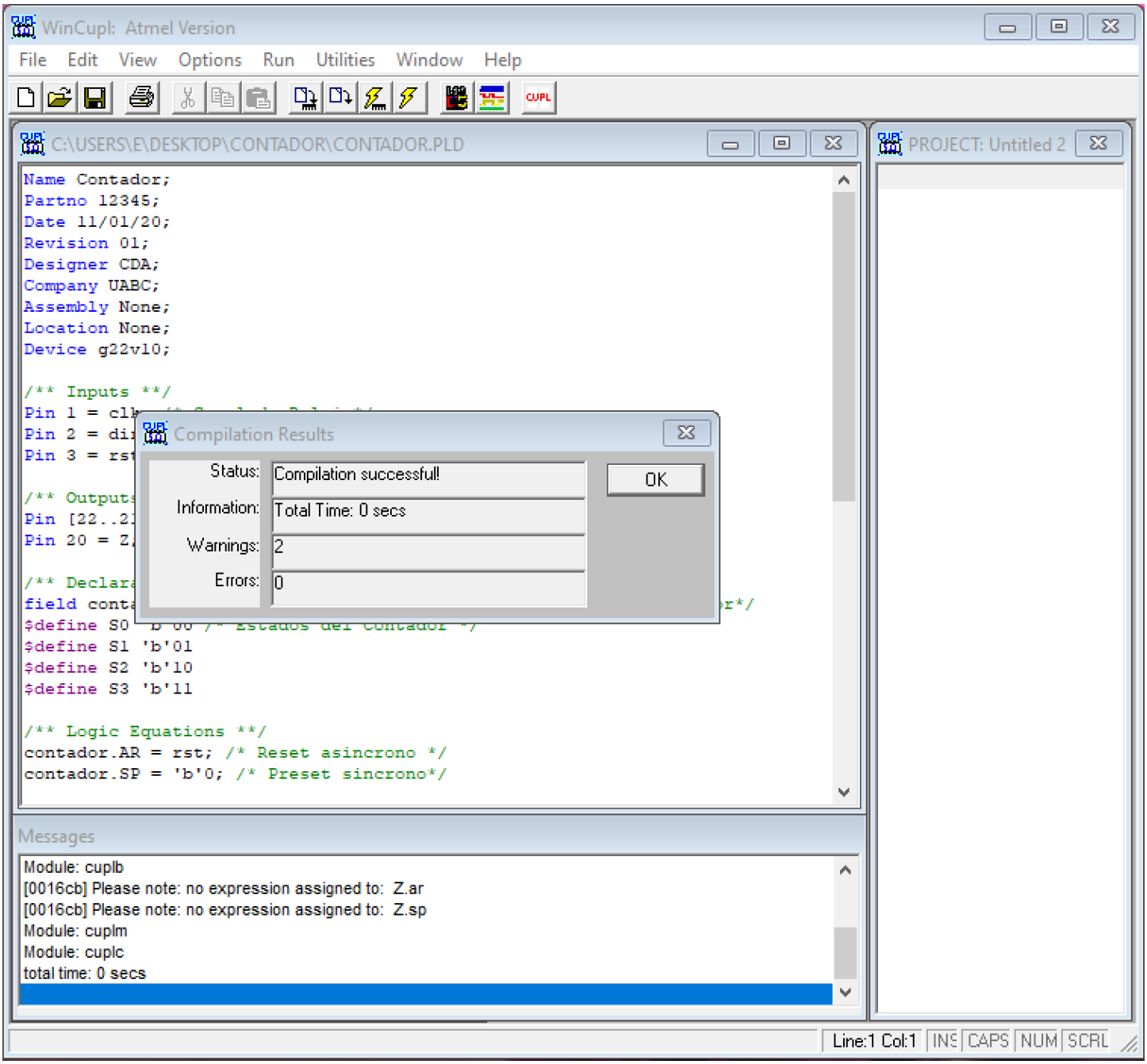
WinCUPL

- **Ventana principal**




```
Name Contador;
Partno 12345;
Date 11/01/20;
Revision 01;
Designer CDA;
Company UABC;
Assembly None;
Location None;
Device g22v10;

/** Inputs **/
Pin 1 = clk; /* Senal de Reloj */
Pin 2 = dir; /* Direccion de conteo */
Pin 3 = rst;

/** Outputs **/
Pin [22..21] = [Q1..0]; /* Salidas del Contador */
Pin 20 = Z; /* Salida Z */

/** Declarations and Intermediate Variable Definitions **/
field contador = [Q1..0]; /* Campo de bits (Bit field) del Contador*/
$define S0 'b'00 /* Estados del Contador */
$define S1 'b'01
$define S2 'b'10
$define S3 'b'11

/** Logic Equations **/
contador.AR = rst; /* Reset asincrono */
contador.SP = 'b'0; /* Preset sincrono*/

Sequenced contador {

present S0 if !dir next S1;
           if  dir next S3 OUT Z;

present S1 if !dir next S2;
           if  dir next S0;

present S2 if !dir next S3 OUT Z;
```

- **Mensaje de compilación exitosa**

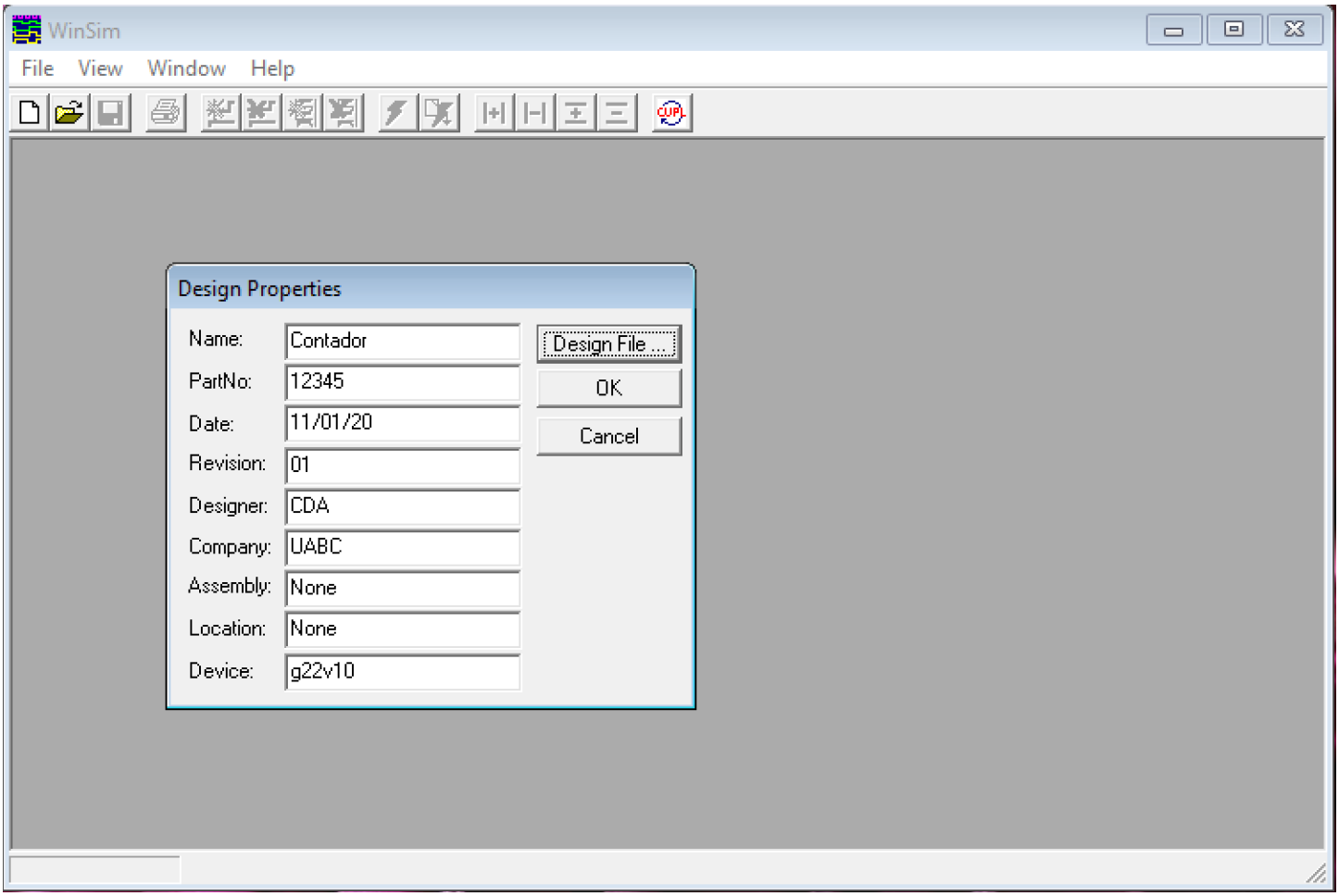


Si WinCUPL indica errores al compilar aun cuando el código es correcto, es necesario volver a especificar el dispositivo:
Clic en **Options**, después en **Devices**, seleccionar **DIP** - **Atmel** y **ATF22V10C**.
La compilación es: clic en **Run** y después en **Device Dependent Compile**.

WinSIM

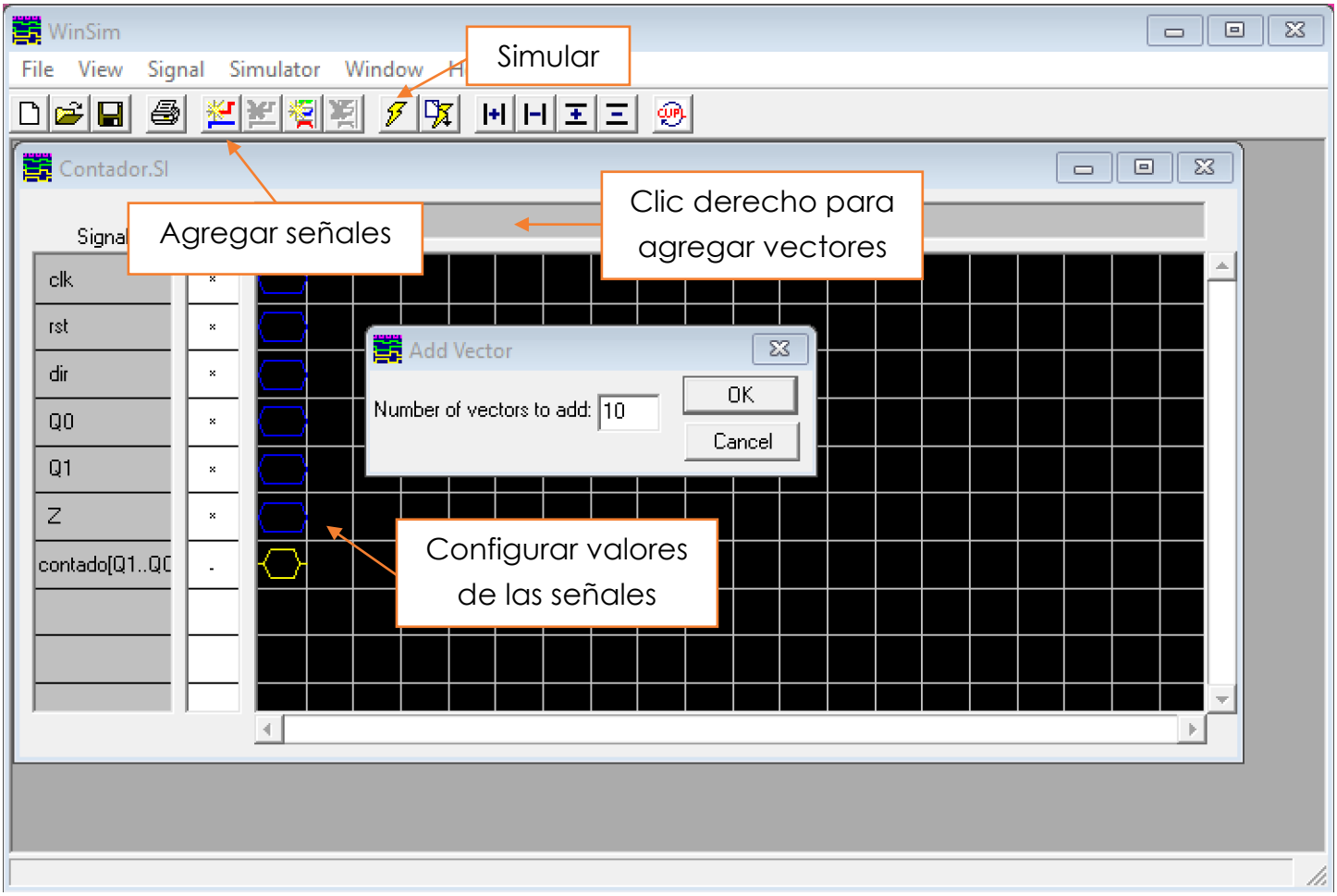
- **Ventana principal**

Clic en **New**, clic en **Design File**, seleccionar **Contador.pld**.



- **Simular**

1. Agregar señales.
2. Agregar vectores.
3. Configurar los valores de las señales.



Ejemplo de un contador descrito en CUPL para un GAL22V10

Diseñe un contador módulo 4 para un GAL22V10.

El circuito tiene tres entradas y tres salidas. Las entradas son la señal de **reloj**, **reinicio** y **dirección**. Cuando se activa el reinicio, el contador pasa a 0. La señal de dirección indica el modo de conteo, cuando dir = 0 es ascendente, con dir = 1 es descendente. Dos de las salidas son el valor de conteo. La tercera, denominada **Z**, se activa al llegar al último estado.

```
Name Contador;
Partno 12345;
Date 03/24/20;
Revision 01;
Designer CDA;
Company UABC;
Assembly None;
Location None;
Device g22v10;

/** Inputs **/
Pin 1 = clk; /* Senal de Reloj */
Pin 2 = dir; /* Direccion de conteo */
Pin 3 = rst; /* Reinicio */

/** Outputs **/
Pin [22..21] = [Q1..0]; /* Salidas del Contador */
Pin 20 = Z; /* Salida Z */

/** Declarations and Intermediate Variable Definitions **/
field contador = [Q1..0]; /* Campo de bits (Bit field) del Contador*/
$define S0 'b'00 /* Estados del Contador */
$define S1 'b'01
$define S2 'b'10
$define S3 'b'11
```

```
/** Logic Equations **/
contador.AR = rst; /* Reset asincrono */
contador.SP = 'b'0; /* Preset sincrono*/

Sequenced contador {
present S0 if !dir next S1;
            if  dir next S3 OUT Z;

present S1 if !dir next S2;
            if  dir next S0;

present S2 if !dir next S3 OUT Z;
            if  dir next S1;

present S3 if !dir next S0;
            if  dir next S2;
}
```

Listado 2. Descripción de un circuito secuencial en CUPL.

Simulación del código del Listado 2 en WinSim