

Universidad de Chile  
Facultad de Ciencias Físicas y Matemáticas  
Departamento de Ingeniería Eléctrica

## **Plataforma de desarrollo Altera®**

Documento preparado por Alejandro Sáez Madaín  
Santiago 29 de Octubre 2003

<b>Introducción .....</b>	<b>3</b>
<b>Convención tipográfica .....</b>	<b>4</b>
<b>Primer diseño .....</b>	<b>5</b>
Compilación .....	9
Simulación .....	10
Asignación de pines .....	11
<b>Dispositivos lógicos programables.....</b>	<b>14</b>
PLDs.....	15
CPLDs y FPGAs .....	16
Arquitecturas de las familias MAX y FLEX de Altera.....	17
Familia MAX.....	17
Familia FLEX.....	18
Aplicaciones de los FPGAs .....	20
<b>Plataforma de desarrollo Altera UP1 y UP2 .....</b>	<b>20</b>
Descripción de la plataforma altera.....	21
Configuración de los jumpers.....	21
<b>Max-Plus II .....</b>	<b>22</b>
Descripción del software .....	22
Descripción de un sistema lógico.....	23
Editor de texto .....	23
Editor de esquemáticos .....	23
Editor de formas de onda.....	24
Archivos de trabajo.....	28

# Introducción

El presente documento tiene como objetivo entregar al lector las herramientas necesarias para que desarrolle y pruebe sus diseños lógicos ocupando para esto la plataforma de desarrollo UP2 de ALTERA.

Las plataformas de desarrollo UP1 y UP2 de ALTERA están compuestas por dos dispositivos lógicos programables ( un CPLD y un FPGA). Además poseen una serie de recursos auxiliares, para poder interactuar con el exterior (*Led* emisores de luz, pulsadores, *switchs*, puerto PS/2 y VGA)

En este documento se realiza una descripción de la plataforma de desarrollo UP1 y UP2 (indicando las diferencias que existen entre ambas), como así también se ve en detalle las herramientas de software disponibles que permiten al usuario: crear , simular sistemas digitales y configurar la plataforma de desarrollo.

Este documento contiene una breve reseña del lenguaje VHDL, que es uno de los lenguajes utilizados para síntesis de sistemas digitales de gran nivel de integración. También se hace una descripción del concepto de jerarquía en un diseño digital. Para una mayor comprensión se adjuntaron una serie de proyectos que el lector puede realizar o puede modificar dependiendo de sus necesidades. Todos los proyectos aquí presentados fueron realizados y funcionan conforme lo estipulado en este documento.

Esta es la primera versión de este tutorial, por lo cual el lector puede encontrar errores o con conceptos que no quedan satisfactoriamente claros. El autor agradece de antemano todo tipo de críticas y correcciones que permitan generar un documento de mayor nivel técnico y pedagógico.



## Convención tipográfica

- **Arial:** *código en VHDL.*
- **Arial bold:** *comando de VHDL.*
- **Bold italics:** *hace referencia a un software.*
- **Italics:** *palabra técnica de habla inglesa.*
- **Italics subrayado:** *ubicación en el sistema de archivos MSDOS.*
- **Subrayado:** *nota de importancia.*
- **Verdana:** nombre de una archivo.

## Primer diseño

La mejor forma de aprender a usar un dispositivo es utilizándolo. A continuación se presenta un proyecto simple que permitirá al usuario familiarizarse con el uso de la plataforma de desarrollo UP2 o UP1, y con el software MAX-PLUS II 10.1 BASELINE.

El primer proyecto para que desarrolle el usuario, consistirá en programar el FPGA de la plataforma de desarrollo UP2 de forma que funcione como un decodificador de 7 segmentos. Para realizar lo anteriormente expuesto se van a seguir los siguientes pasos ( este esquema de acción es aplicable en todos los proyectos):

- Crear un directorio en donde residirán los distintos archivos que componen el proyecto, en adelante a este directorio la llamaremos “directorio de trabajo”.
- Crear dentro del directorio de trabajo el archivo que va a ser la entrada al compilador. Para crear el archivo de entrada podemos utilizar los editores que vienen incluidos en el software *MAX-PLUS II*.
- Especificar el nombre del archivo que describe el sistema principal y el nombre del proyecto.
- Asignar un dispositivo asociado al proyecto.
- Abrir la ventana ***MAX-PLUS II compiler*** con  y seleccionar el botón *start*. Si no hay errores en el proyecto, el compilador creará una serie de archivos auxiliares que tienen como finalidad proveer de información necesaria para el simulador y el analizador temporal.
- Si el proyecto compiló exitosamente el usuario puede usar las siguientes funcionalidades.
  - *Timing Analysis*, esta funcionalidad permite determinar el tiempo de retraso que existe entre las diversas entradas y salidas. Para ejecutarlo se debe seleccionar *analysis mode*, y escoger el botón *start*.
  - Para ejecutar una simulación, es necesario crear un archivo que contenga la forma de onda de las entradas (*waveform editor*). Después se debe abrir la ventana *MAX-PLUS II simulator* con , y se debe seleccionar el botón *start*.
- Abrir la ventana *MAX-PLUS II programmer*, y conectar la plataforma de desarrollo al PC, a través del dispositivo byte-blaster.
- Elegir el botón *configure* para cargar el programa en la FPGA. Si se utiliza la CPLD se debe seleccionar el botón *program* (las distintas opciones se deshabilitan dependiendo del dispositivo).

Es importante señalar que para especificar el funcionamiento de un sistema digital se puede utilizar entradas grafica (Editor de esquemáticos) o entrada de texto (usando las convenciones que imponen los distintos lenguajes de descripción). En la figura 1 se puede apreciar un diagrama de las distintas etapas de diseño e implementación.

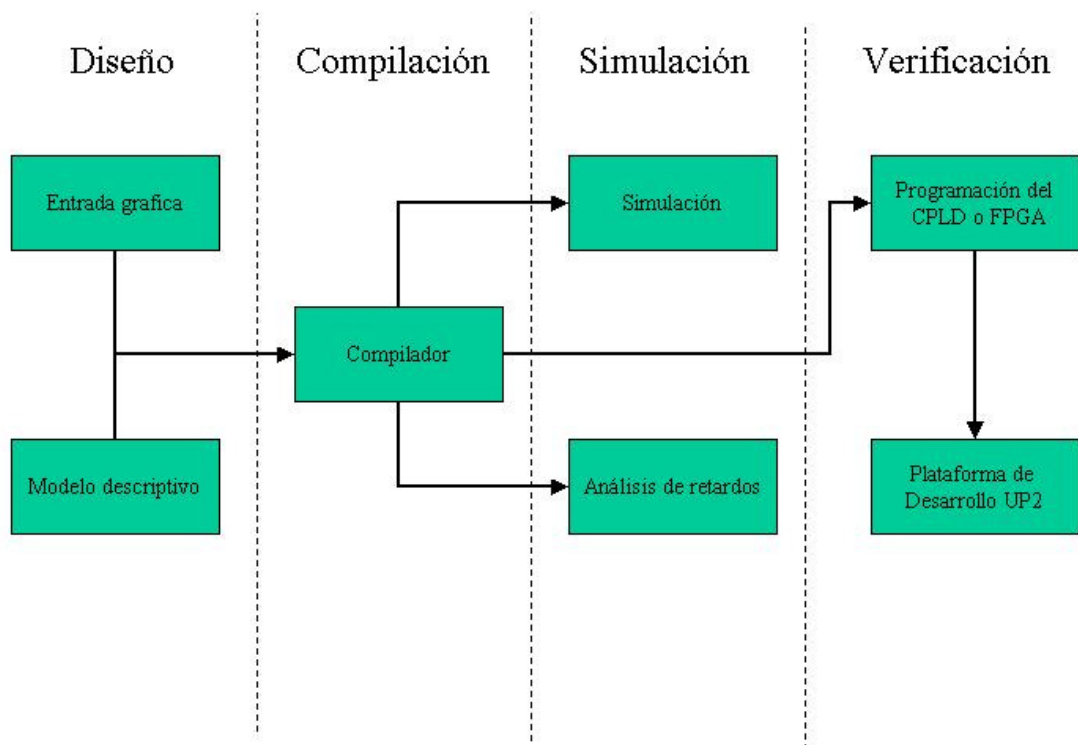


Figura 1: Etapas del proceso de diseño en la plataforma ALTERA

A continuación se describirá detalladamente el proceso de: creación, compilación, simulación de un sistema digital utilizando para esto el programa MAXPLUSII. También se explicará el proceso de configuración del FPGA que está en la plataforma de desarrollo.

Para empezar se debe ejecutar el programa MAX-PLUS II 10.1 BASELINE, ejecutando *inicio/programas/ MAX-PLUS II 10.1 BASELINE/ MAX-PLUS II 10.1 BASELINE*. Aparecerá la área de trabajo que se indica en la figura 2.

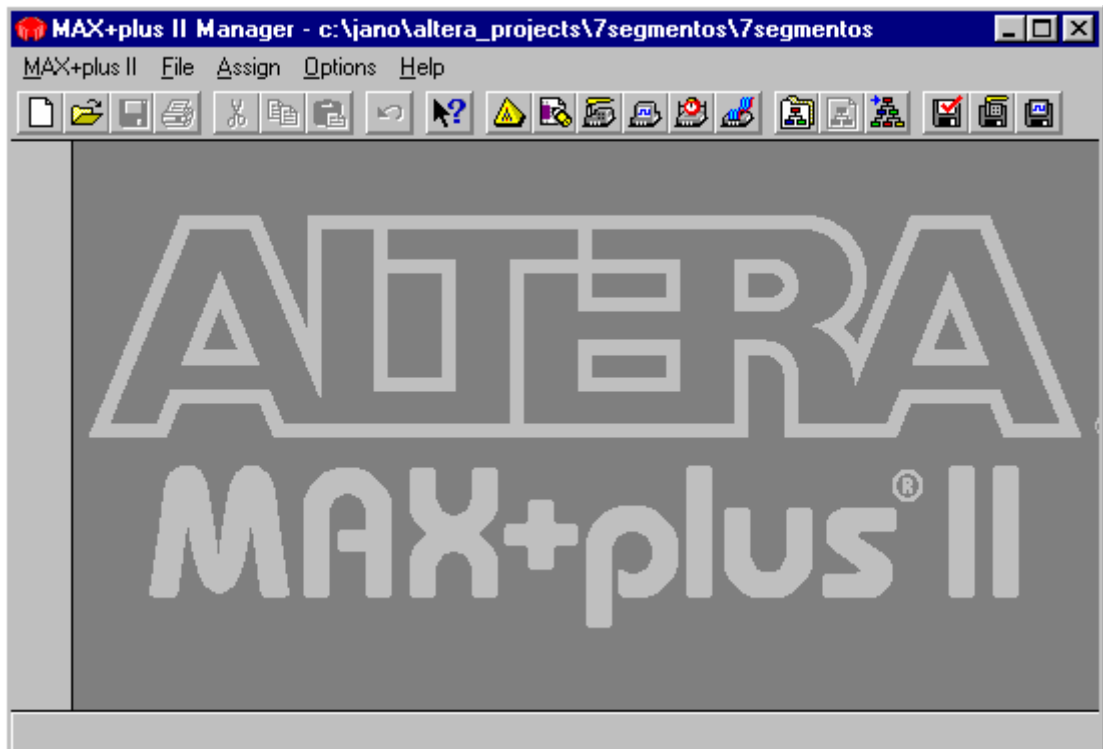


Figura 2: Área de trabajo del software MAX-PLUS II.

Ahora se creará un nuevo archivo con la descripción en forma esquemática de nuestro sistema. Para ello seleccionamos dentro del área de trabajo de MAX-PLUS II la opción *file*⇒*new* donde aparecerá el formulario que sale indicado en la figura 3.

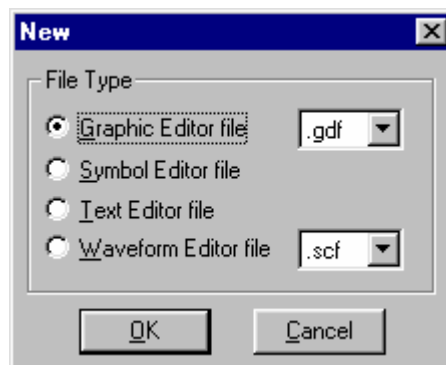


Figura 3: Formulario para seleccionar el tipo de archivo que se desea crear

Se seleccionará la opción *Graphic Editor File*, para esto seleccionamos la extensión *.gdf*. Ahora se desplegará un área en donde podremos editar nuestro sistema, en forma esquemática. Ver figura 4.

En esta ventana de trabajo tenemos la opción de seleccionar diversos elementos desde librerías (por ejemplo hay una librería que tiene elementos lógicos estándar, como compuertas y flip-flops, y otra con toda la familia de circuitos integrados TTL)

Tenemos dos opciones de archivo: *.gdf* y *.sch* la primera es propietaria del software ALTERA y la segunda es compatible con el software ORCAD

En nuestro proyecto utilizaremos el modelo del circuito integrado 7447 perteneciente a la familia TTL, el cual funciona como decodificador para un *display* de 7 segmentos

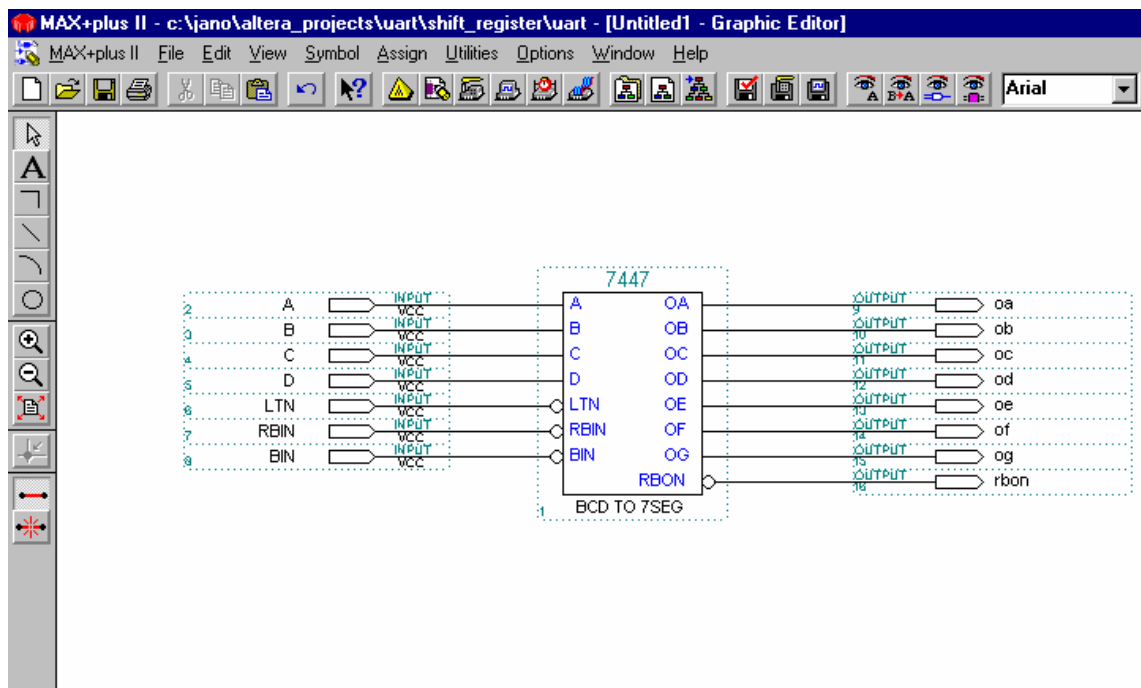

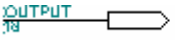


Figura 4: Área de trabajo del editor gráfico. En el lado izquierdo se encuentra la barra de herramientas del editor gráfico.


Las entradas al sistema se definen con el símbolo , y las salidas se definen con el símbolo .

En todos los símbolos de *input*, aparece la sigla VCC, esto no tiene ningún significado especial para nuestro proyecto.

Ahora que esta definido el sistema, falta determinar como se implementará el sistema en la plataforma de desarrollo. Esto significa que se debe asignar un *pin* o terminal a las salidas y entradas del sistema. Lo anterior será explicado en detalle en el capítulo “asignación de pines”, primero se abordará el tema de la compilación y simulación.



## Compilación

Para compilar el proyecto se selecciona el botón *compiler* , como resultado de lo anterior aparecerá una ventana que indica el estado del proceso de compilación. Ver figura 5.

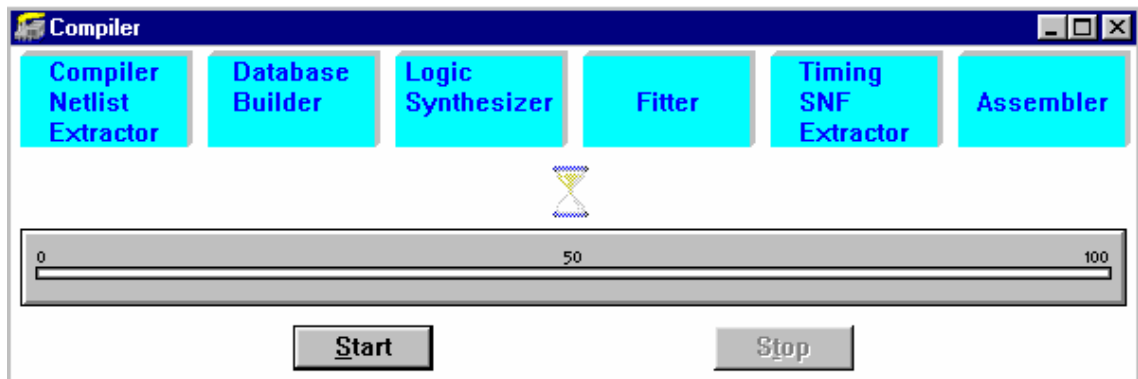


Figura 5: Información del estado del proceso de compilación.

- Compiler Netlist Extractor: Durante este proceso el compilador crea un archivo binario con extensión *.cnf*. Este archivo contiene información de los parámetros y valores usados en las distintas entradas del proyecto. Cada vez que se hace una modificación en los archivos de entrada, este archivo se modifica. En esta etapa se crea, también, un archivo con extensión *.hif* que contiene información referente a la conexión que hay entre los distintos elementos del proyecto. También se crea un archivo con extensión *.ndb* el cual contiene información referente al nombre de los distintos terminales de entrada y salida.
- Database Builder: En este proceso se concatenan los distintos archivos *.cnf* pertenecientes a cada uno de los elementos que componen la jerarquía del sistema. El resultado de este proceso es la creación de una base de datos única que contiene información de conectividad de todo el proyecto. En esta etapa es cuando se detecta una eventual falla en la conexión de los distintos componentes.
- Logic Synthesizer: En esta etapa del proceso de compilación, se aplican diversos algoritmos tendientes a reducir el tamaño de la implementación. Dicho de otra forma en este proceso se optimizan los recursos del dispositivo lógico que se está programando.
- Fitter: Usando la información obtenida durante el proceso *Database builder*, se asigna la mejor locación dentro de los recursos del dispositivo de los elementos lógicos a utilizar.
- Timing SNF extractor: Durante este proceso se crea un archivo con extensión *.snf* que contiene la información referente a los tiempos de retrasos asociado al dispositivo sobre el cual se está trabajando.
- Assembler: En este proceso se crea un archivo con la sintaxis necesaria para ser descargado en el dispositivo lógico que se desea programar. En este proceso se crean archivos con extensión *.pof* y *.sof*.

## Simulación

Ahora se esta en condiciones de simular el sistema, para esto es necesario crear un archivo que contenga la información de las entradas que estimulan al sistema. Se usará el editor de formas de onda. Seleccionando *file*⇒*new* se desplegará un formulario que se indica en la figura 6. Se seleccionará *Waveform Editor file*

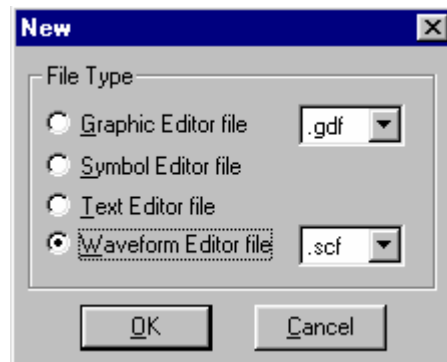


Figura 6: Formulario para crear una nueva

En la figura 7 se puede apreciar el aspecto que tiene el editor de formas de onda. La escala de tiempo es de 50[nS] y por defecto el tiempo máximo es 1000[nS], aunque se puede aumentar dicho tiempo seleccionando *file*⇒*End Time*.

Ahora se debe seleccionar las señales que se desean estudiar, en una simulación no es necesario que se incluyan todas las señales que intervienen en el funcionamiento del sistema. Para agregar una señal a la simulación, se debe colocar el cursor del *mouse* en el área de la columna *Name*, después se debe oprimir el botón derecho del *mouse* y elegir la opción *add node*, a continuación saldrá un formulario en donde el usuario puede elegir la señal que desea analizar. Las señales que el usuario puede hacer parte de la simulación no sólo son las de entrada y salida del sistema, también puede acceder a registros y señales internas.

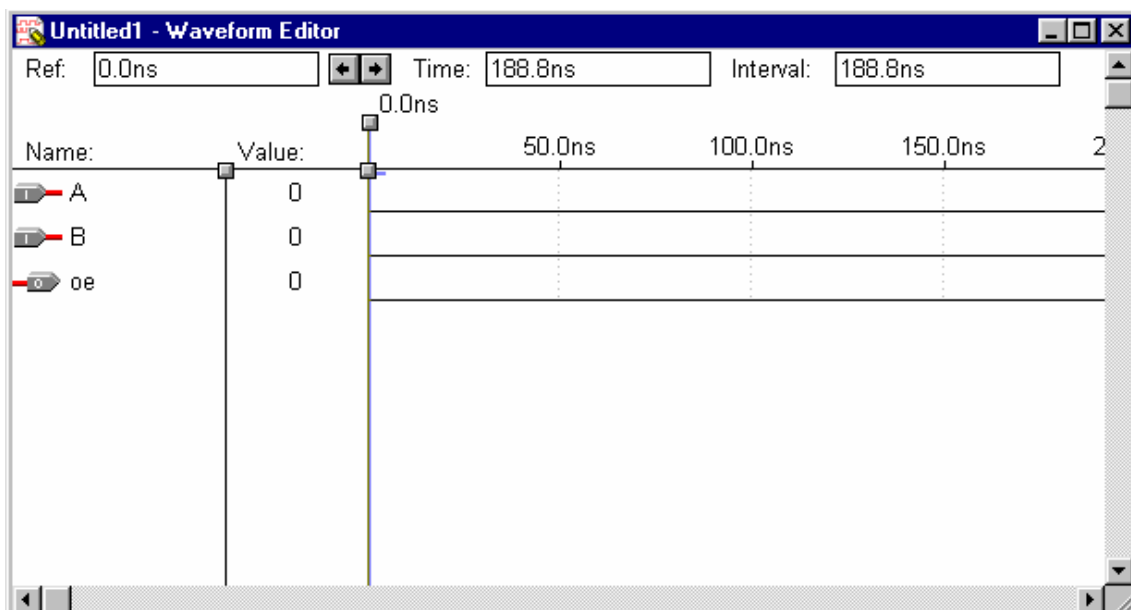



Figura 7: Editor de formas de onda.

Una vez seleccionadas todas las señales que se analizarán, es necesario configurar las entradas, vale decir señalar cuál va a ser la función del tiempo que tengan las señales de entrada. El editor de formas de onda, provee las herramientas necesarias para editar un tren de pulsos, un contador etc. Terminado el proceso anterior, se está en condiciones de ejecutar la simulación, para lo cual se debe seleccionar el botón *simulator*  que se encuentra en el área de trabajo principal, aparecerá la ventana que se indica en la figura 8:

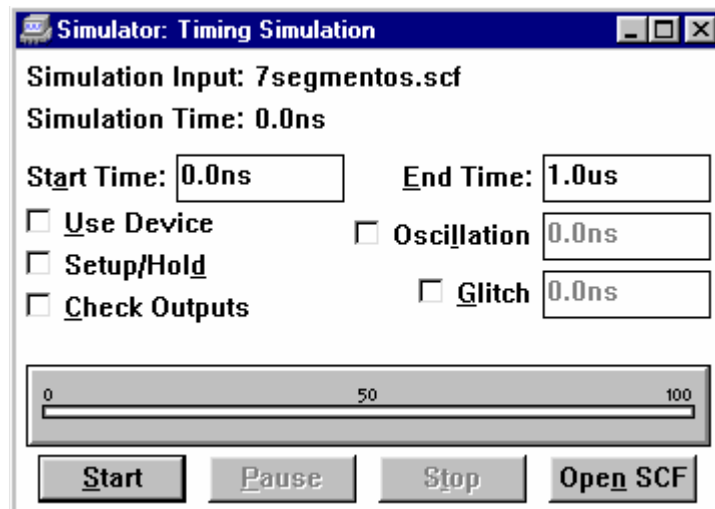


Figura 8:

Con el botón *start* se empieza la simulación, una vez que esta haya concluido, el archivo que contiene las formas de onda será modificado, obteniendo el valor esperado para las salidas del sistema.

### ***Asignación de pines***

En esta etapa del proyecto, es necesario compilar previamente el código, una vez que este compilado, y por ende no haya errores, estamos en condiciones de realizar la asignación de pines.

Para realizar la asignación de pines es necesario especificar el modelo de dispositivo que estamos ocupando. En este ejemplo utilizaremos el FPGA EPF10K70RC240-3 (perteneciente a la plataforma de desarrollo UP2. En caso de ocupar la plataforma de desarrollo UP1 se debe seleccionar el modelo EPF10K20RC240). Para determinar el dispositivo se selecciona *Assign⇒Device*, se presentará un formulario en donde el usuario deberá seleccionar el dispositivo de una lista de modelos, Ver figura 8.

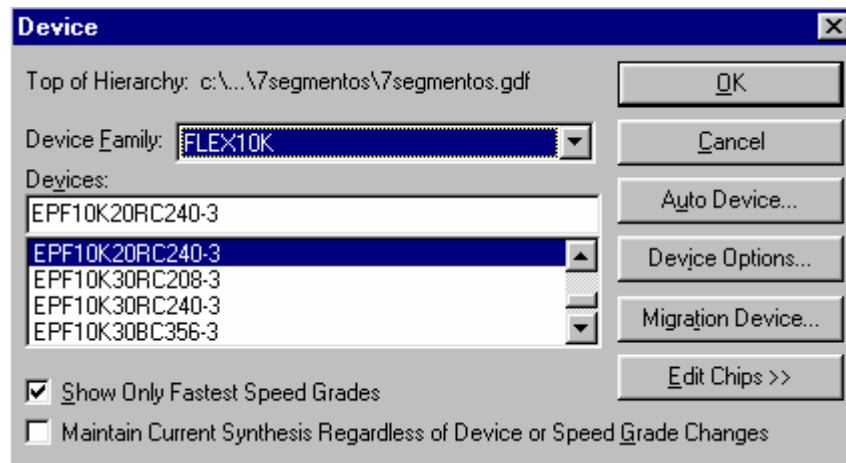


Figura 9: Formulario con la lista de dispositivos que pueden ser programados

Para asignar los pines se debe seleccionar dentro del programa MAX-PLUS II BASELINE la opción *Assign⇒Pin⇒Location/Chip*, aparecerá el formulario que se indica en la figura 10:

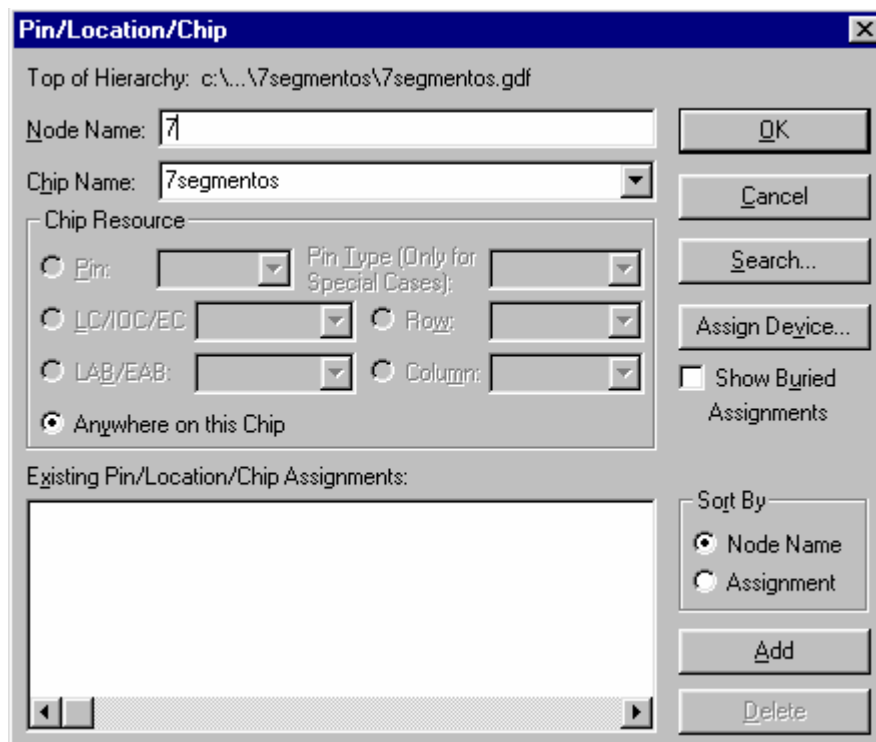


Figura 10: formulario de entrada para la asignación de pines

Para agregar una nueva Asignación de *pin*, se debe proceder en la siguiente forma:

- Hacer clic en el botón *Search*.
  - Se abrirá un formulario en donde el usuario deberá seleccionar la entrada salida a la cual desea asignar un *pin*.
- Asignar el número de *pin* a la entrada o salida seleccionada (si el numero de *pin* no es compatible con el dispositivo seleccionado, el programa reportara dicho error)

- Se deben realizar los pasos anteriores, para todos las entradas y salidas del proyecto


Al finalizar el proceso de asignación, el formulario para la asignación de pines tendrá el aspecto que se indica en la figura 11.

Figura 11: formulario de asignación de pines ya completado

Para probar el correcto funcionamiento del diseño se utilizará uno de los dos *displays* que están conectados con el FPGA, y el banco de *switchs* que también esta conectado con el FPGA. La asignación es la que se presenta en la tabla 1(para obtener mayores detalles remítase al documento *Univesity Program Design Package*):

Pin	Asignación
A	41
B	40
C	39
D	38
BIN	36
LTN	35
RBIN	34
Oa	6
Ob	7
Oc	8
Od	9
Oe	11
Of	12
Og	13
Rbon	14

Tabla 1: Asignación de pines para el display y el banco de switchs.

Ahora que el proyecto ya esta definido dentro de la plataforma de desarrollo, es el momento de configurar el FPGA. Para hacer esto es necesario programar el FPGA. Para iniciar el proceso de programación se utiliza el botón *programmer* , después de seleccionar la opción *programmer* aparecerá un cuadro de dialogo como el que se indica en la figura 12.

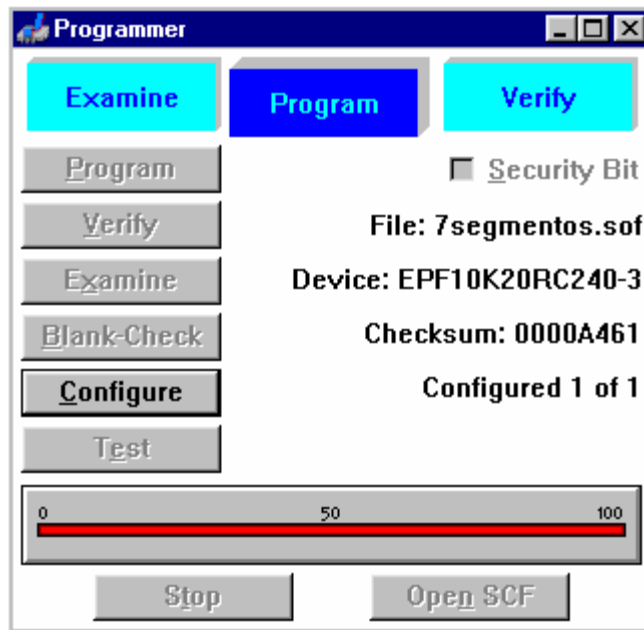


Figura 12: Cuadro dialogo del proceso de la aplicación para programar

Con el botón *configure* se configura la FPGA, es necesario recordar que para configura el FPGA este debe estar polarizado, y el dispositivo *byte-blaster* (*byte-blaster* es la interfaz entre el P.C. y la plataforma de desarrollo) debe estar conectado al PC.

En el caso de este proyecto, sólo estará activo el botón *Configure*, ya que un FPGA pierde su configuración en el momento en que se deje de suministrar energía. El botón *Program* estará activo en caso de que se este trabajando con el CPLD, los CPLD no pierden su configuración en caso de que se corte el suministro de energía.

## Dispositivos lógicos programables

Existe una amplia gama de circuitos integrados que tienen como finalidad la implementación de diseños digitales. Hay circuitos integrados en que su función esta definida por el fabricante, por ejemplo los c.i. de la familia TTL y CMOS. En este caso es el usuario el que define la función del sistema interconectando diversos circuitos integrados.

También existe una clase de circuitos integrados en los cuales el usuario define la función que este realizará, dentro de este conjunto de circuitos integrados podemos encontrar

## PLDs

Dentro de la familia de los PLD (*Programmable Logic Devices*) encontramos los PLAs (*Programmable Array Logic*) y los PALs (*Programmable Array Logic*). Estos dispositivos permiten programar por parte del usuario una función lógica. Dicho de otra forma los PLD permiten implementar un circuito combinacional. En la figura 13 se muestra un ejemplo de una pequeña **PLA** antes de ser programada. Se puede observar que posee múltiples entradas, cada una de las cuales tiene a su vez su valor negado.

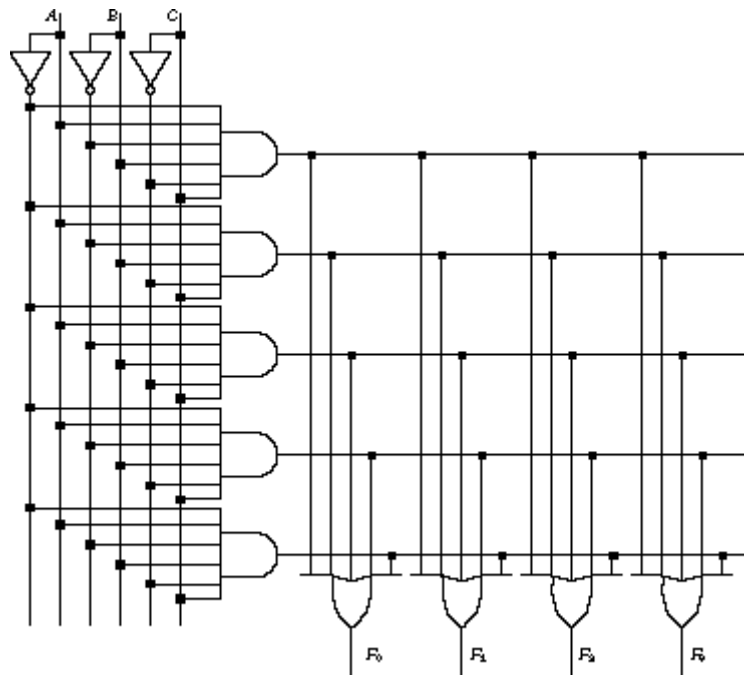


Figura 13: PLA antes de ser programada.

Si por ejemplo queremos implementar la función lógica, cuya tabla de verdad es la indicada en la tabla 2:

Entradas			s.o.p.	Salidas			
A	b	c		F <sub>0</sub>	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>
1	1	-	ab	0	1	1	0
-	0	1	c/b	0	0	0	1
1	-	0	a/c	0	1	0	0
-	0	0	/b/c	1	0	1	0
1	-	-	a	1	0	0	1

Tabla 2: tabla de verdad de una función lógica

Después debemos “quemar” ciertos puntos que unen las conexiones internas del PLA. Para el caso del ejemplo anterior, en la figura 14 se puede ver un esquema de cómo quedaría el PLA después de ser programado. Para resumir el proceso de diseño con PLA, se deben tener en cuenta los siguientes puntos:

- Reducir la función a suma de productos

- Eliminar ciertos contactos entre los distintos recursos del PLA, de forma de obtener la suma de productos deseada

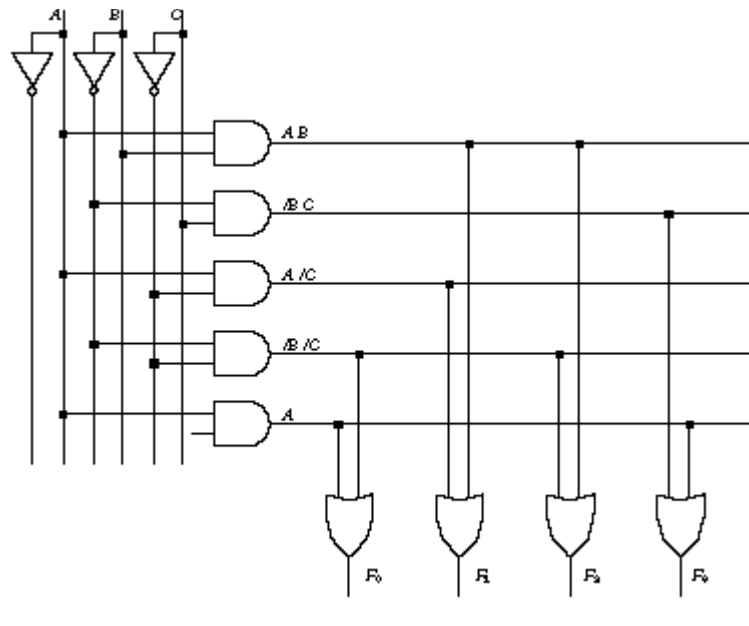


Figura 14: modelo teórico del PLA después de ser programado.

## CPLDs y FPGAs

Los CPLDs y FPGAs están constituidos internamente por un conjunto de entidades básicas llamadas *cell* o *logic element (LE)*. Cada LE puede ejecutar la función de una red de varias compuertas lógicas que pueden alimentar la entrada de un *flip-flop*.

Los elementos lógicos pueden ser ordenados en una columna o una matriz en el circuito integrado. Para ejecutar funciones más complejas se pueden interconectar varios LE. Para realizar esto existe una red de interconexión que puede ser programada, esta red de interconexión está contenida en el mismo circuito integrado.

En los FPLD de mayor volumen, existe una red dedicada para la señal de reloj. Esta red es usada para distribuir en forma eficiente la señal de reloj a todos los LE.

Así como en las memorias su volumen se describe en términos de bits que puede almacenar, en los FPLD su tamaño se describe en términos de compuertas usables o equivalente, una medida de capacidad, es el equivalente en compuertas NAND de dos entradas.



## Arquitecturas de las familias MAX y FLEX de Altera

### Familia MAX

Los dispositivos pertenecientes a la familia MAX7000S, por ejemplo el CPLD EPM7128 perteneciente a la plataforma UP2, se configuran programando una memoria interna del tipo EEPROM.

Los dispositivos de la familia max7000s están compuestos por entre 32 y 256 *macrocells*. En la figura 15 se puede ver la estructura de una *macrocell*.

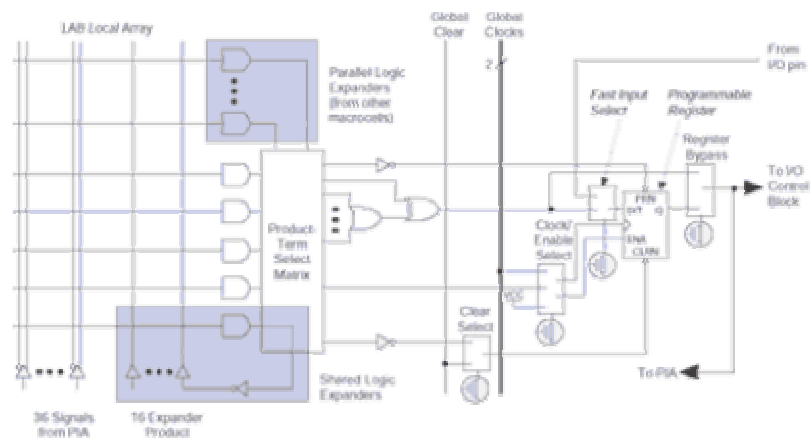


Figura 15: Estructura de una *macrocell*.

Al igual que los PLA, las *macrocell* basan su funcionamiento en la síntesis de una función *booleana* a una suma de productos. En el caso de una *macrocell* esta posee cinco compuertas AND, cada una de las cuales tiene múltiples entradas, también se puede expandir el número de compuertas AND para así incluir productos provenientes de otra *macrocell* adyacente, o del **Programmable Interconnect Array (PIA)**. Las compuertas NAND están conectadas a las entradas de una compuerta OR, que es la que finalmente ejecuta la suma de productos. Esta red está finalmente conectada a un *flip-flop* programable (es decir puede ejecutar la función de un *bypass*, o bien funcionar como *flip-flop* tipo SR, T, D o JK). Finalmente la salida de una *macrocell* se conecta con el PIA.

Las *macrocell* se agrupan en entidades llamadas **Logic Array Blocks (LABs)** como se muestra en la figura 16.

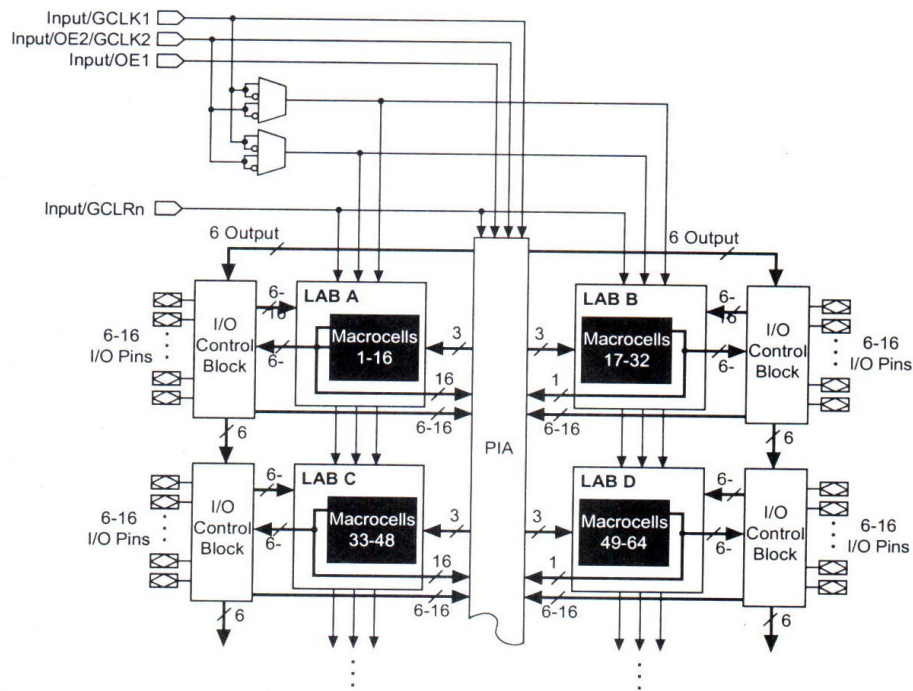


Figura 16: Estructura de los LABs (Logic Array Blocks).

## Familia FLEX

FLEX (Flexible Logic Element Matrix) es una familia de CPLD cuyos dispositivos pueden tener entre 10,000 y 250,000 compuertas. Los dispositivos de la familia FLEX se configuran programando la memoria RAM estática que tienen en su interior. En el momento en que el CPLD se desconecta de la fuente de alimentación, se pierde el contenido de su configuración.

Al igual que en el caso de la familia MAX, los dispositivos de la familia FLEX poseen entidades básicas en donde se programan funciones *booleanas*. En la figura 17 se muestra un diagrama de la entidad básica de la familia FLEX

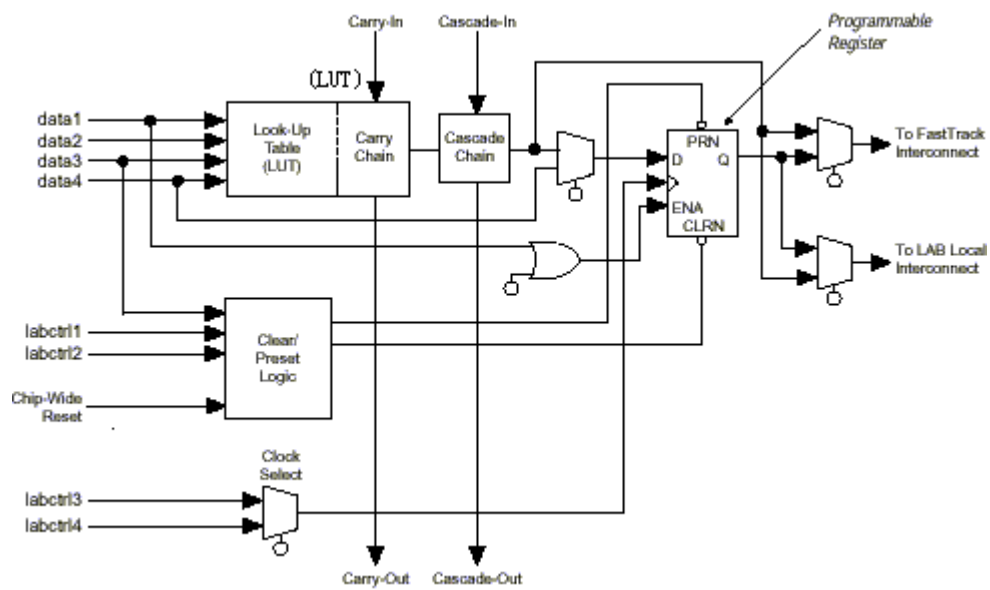


Figura 17: entidad básica de la familia FLEX.

## Aplicaciones de los FPGAs

Los FPGAs son dispositivos ampliamente usados en el proceso de diseño de circuitos integrados digitales, por ejemplo un FPGA resulta útil para diseñar un microprocesador, o un DSP. Los FPGA realizan la función de prototipo de un circuito lógico.

Los FPGA también encuentran aplicación como interfaz para probar otros circuitos integrados, por ejemplo se podría ocupar un FPGA para conectar una CPU con una memoria RAM y una memoria ROM, y así poder analizar su funcionamiento sin la necesidad de crear una hardware especial para interconectar los diversos dispositivos.

## Plataforma de desarrollo Altera UP1 y UP2

La plataforma de desarrollo UP (*University Program*), fue desarrollada para satisfacer las necesidades de la enseñanza de sistemas digitales. La plataforma provee de las herramientas necesarias para la implementación de diseños lógicos, la plataforma incluye las siguientes características:

- MAX-PLUS II software
- Placa UP
  - FPGA modelo EPF10K20 para la placa UP1, y EPF10K70 para la placa UP2
  - CPLD modelo EPM7128S para las placas UP1 y UP2
- ByteBlasterMV, adaptador para el puerto paralelo

En la figura 18 se puede apreciar la vista superior de la plataforma de desarrollo

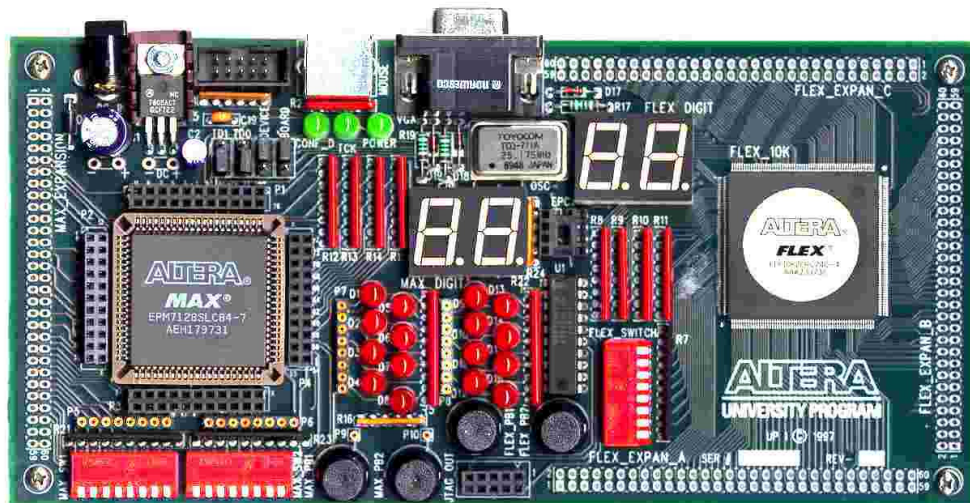


Figura 18: Fotografía de la plataforma de desarrollo UP2 de ALTERA

## ***Descripción de la plataforma altera***

La plataforma de desarrollo ALTERA UP1 y UP2 (*University Program*) esta compuesta por los siguientes elementos:

- Un FPGA 10K20RC240 en la UP1 y 10K70RC240 en UP2.
- Un CPLD modelo
- Dos Display dobles de 7 segmentos. Uno conectado al FPGA y otro con terminales libres.
- 16 diodos emisores de luz, agrupados en 8 diodos conectados directo al FPGA y 8 con terminales libres.
- 4 pulsadores. Dos conectados directamente al FPGA y 2 con terminales libres.
- 24 switchs. Agrupados en 8 switchs conectados directo con el FPGA y 16 con terminales libres.
- Un oscilador de cristal de cuarzo que esta conectado con el FPGA.
- Un conector DB15 con la configuración necesaria para ser usado como termina VGA
- Un conector PS/2 para ser usado como interfaz física entre teclados o mouses.
- Un conector para fuente de poder de 9 V

## ***Configuración de los jumpers***

A continuación se presentan el conjunto de configuraciones posibles de los *jumers* de la tarjeta de desarrollo:

### **Programación del CPLD EPM7128S**

TDI TDO DEVICE BOARD

C1	C1	C1	C1
C2	C2	C2	C2
C3	C3	C3	C3

### **Programación del FPGA FLEX**

TDI TDO DEVICE BOARD

C1	C1	C1	C1
C2	C2	C2	C2
C3	C3	C3	C3

### **Programación de ambos dispositivos**

TDI TDO DEVICE BOARD

C1	C1	C1	C1
C2	C2	C2	C2
C3	C3	C3	C3

# Max-Plus II

## *Descripción del software*

Las plataformas de desarrollo UP1 y UP2 de Altera incluyen el disco de instalación del software MAX-PLUS II BASELINE. Dicho software permite al usuario crear proyectos de sistemas digitales usando el lenguaje VHDL o usando un editor grafico, como se vio anteriormente. Además el usuario puede simular sus proyectos usando como entrada el editor de formas de onda. Por último el software provee de la interfaz necesaria para programar el CPLD y el FPGA que vienen incluidos en la plataforma de desarrollo.

El software MAX-PLUS II BASELINE tiene las siguientes capacidades:

- Posee editores de texto, con plantillas prefabricadas para distintos leguajes de descripción de hardware (VHDL, VERILOG, ALTERA).
- Posee editor de circuitos esquemáticos.
- Posee un editor de formas de onda que puede ser utilizado para describir un sistema digital, como para determinar las entradas en una simulación.
- Posee un compilador que es capaz de recibir como entrada tanto archivos con la descripción del sistema (VHDL, VERILOG, ALTERA), como archivos con circuitos esquemáticos.
- Puede ejecutar simulaciones, usando como archivo de entrada uno creado por el editor de formas de onda.
- Puede programar un dispositivo CPLD (primero es necesario especificar el modelo y los pines de entrada y salida que se usaran).

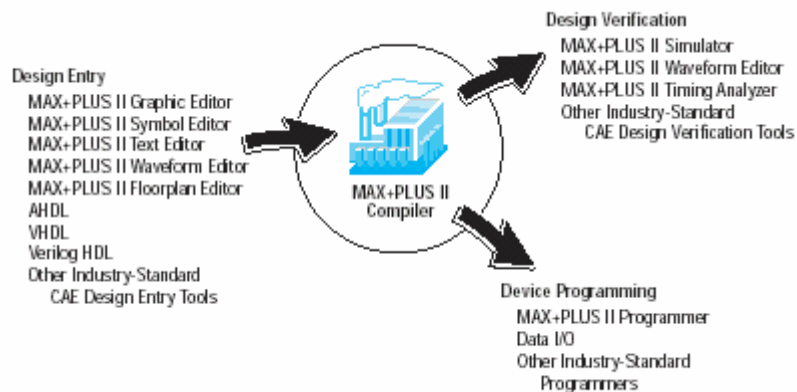


Figura 19: Esquema de entradas y salidas del compilador del software MAX-PLUS II.

En la figura 19 se presenta un esquema en donde se ve el programa MAX-PLUS II como una “caja negra” en donde la entrada puede ser cualquier tipo de descripción de un sistema digital, y la salidas son el resultado de la simulación del sistema y el archivo de configuración, el cual será finalmente descargado en el dispositivo lógico programable ( en este caso un FPGA o un CPLD).

## ***Descripción de un sistema lógico***

Para describir un sistema lógico se puede utilizar el editor gráfico, en donde se puede ver la realización a nivel de compuertas y circuitos integrados del diseño propuesto, o bien podemos utilizar una descripción del comportamiento de nuestro sistema, para esto usamos el lenguaje VHDL.

### **Editor de texto**

El software MAX-PLUS II, provee de un editor de texto que puede ser empleado para editar programas en VHDL. El editor es capaz de reconocer instrucciones y modificar el color de la fuente, esto facilita la tarea al programador. El editor de texto sólo reconoce la sintaxis de un lenguaje en particular, a partir del momento en que el archivo es almacenado con la extensión indicada (en el caso de vhd la extensión es .vhd el editor de texto también resulta útil cuando se esta programando en Verilog, o ALTERA)

### **Editor de esquemáticos**

Como lo indica el título este editor permite hacer la descripción de un sistema digital a partir de elementos gráficos como los son la representación de una compuerta o de un circuito integrado de mayor complejidad. El software NO es capaz de pasar un archivo esquemático a VHDL.

Se pueden crear archivos con extensión *.gdf* y *.sch* el primer tipo de archivo sólo se puede utilizar dentro del software ALTERA, el segundo, en cambio, es compatible con el software ORCAD.

### **Ejemplos del editor de esquemáticos**

#### **Ejemplo 1: Contador decimal de dos dígitos**

En el siguiente ejemplo se creará a través del editor de esquemáticos un sistema digital que funciona como contador de dos dígitos decimal. Para eso seleccionamos los siguientes circuitos integrados que están incluidos en la librería \up2core\ (para agregar una librería, se debe seleccionar la opción *Option*  $\Rightarrow$  *user libraries* donde aparecerá el formulario que se indica en la figura 19.

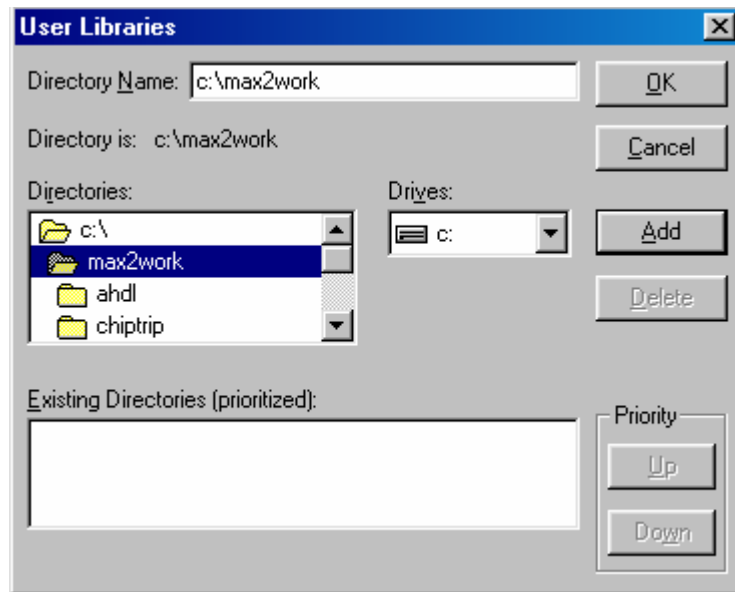


Figura 19: Formulario para añadir librerías

Se debe seleccionar el directorio donde se encuentran los elementos se desean agregar y después se debe hacer clic en el botón *add*. El resultado de esta operación se puede ver en la figura 19.

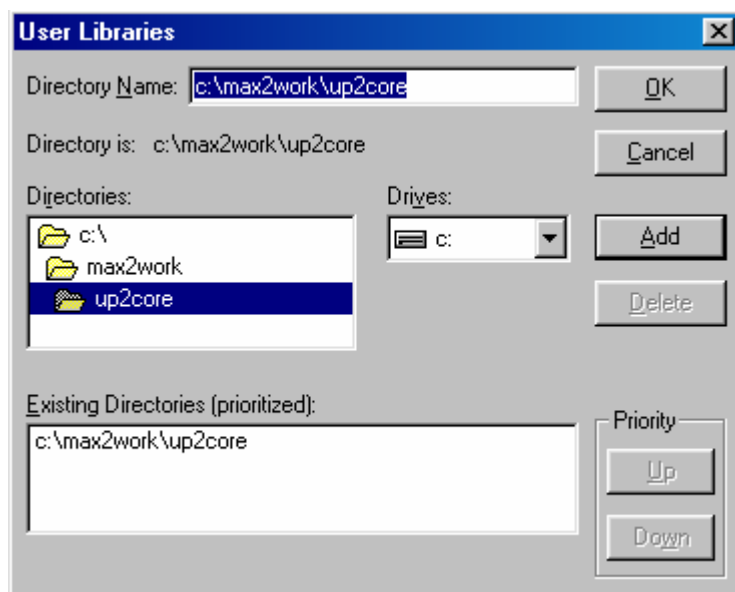


Figura 19: Formulario para añadir librerías después de seleccionar una librería

## Editor de formas de onda

Este editor permite crear dos tipos de archivos, uno con extensión .scf y otro con extensión .wdf. Los archivos con extensión .scf son utilizados como entrada para el simulador, el usuario configura la forma de onda de las señales que desea estudiar, y el simulador determina la forma de onda de las señales de salida y los registros que componen el sistema bajo estudio. En el proceso de edición de este archivo, el usuario debe tener en cuenta que el es el responsable de incluir las señales necesarias para obtener resultados en la simulación.



Los archivos con extensión *.wdf*, sirven para describir un sistema digital, a nivel de formas de onda de las entradas y salidas. Esta forma de entrada al compilador resulta sumamente útil cuando tenemos la descripción en función del tiempo de un sistema digital.

Ejemplos del editor de forma de onda:

Por ejemplo vamos a describir con el editor de formas de onda un sistema que se comporta como una compuerta OR. Para eso creamos un nuevo archivo con la opción *File⇒New*, En el formulario que aparecerá elegimos la opción *wave form editor* con la extensión *.wdf* como se indica en la figura 20.

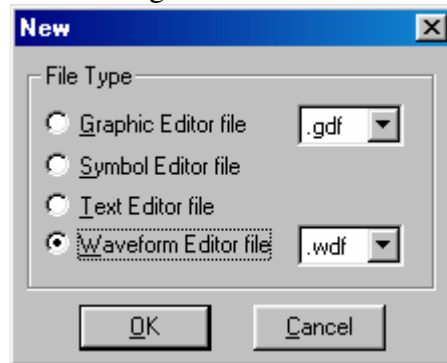


Figura 20: Formulario para la creación de un nuevo archivo de entrada.

Para añadir las señales que van a ser parte del sistema que se desea sintetizar (en este caso van a ser dos señales de entrada y una de salida) se debe posicionar el puntero del *mouse* en la columna *name* y hacer click con el botón derecho, a continuación aparecerá un cuadro dialogico en donde se debe seleccionar la opción *add node*, ver figura 21.

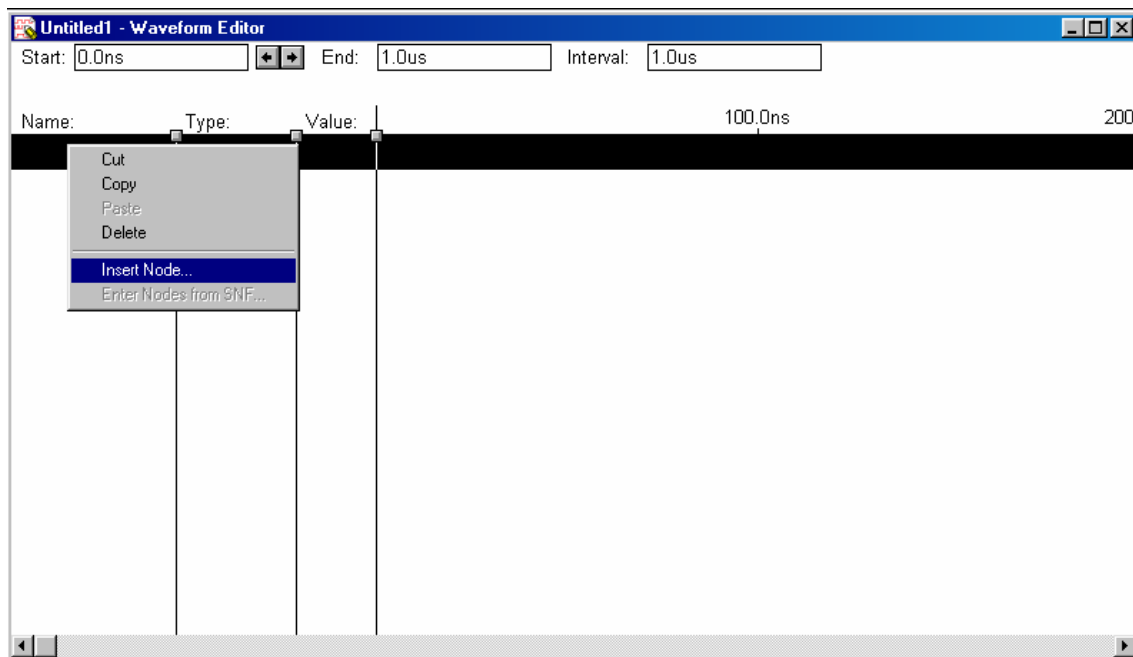


Figura 21: añadiendo señales al sistema

Ahora aparecerá un formulario en donde se debe especificar el nombre que tendrá la señal y sus características (dirección, si es parte de un registro etc). Ver figura 22.

Figura 22: Formulario para especificar las características de la señal que será añadida.

El proceso anterior se repite tantas veces como señales queramos agregar. Una vez terminada la especificación de las señales que serán parte del proyecto se debe proceder a especificar la forma en como estas señales evolucionan a lo largo del tiempo. En la figura 23 se puede apreciar la realización de una compuerta OR.

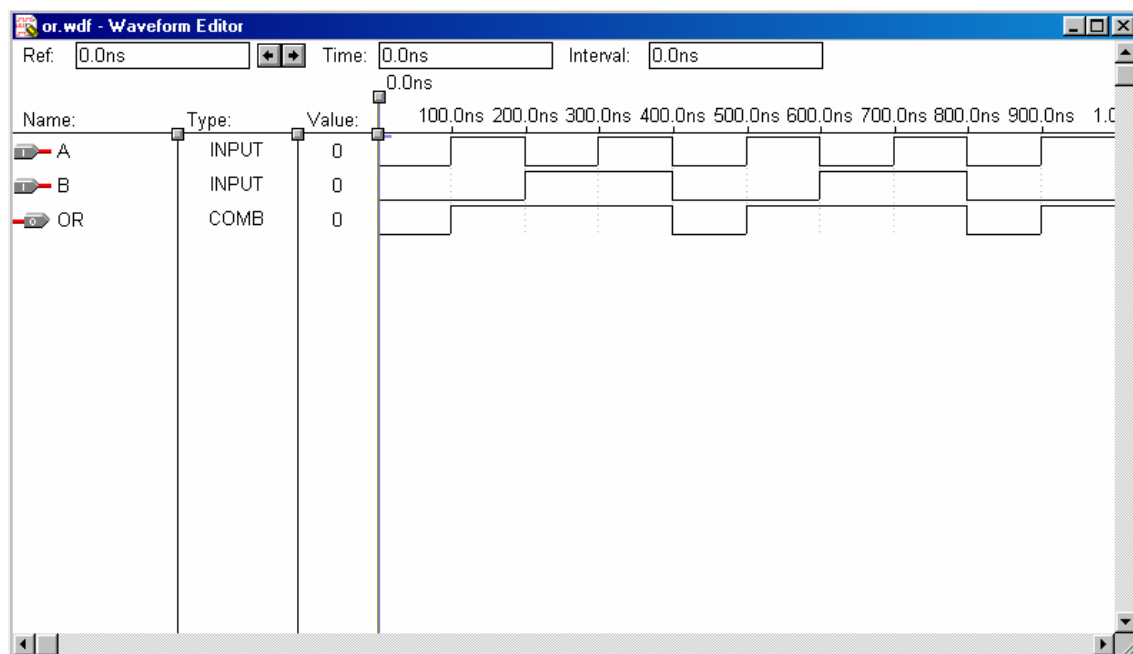


Figura 23: formas de ondas de las entradas y salidas de una compuerta OR.

Es importante destacar que cuando se ocupa esta opción de archivo de entrada, se debe especificar el sistema sin ambigüedades, como así es preciso procurar que la descripción sea consistente. Por ejemplo se podría tener el archivo de entrada que se indica en la figura 24:

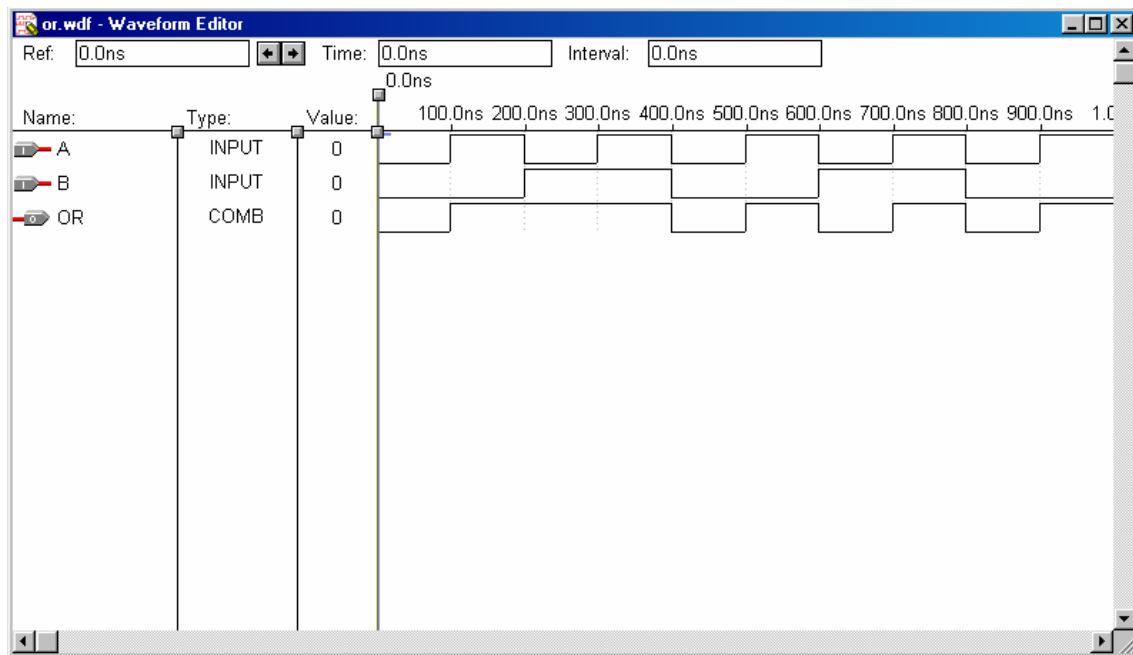


Figura 24: Ejemplo de una especificación imprecisa de un sistema.

Se puede ver en la figura 23 que hay dos posibles valores para la combinación  $A=0$  y  $B=1$ , en el intervalo de 200 a 300 nS el valor de salida es 1, en cambio en el intervalo de 600 a 700 nS el valor es 0. Si se compila con este archivo de entrada el compilador arrojará un mensaje de error como el que se indica en la figura 25.

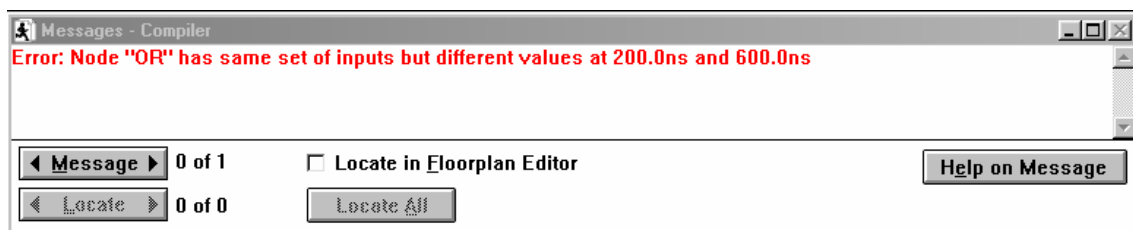


Figura 25: mensaje de error en caso de existir alguna ambigüedad en la descripción del sistema.

## Archivos de trabajo

En la figura 26 se puede apreciar la forma en que están asociados los distintos archivos de trabajo con los procesos que forman parte de la compilación. Es importante tener en cuenta que todos los archivos de trabajo, para un proyecto en particular, están en el mismo directorio (es por eso que resulta aconsejable crear un directorio con el nombre del proyecto, y ejecuta dentro de ese directorio el proceso de compilación).

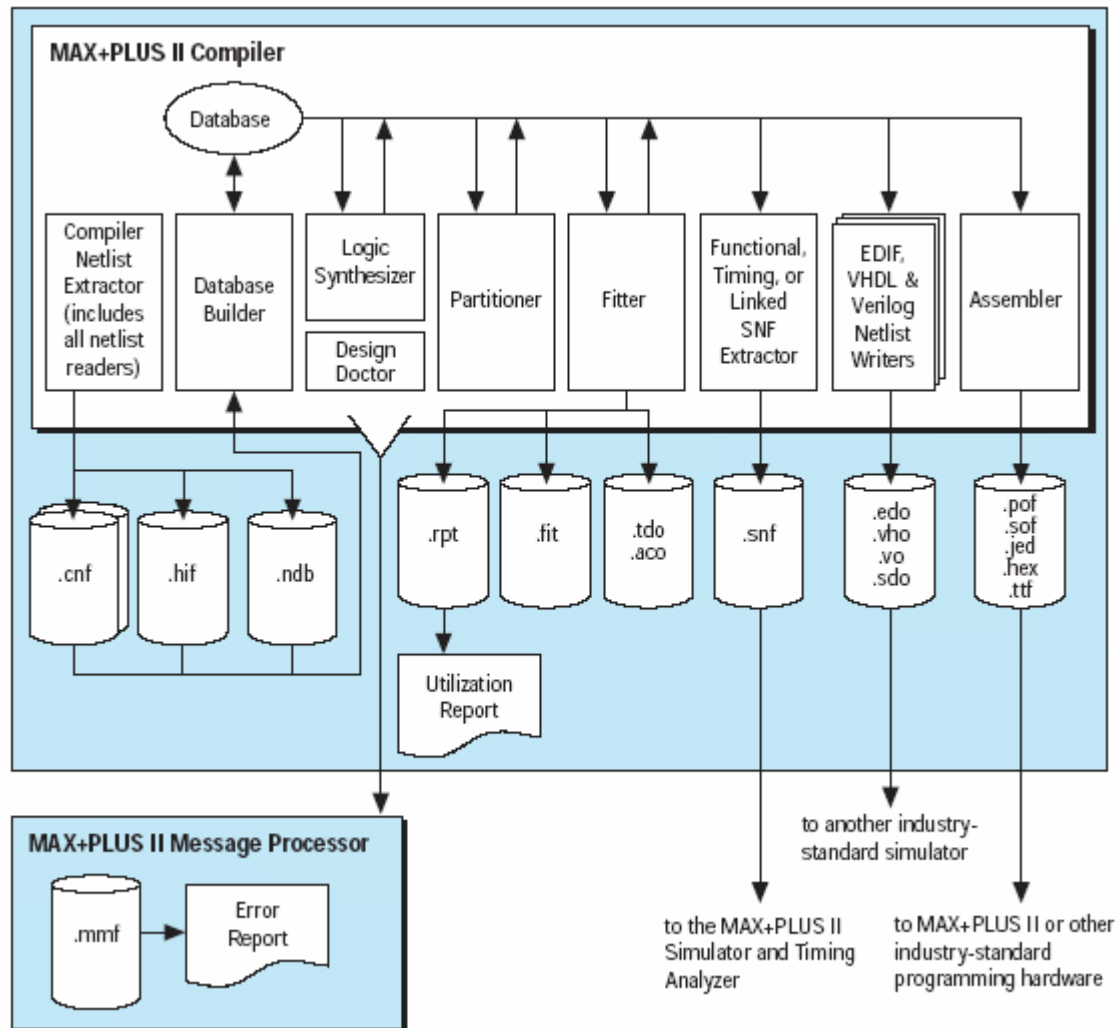


Figura 26: archivos asociados a los distintos procesos que componen el proceso de compilación.