
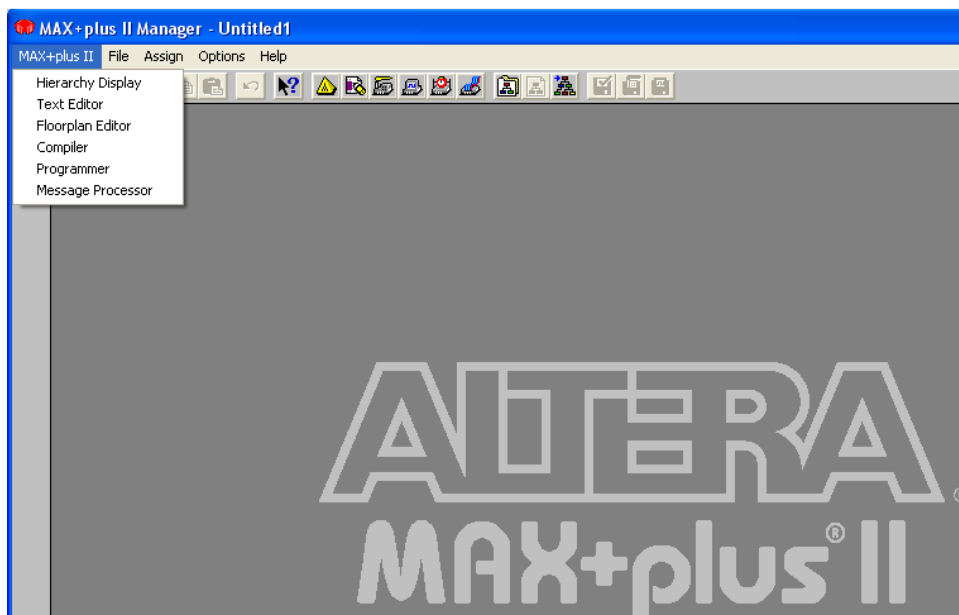


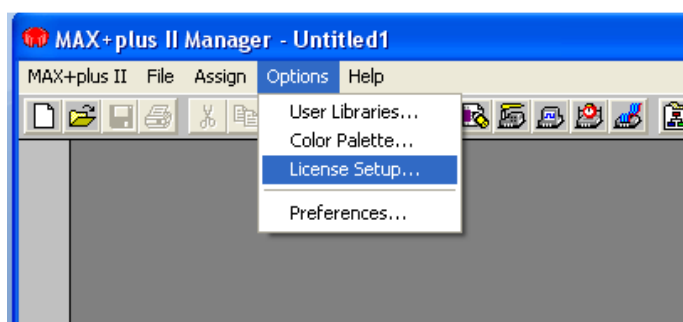
Practica1 de VHDL con altera. FCHE 2010

Primero instala altera:  student102.exe

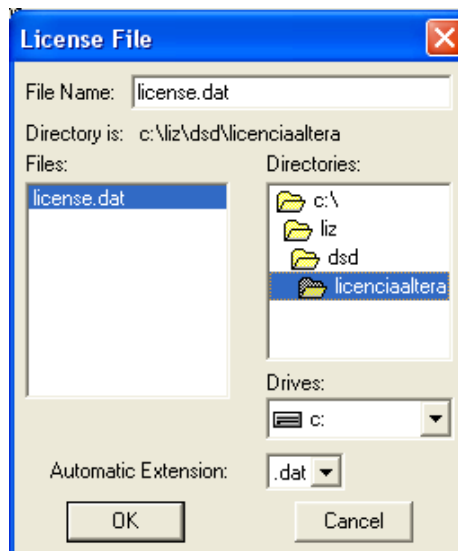
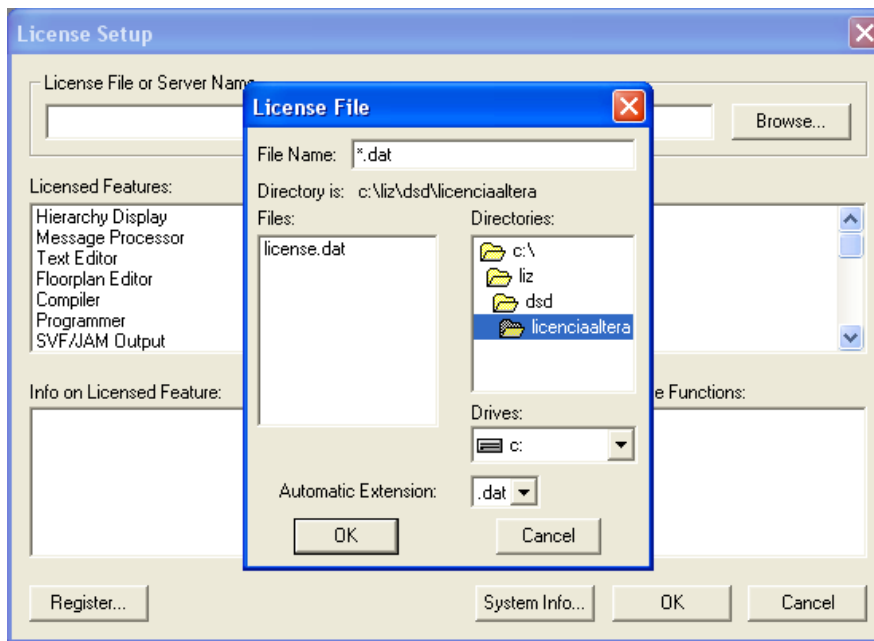
Ya instalado falta avisar donde está la “license.dat”. Si no tiene licencia solo aparecen como se observa solo seis menús, deben aparecer once. En el menú MaxpluxII aparecen.



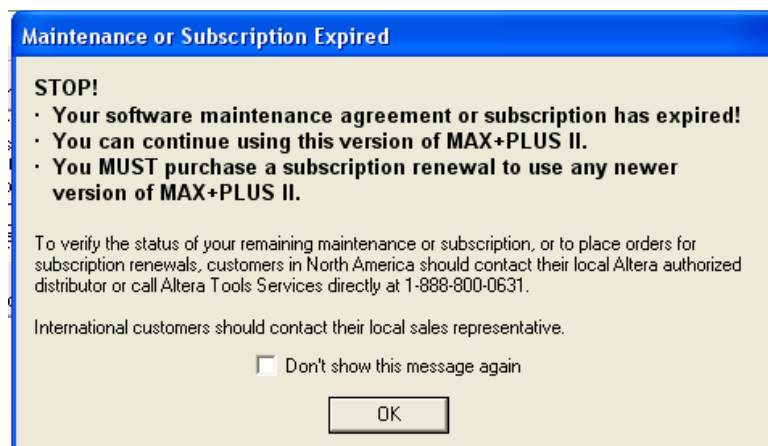
Ya debemos tener la licencia y saber donde está. Ahora le diremos al programa donde está.



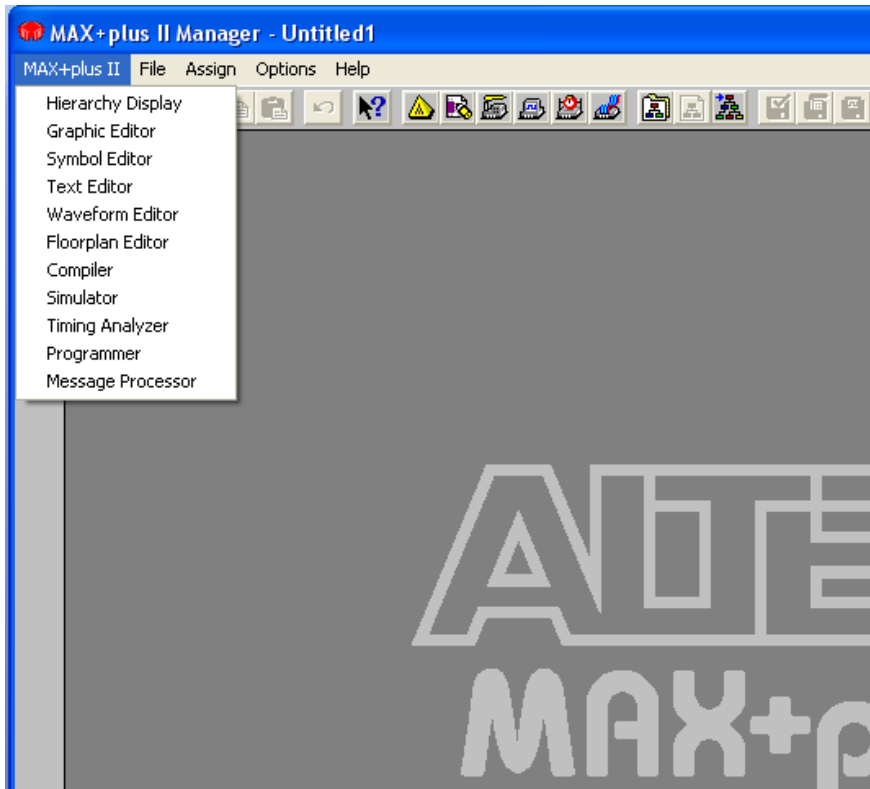
Después de entrar al menú option y seleccionar el submenú “licence Setup”, aparece una caja de diálogo, donde seleccionamos el primer botón “Browse”, para buscar en la pc el archivo.



Si aparece a tu izquierda el archivo “license.dat”, lo seleccionamos y apretamos OK. Y ya esta listo. Si te aparece la caja de dialogo siguiente, solo aprieta OK.



Ahora chequeemos en el menú de max+plusII si aparecen los once menus, si es asi estamos listos para trabajar.

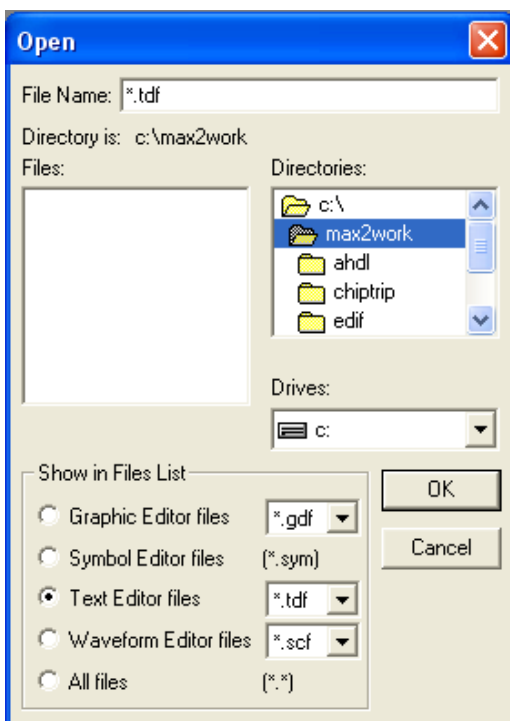


Listo para trabajar (con once menú vistos). La secuencia a seguir es:

1. Abrir ó crear nuevo documento(escribiendo el código) VHDL
2. Guardar proyecto y SELECCIONAR que se utiliza actualmente(muy importante)
3. Compilar (se verifica sintaxis)
4. Se asignan los valores numéricos de entrada de nuestra tabla de verdad(Waveform)
5. Se Simula
6. Se observan los resultados

1) En nuestro caso abriremos documento. El nuestro es clase1.vhd del folder

vhdlclase. Primero véase un ejemplo genérico y al final con lo pedido aquí.



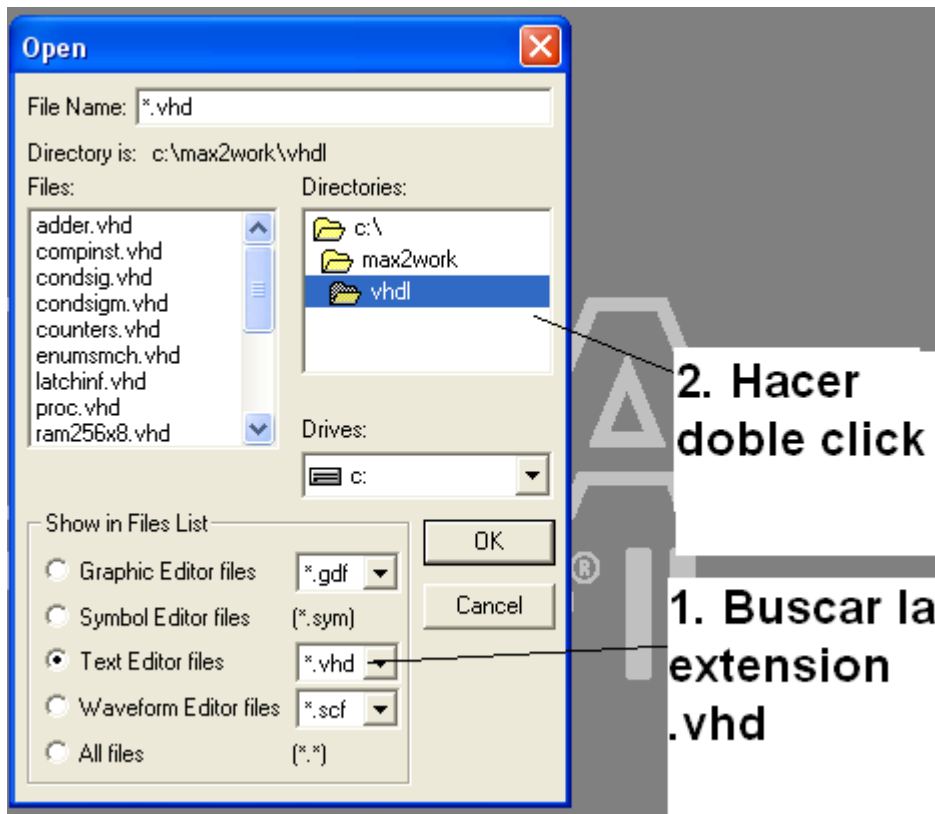
Deberemos colocarnos en el archivo deseado.



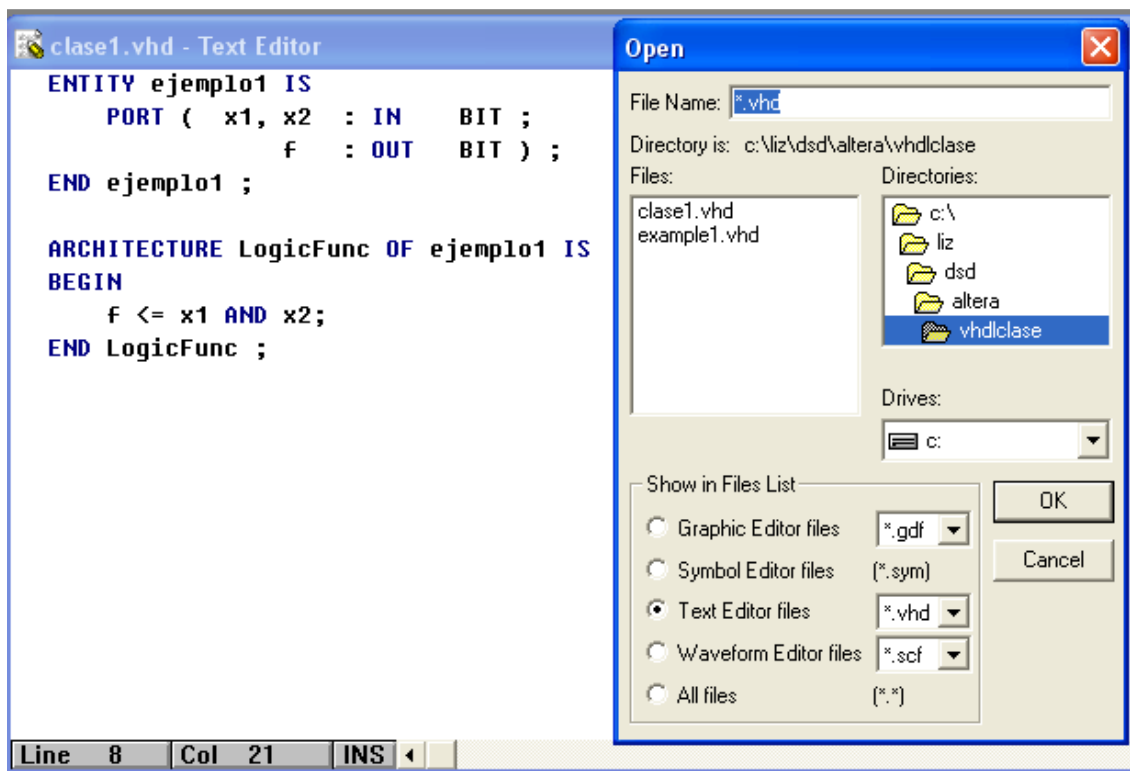
Para el caso de VHDL debemos elegir la opción de Text Editor File cambiando a extensión Vhd



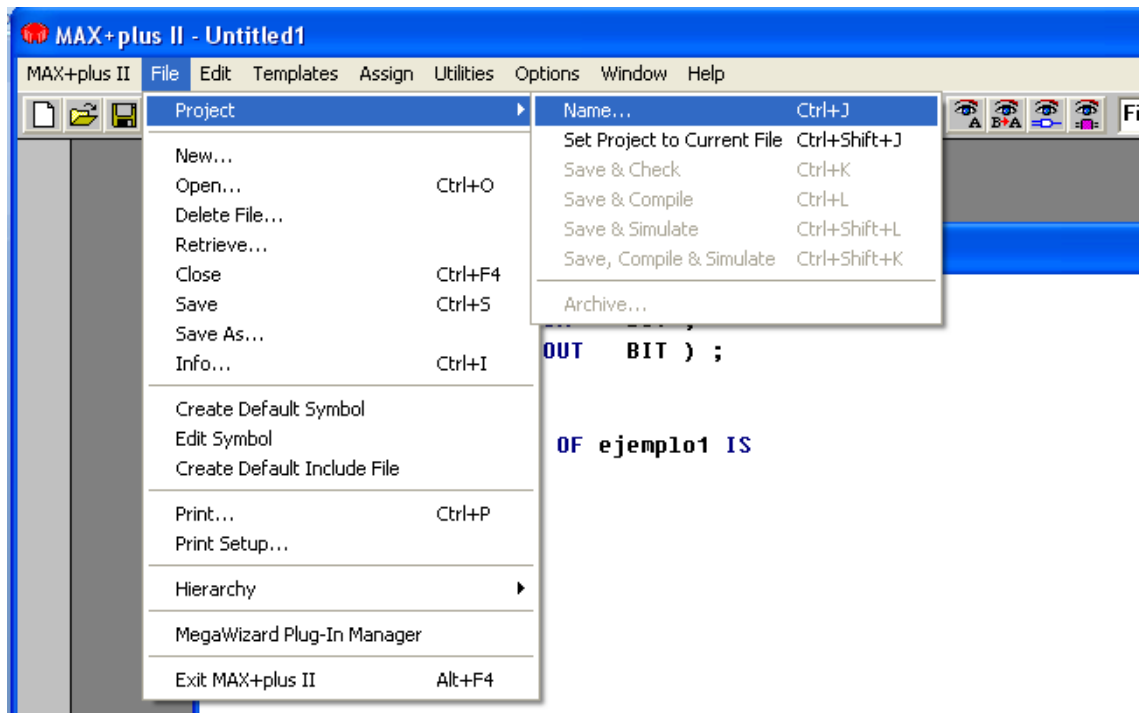
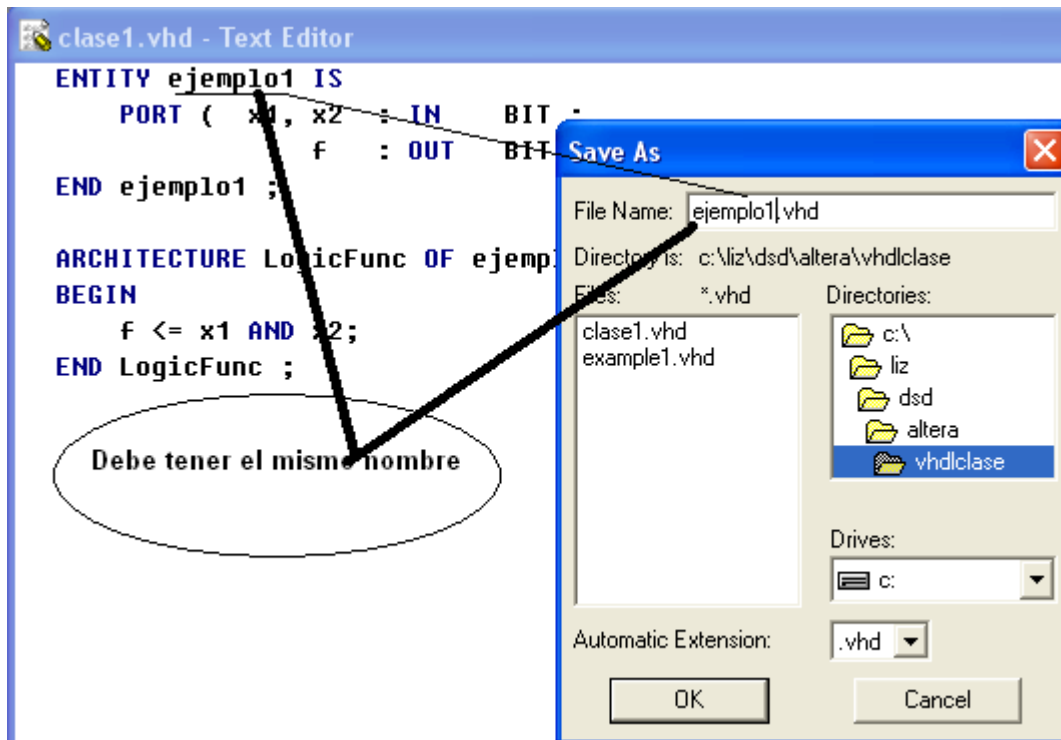
Ya que demos doble click nuevamente a nuestro folder donde están nuestros códigos, a la izquierda aparecen (Files), y le damos click al código deseado.

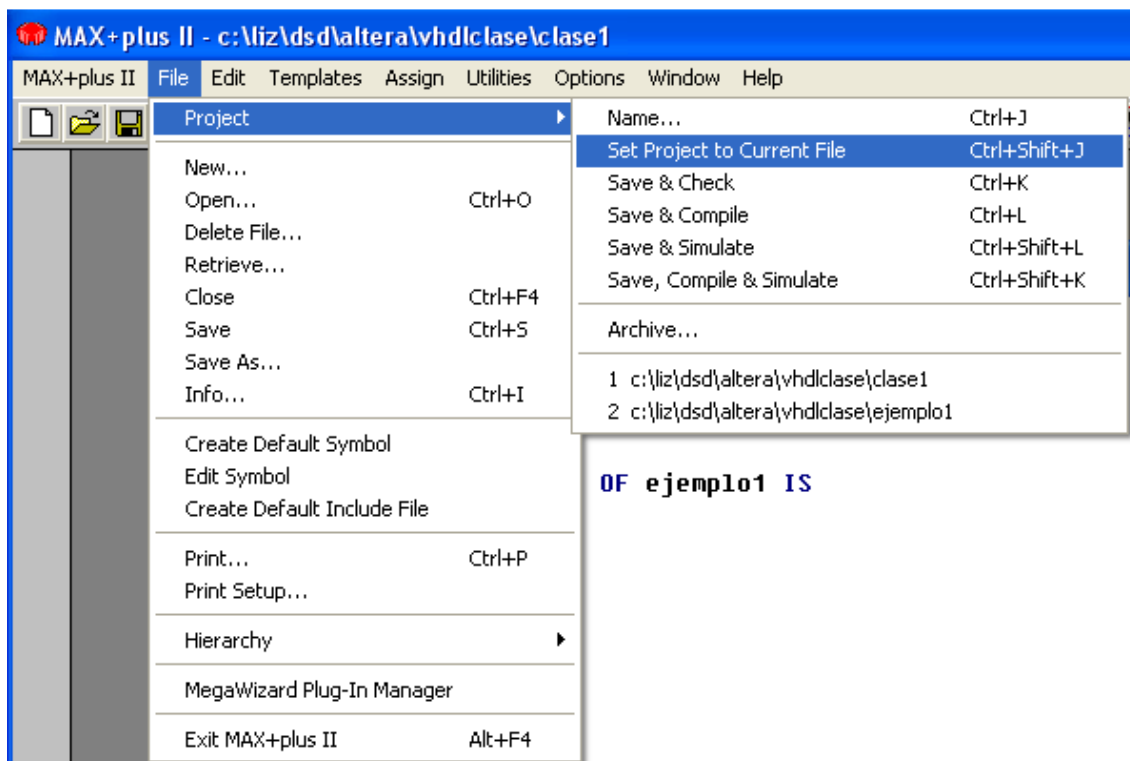
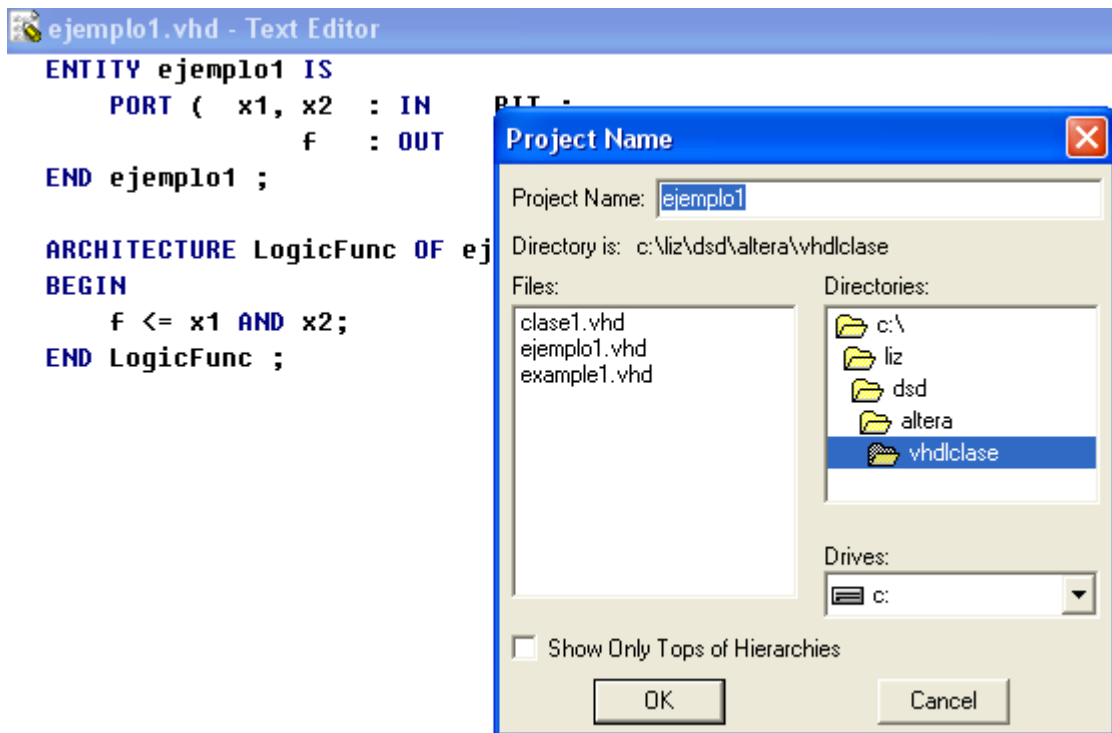


Seleccionar en Files el archivo deseado. El nuestro es clase1.vhd del folder VHDCLASE.



2. Primero guardamos nuestro código en un proyecto, y luego le decimo que es el actual de trabajo.





3. Chequemos que este código sirve para insertar una tabla de verdad básica, para nuestro ejemplo solo es una compuerta AND.

```

clase1.vhd - Text Editor
ENTITY ejemplo1 IS
    PORT ( x1, x2 : IN    BIT ;
          f   : OUT   BIT ) ;
END ejemplo1 ;

ARCHITECTURE LogicFunc OF ejemplo1 IS
BEGIN
    f <= x1 AND x2;
END LogicFunc ;

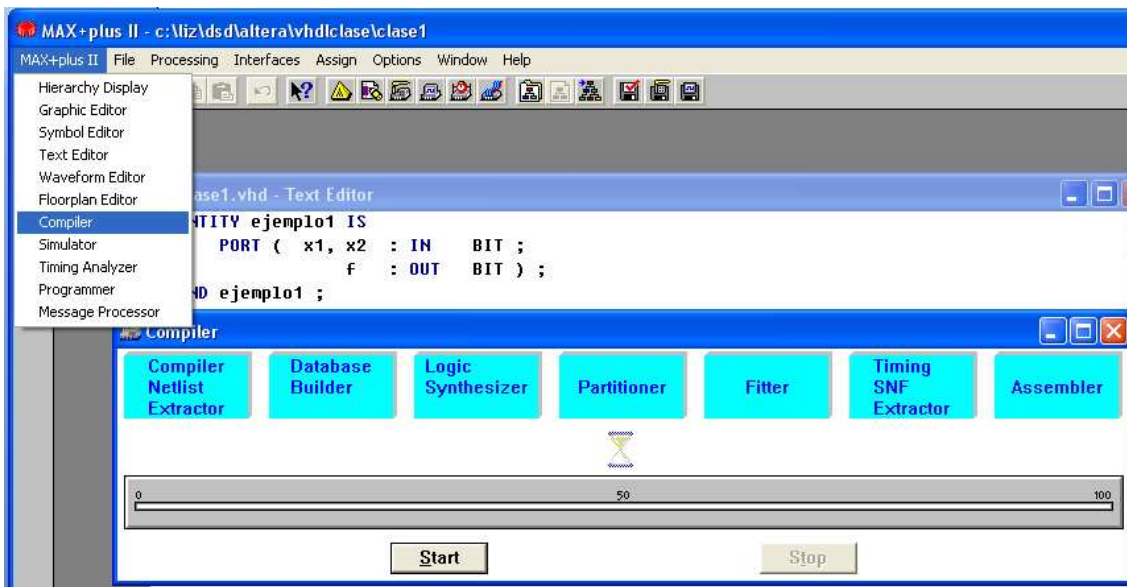
```

AND		
x1	x2	f
0	0	0
0	1	0
1	0	0
1	1	1

Nota: Falta asignar los valores de x1 y x2 para obtener NUMERICAMENTE el resultado f. Solo tenemos las variables y su comportamiento.

4. Compilemos para verificar sintaxis.

Nota que si el botón Start está apagado es porque no salvaste ni asignaste como actual.

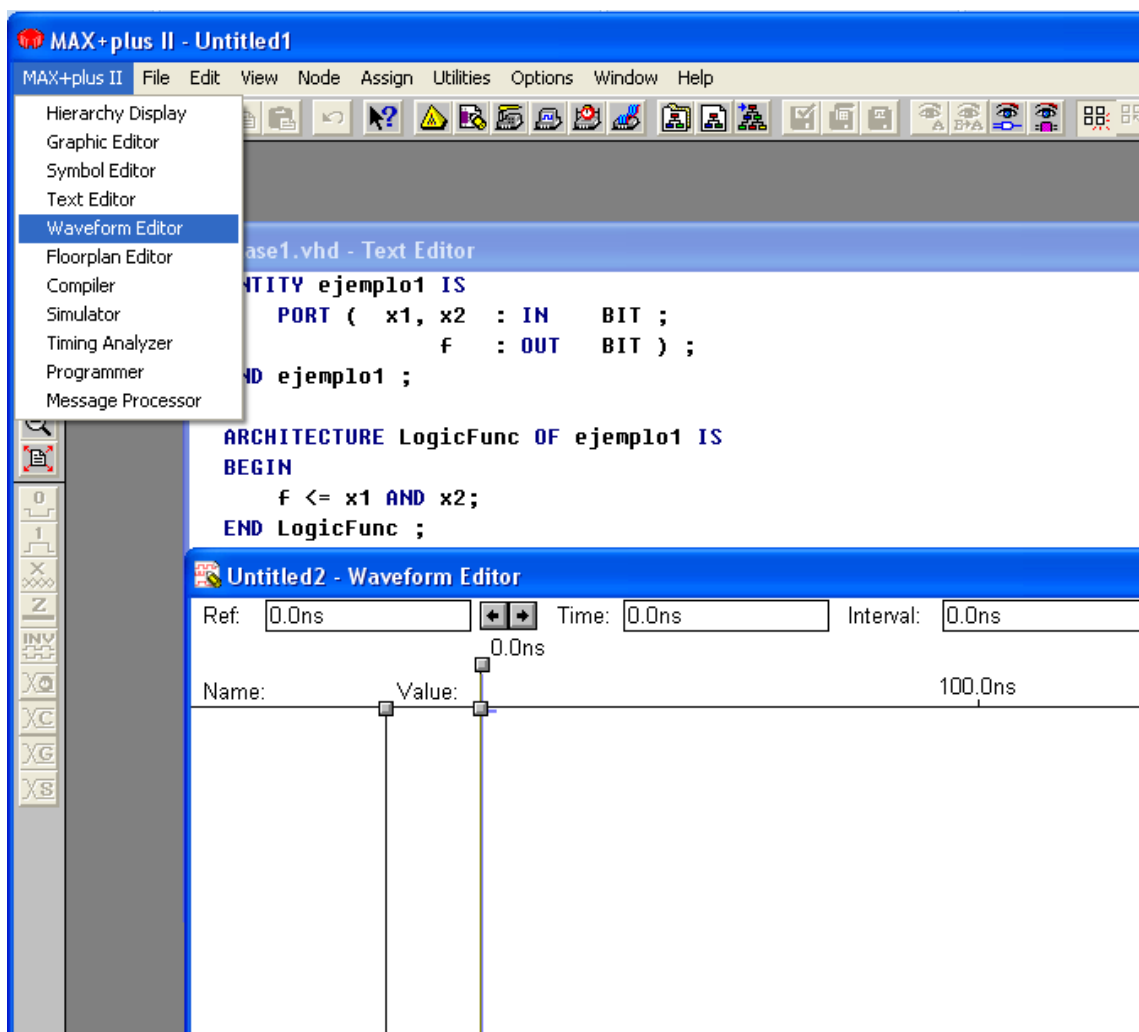


Apretamos Start

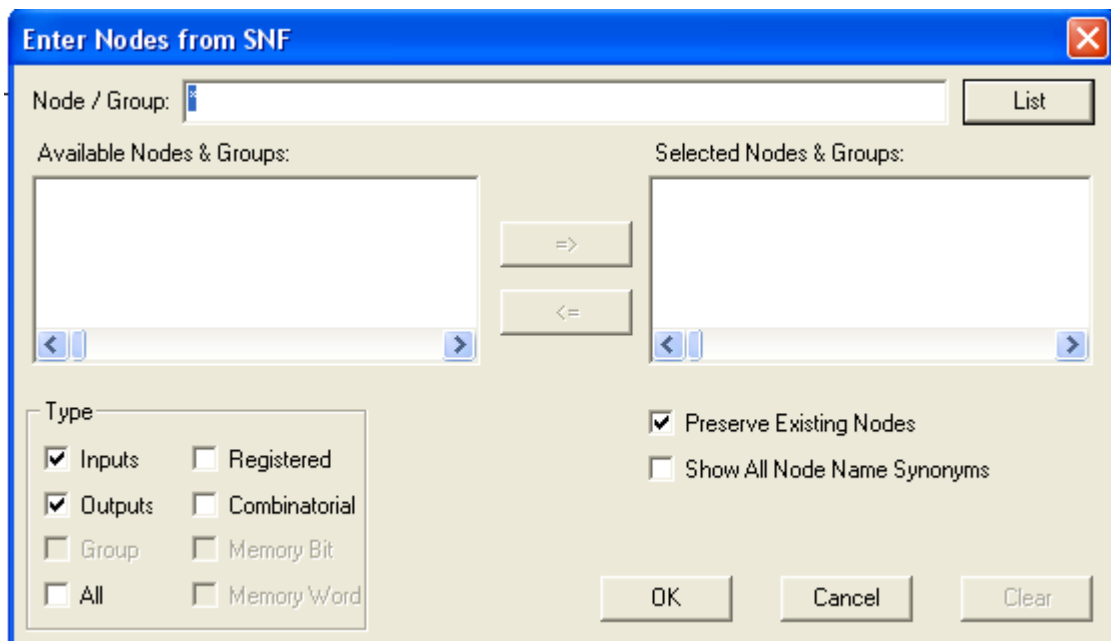
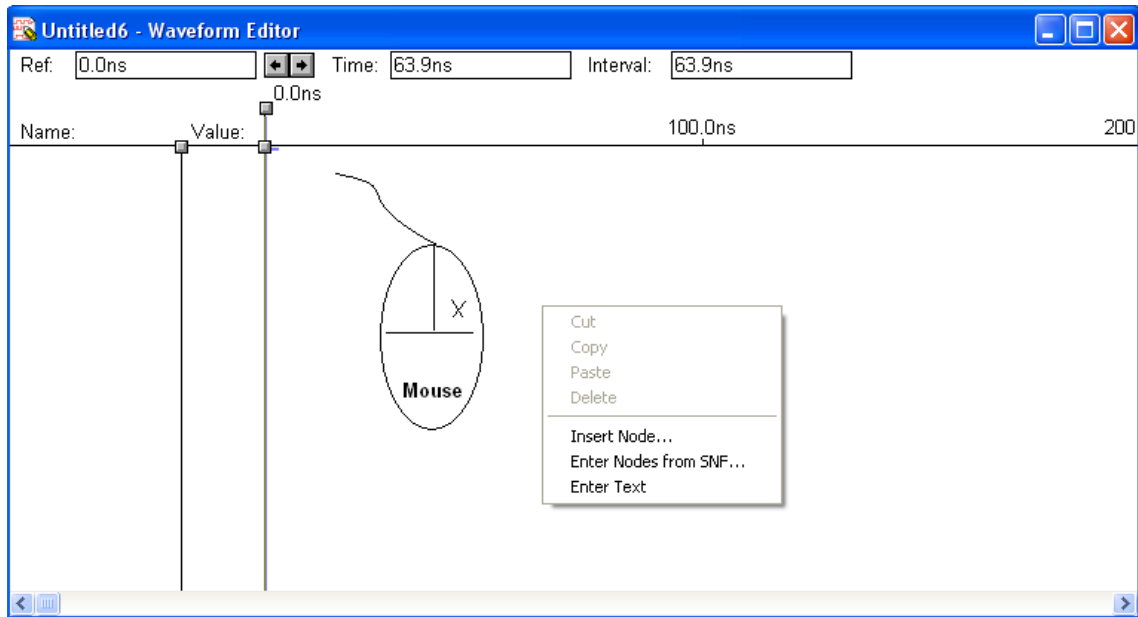


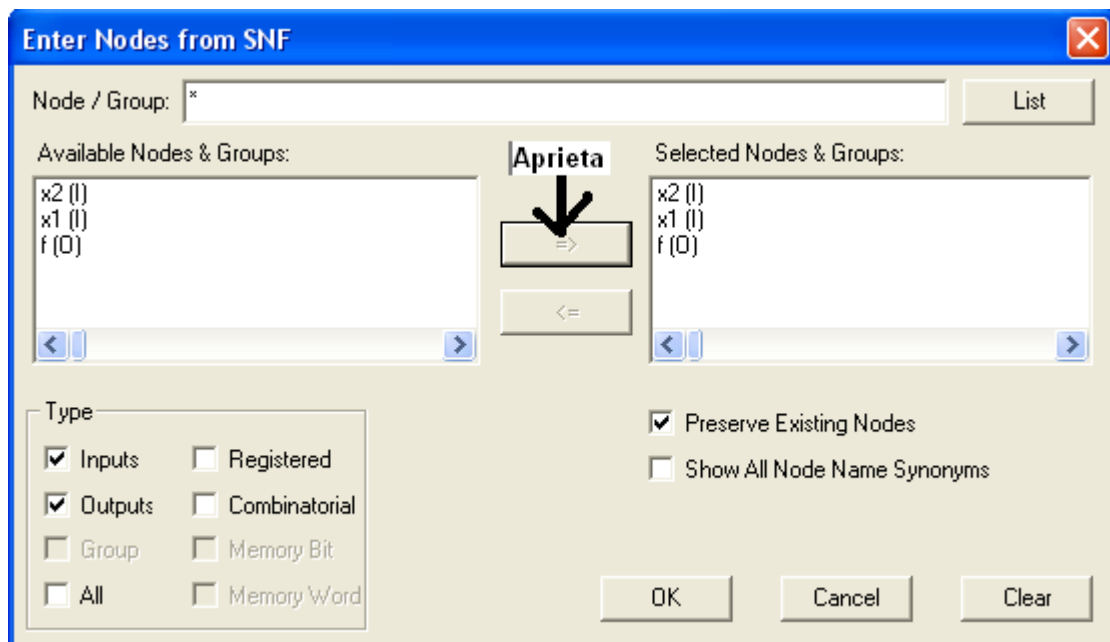
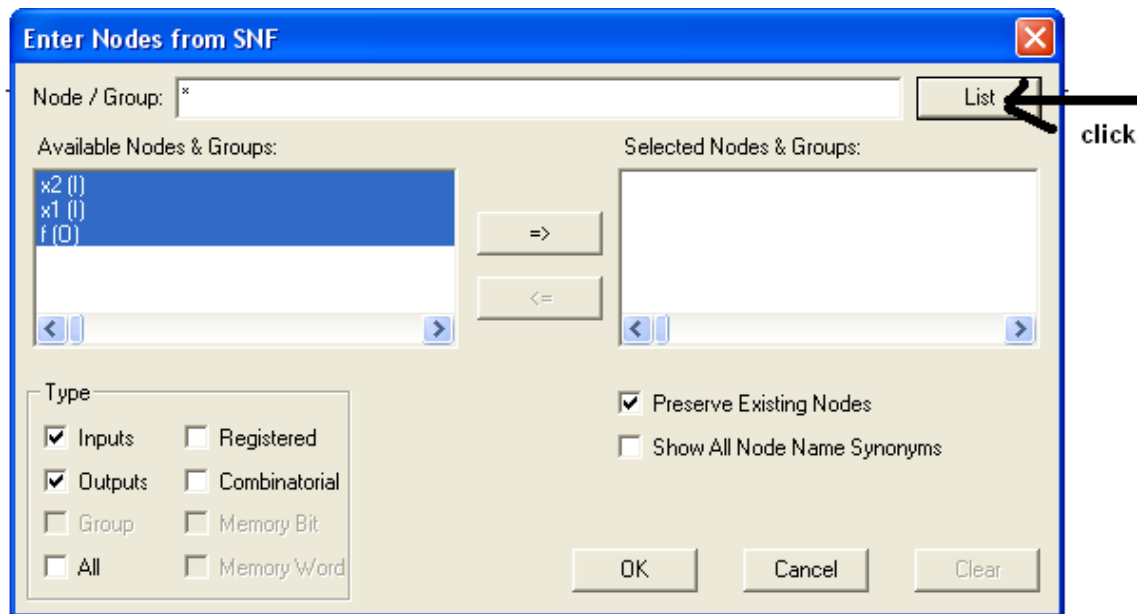
Si ya no tenemos errores podemos continuar, sino regresar a checar lo que te piden, puede estar equivocado el nombre, variables que no concuerdan en entity y architecture.

5. Ahora, asignamos valores a las variables de entrada. Para nuestro caso x1 y x2.

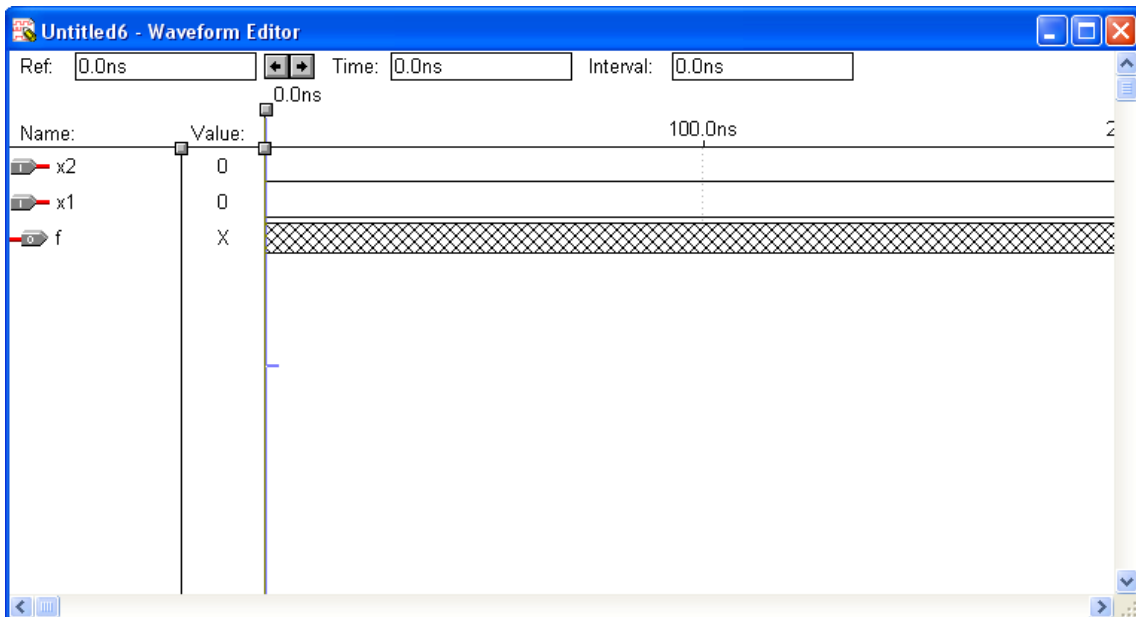


Ponemos el mouse al centro como se ve en la siguiente figura y se aprieta botón derecho, aparece menú dado. Debemos seleccionar ...SNF, sino aparece no compilaste bien.



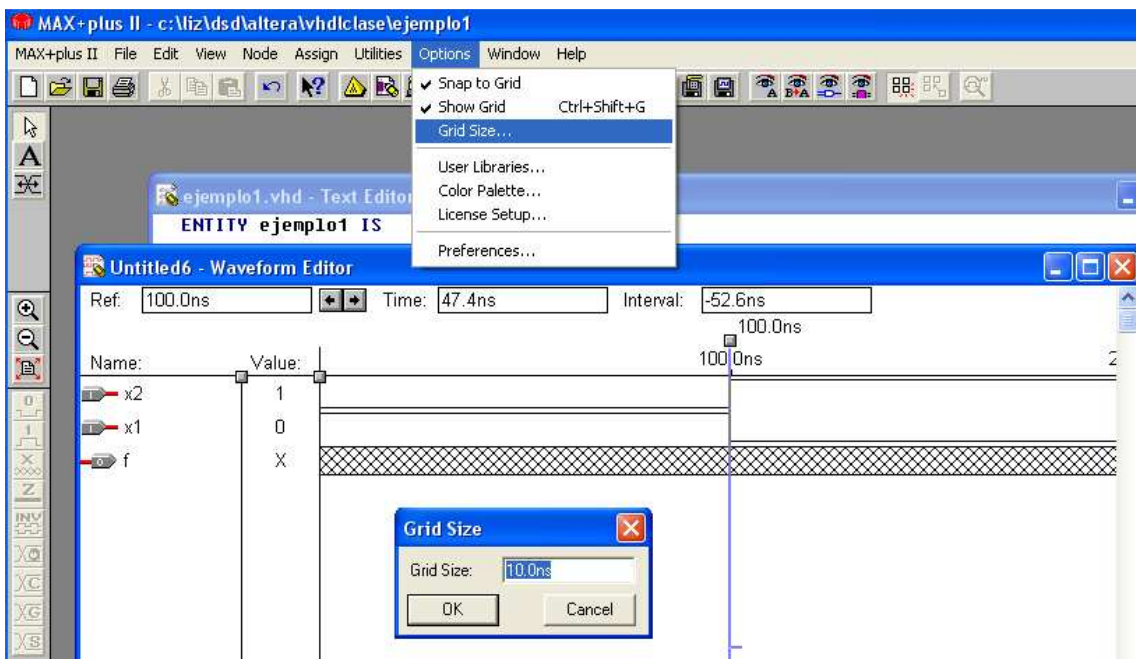


Después aprieta OK

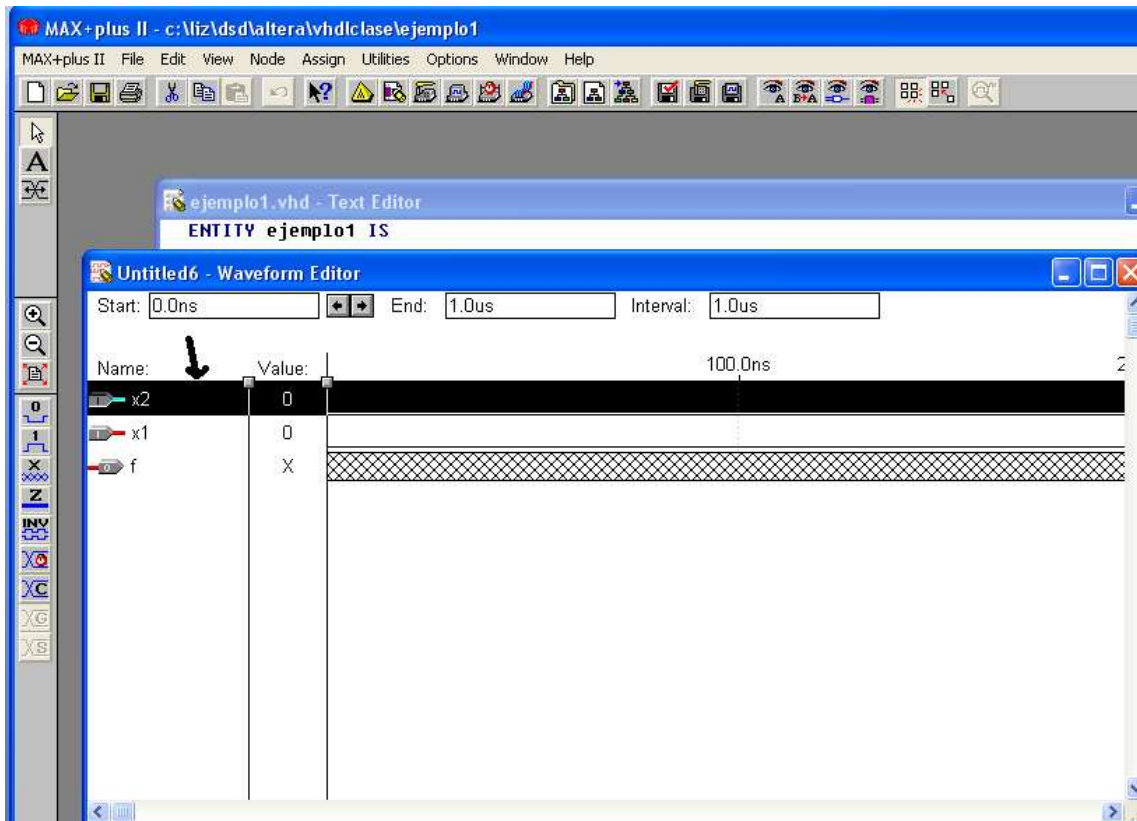


Ya se insertaron nuestras variables de entrada y de salida. PERO falta asignarle valores numéricos.

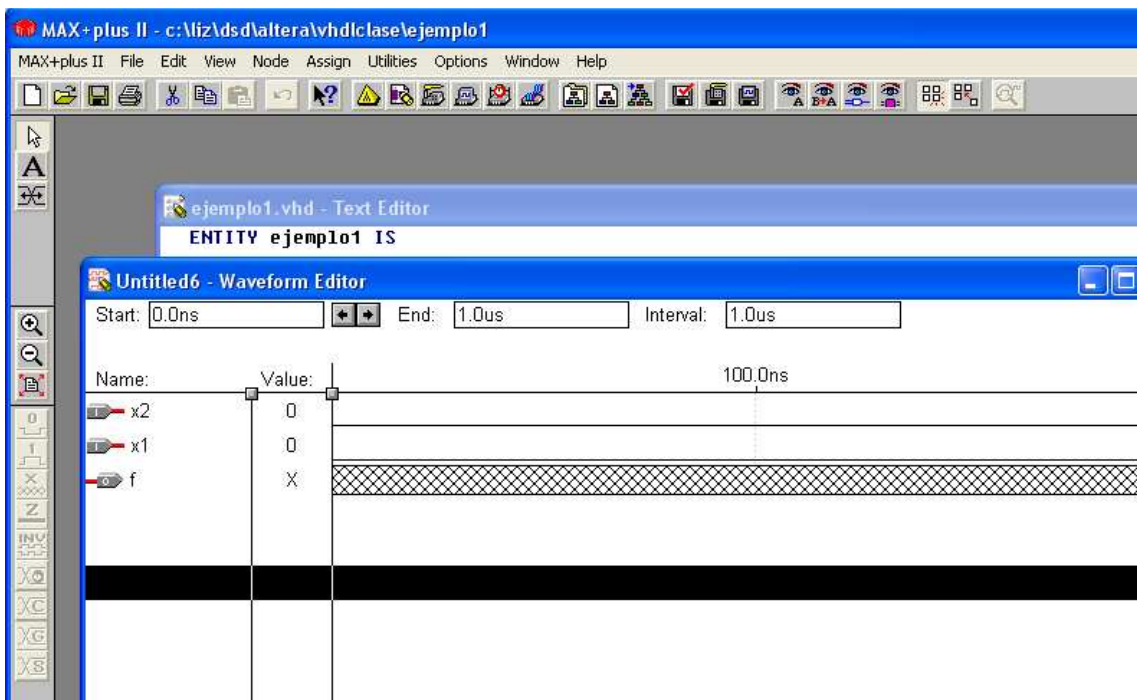
Primero arreglemos como lo queremos ver , con la talla de duración de la señal.



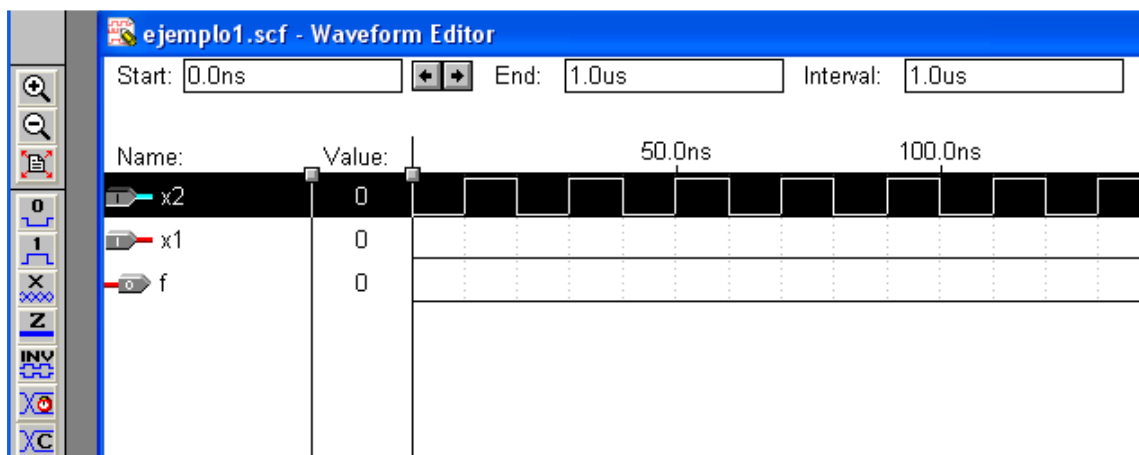
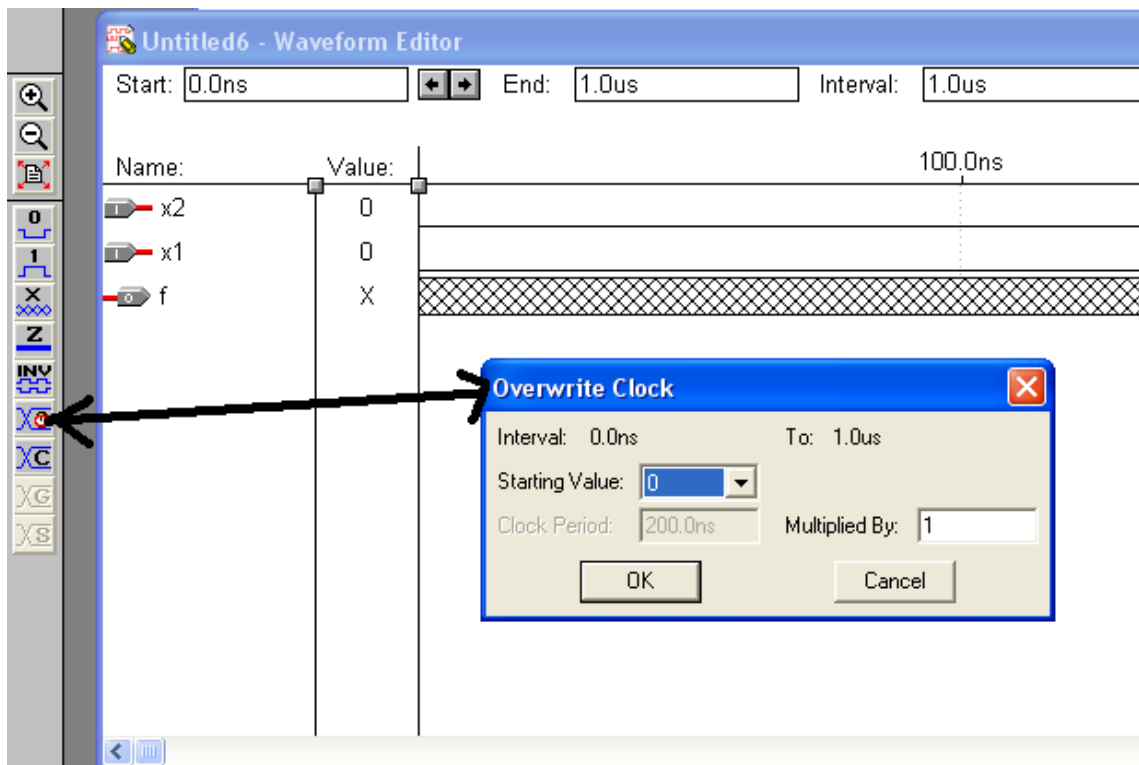
Selecciona la variable x2, observa como se activan los botones de la izquierda.



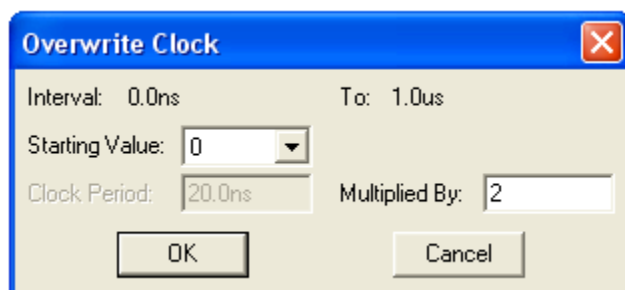
Si seleccionaras otra opción se desactivan los botones, compruébalo.

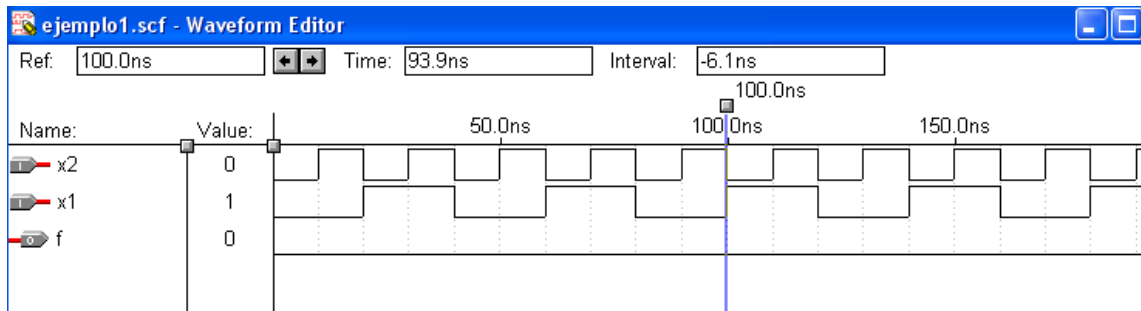


Entonces volvemos a seleccionar X2 y escogemos el relojito rojo. Le damos OK

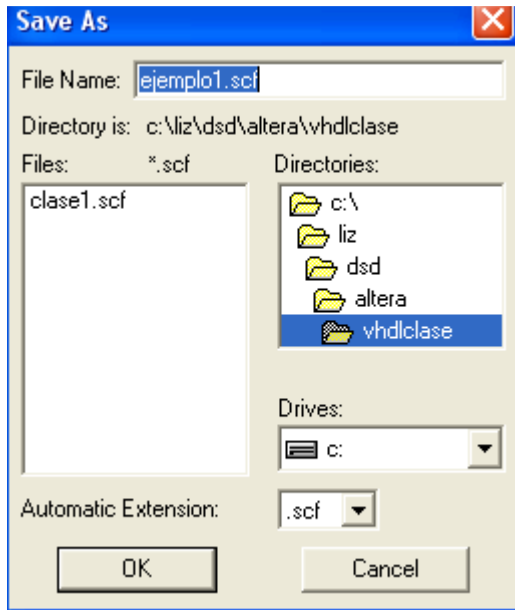


Seleccionamos x1, el relojito y hacemos un cambio, que comience también en cero pero que se multiplique por dos.

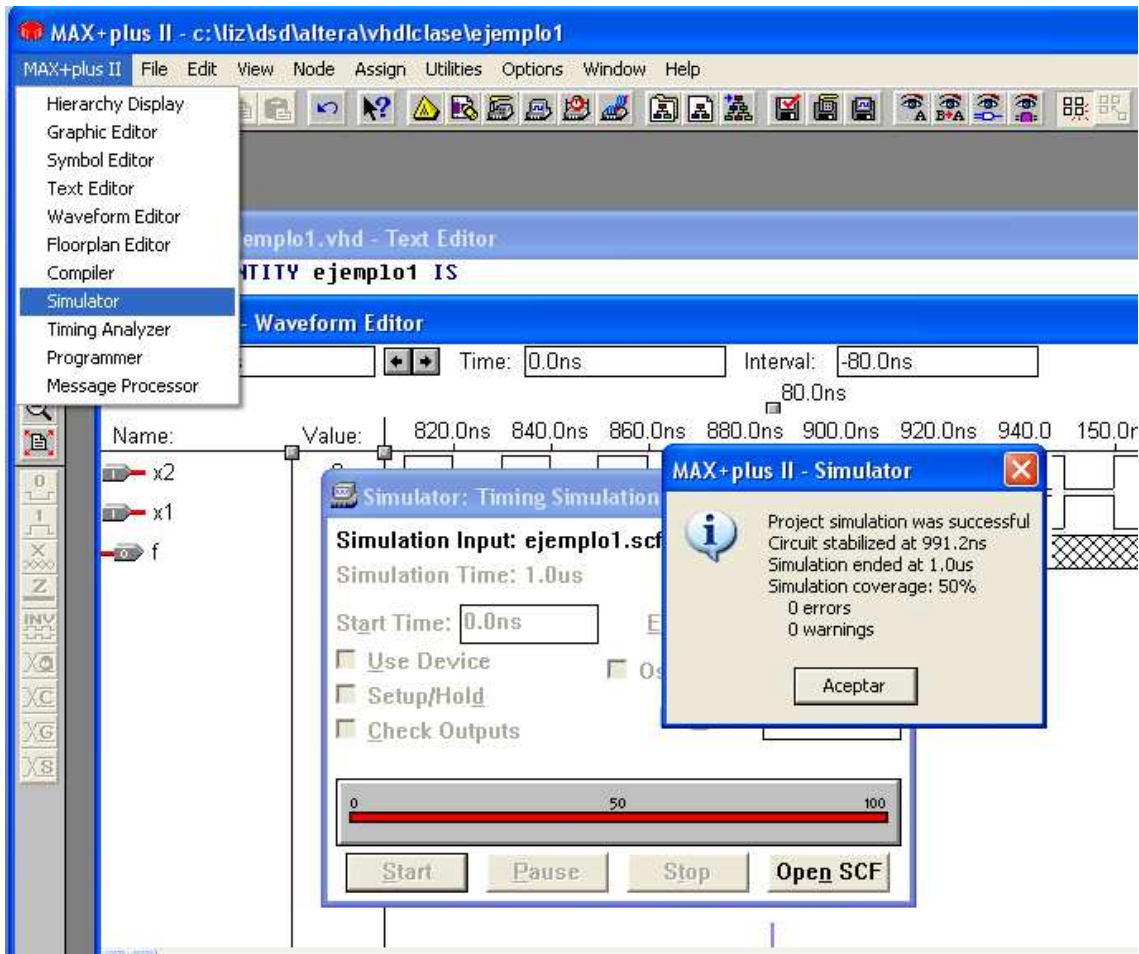




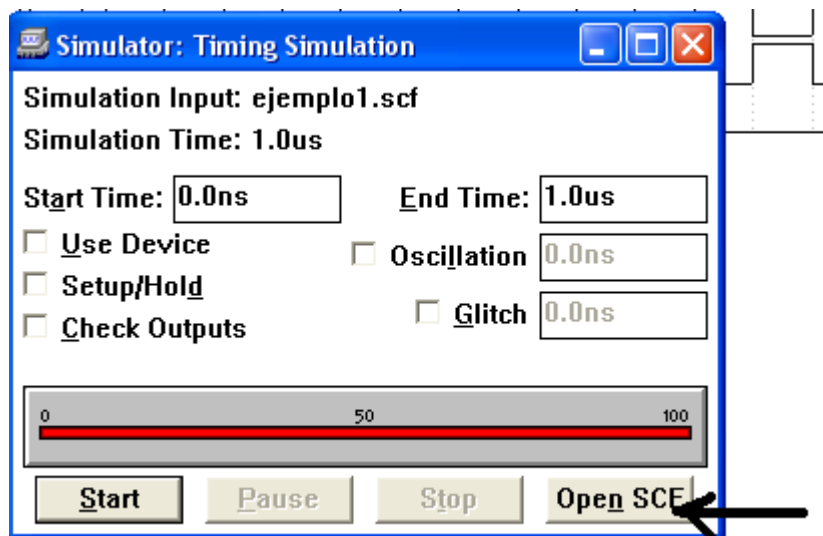
MUY importante, guardar por default. OK.



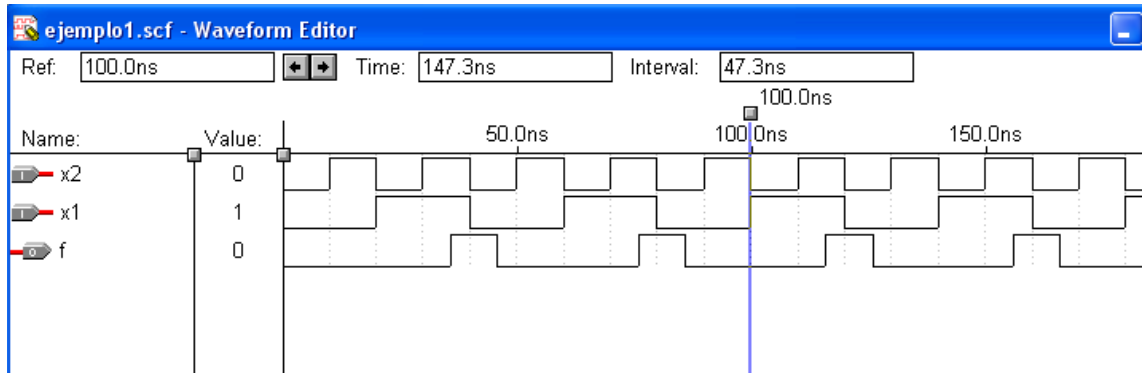
5. Listos para simular.



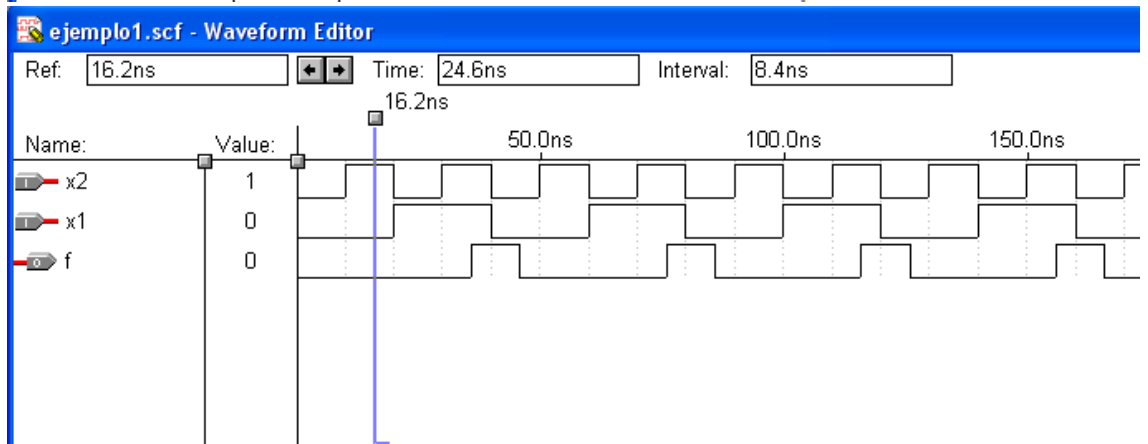
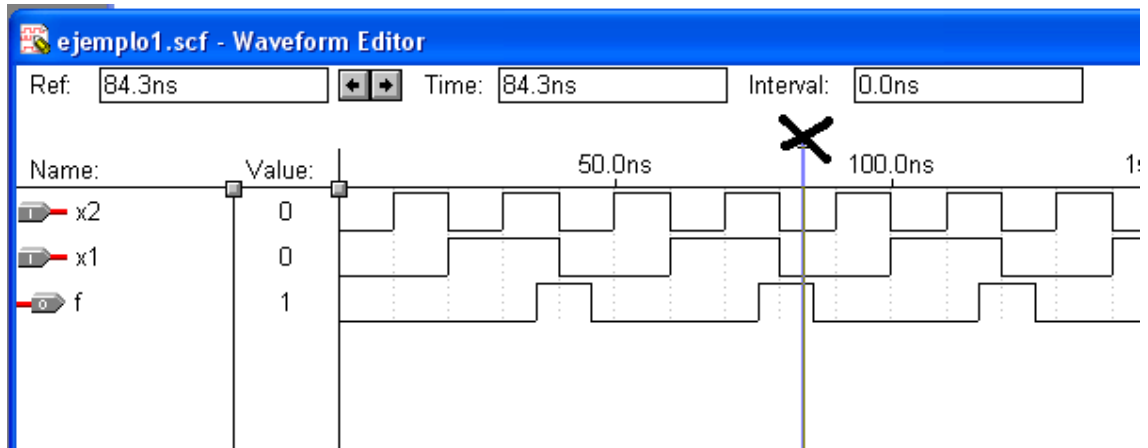
7. Observar los resultados de la simulación



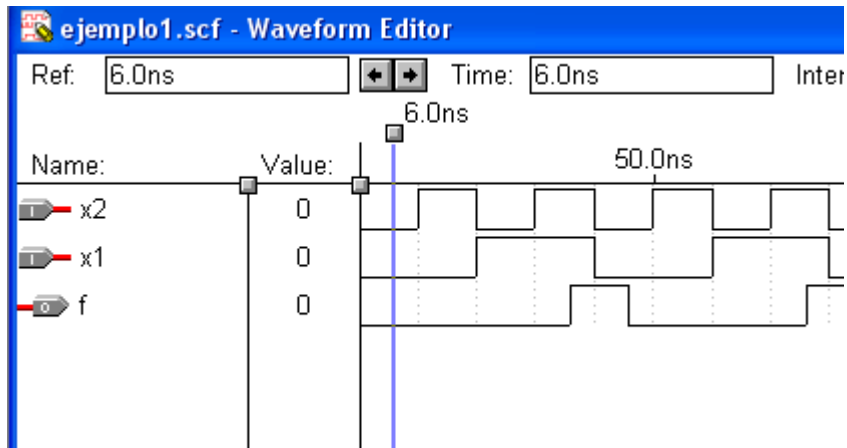
Y entonces tenemos nuestro resultado.



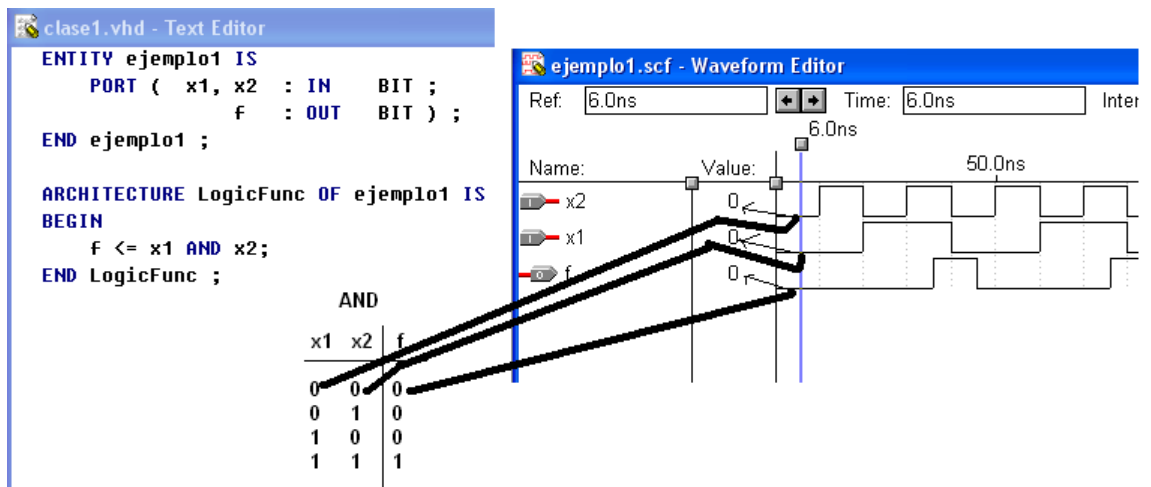
Interpretemos. Moviendo la regleta vertical azul, con el mouse (sostener deslizar).



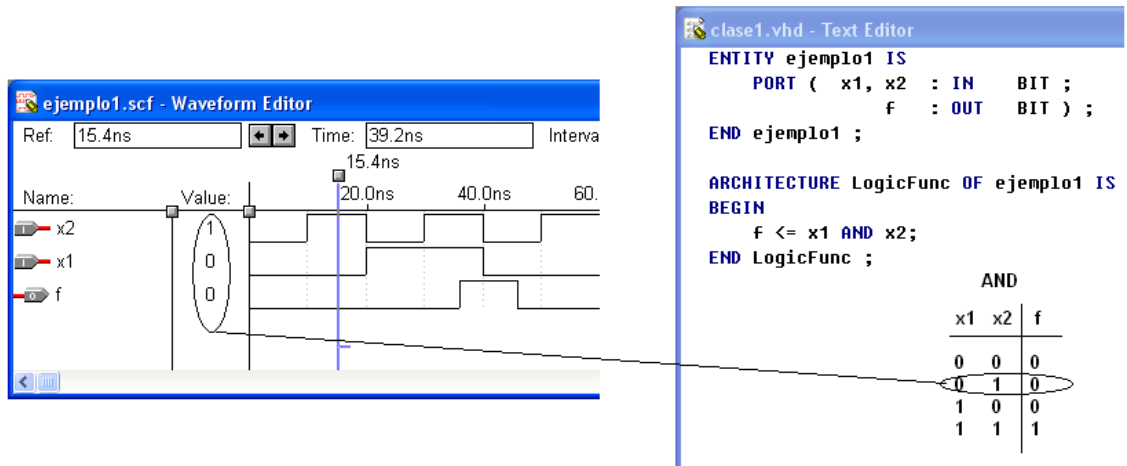
Cada vez que movemos la regleta los valores de bit van cambiando. Coloquemonos en la celda inicial.

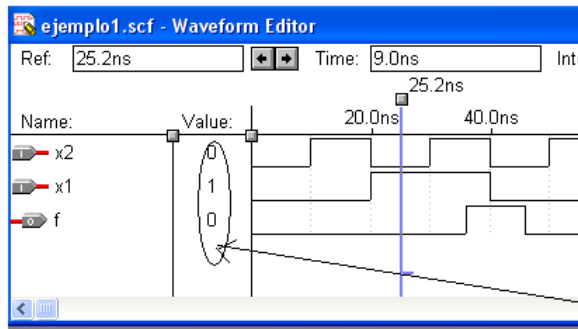


Relacionemos la teoría con la programación y este resultado.



Ahora observemos para los siguientes ciclos de reloj o valores de la tabla.





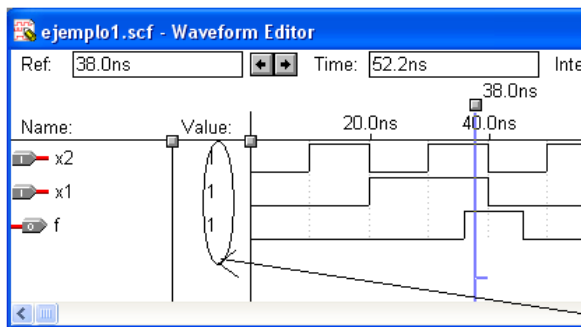
```

class1.vhd - Text Editor
ENTITY ejemplo1 IS
  PORT ( x1, x2 : IN  BIT ;
        f : OUT  BIT ) ;
END ejemplo1 ;

ARCHITECTURE LogicFunc OF ejemplo1 IS
BEGIN
  f <= x1 AND x2;
END LogicFunc ;

```

AND		
x1	x2	f
0	0	0
0	1	0
1	0	0
1	1	1



```

class1.vhd - Text Editor
ENTITY ejemplo1 IS
  PORT ( x1, x2 : IN  BIT ;
        f : OUT  BIT ) ;
END ejemplo1 ;

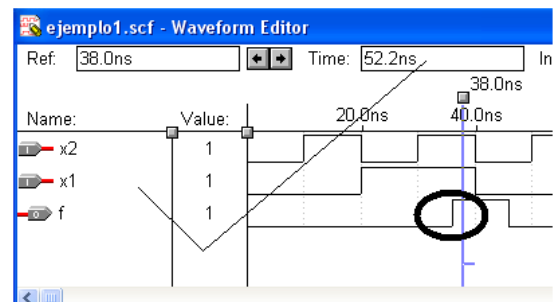
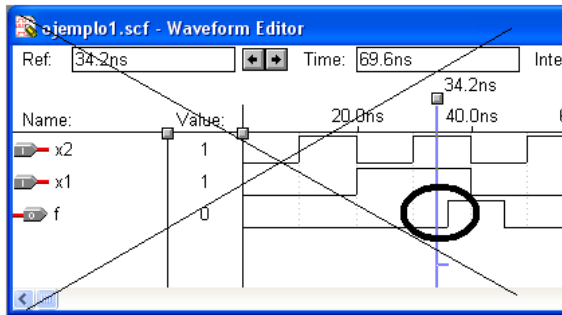
ARCHITECTURE LogicFunc OF ejemplo1 IS
BEGIN
  f <= x1 AND x2;
END LogicFunc ;

```

AND		
x1	x2	f
0	0	0
0	1	0
1	0	0
1	1	1

Nota que en este último ciclo de reloj, la regleta se colocó hasta donde se realizó la transición. Porque se presenta el efecto real, que cuando pasa la energía por el circuito, lo hace con un tiempo de retardo. Por cierto los nanosegundos de retardo, nosotros no lo notamos cuando el circuito es pequeño o su aplicación no requiere un tiempo tan exacto.

Con esto se coloca visualmente la regleta con error y sin error de interpretación.



Ahora para terminar, modifiquemos nuestro ejemplo, en vez de un AND utilicemos un OR, compilemos, adicionamos valores de entrada en la forma de onda (serán los mismos), simulamos e interpretamos con respecto a nuestra tabla de verdad de OR.

Primero hazlo y después checa mis resultados.

```
ejemplo1.vhd - Text Editor
ENTITY ejemplo1 IS
    PORT ( x1, x2 : IN    BIT ;
          f   : OUT    BIT ) ;
END ejemplo1 ;

ARCHITECTURE LogicFunc OF ejemplo1 IS
BEGIN
    f <= x1 OR x2;
END LogicFunc ;
```

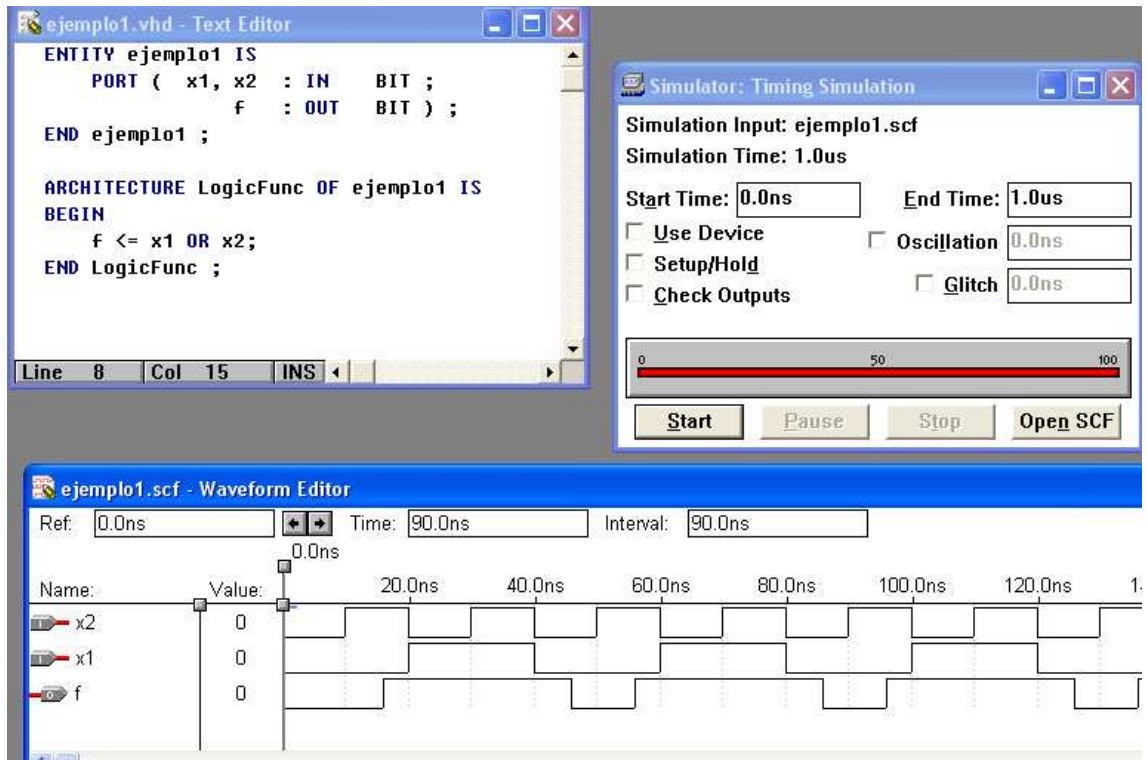
Line 8 Col 15 INS

guardo.

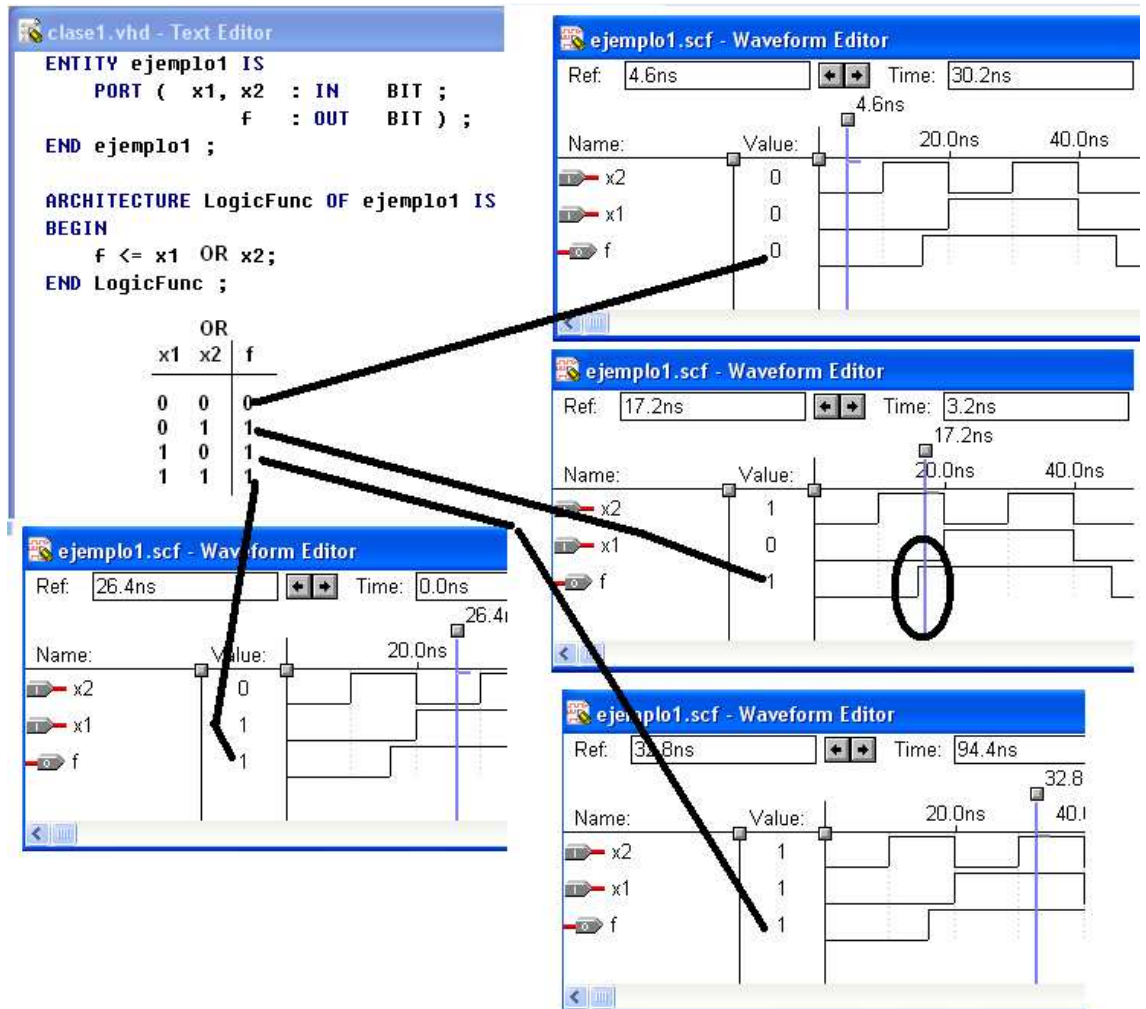
Compilo.

The screenshot shows the MAX+plus II Compiler interface. At the top, there are buttons for various compilation steps: Compiler Netlist Extractor, Database Builder, Logic Synthesizer, Partitioner, Fitter, Timing SNF Extractor, and Assembler. Below these is a progress bar with a red line indicating progress from 0 to 100. There are 'Start' and 'Stop' buttons. Below the progress bar is a 'Messages - Compiler' window showing two informational messages: 'Info: Selecting a device from 'MAX7000' family for AUTO device 'ejemplo1'' and 'Info: Chip 'ejemplo1' successfully fit into AUTO device 'EPM7032LC44-6''. At the bottom, there is a 'MAX+plus II - Compiler' dialog box with an information icon and the text 'Project compilation was successful', '0 errors', and '0 warnings'. There is an 'Aceptar' button in the dialog box.

Simulo y deajo los valores de entrada en la forma de onda igual.(se puede hacer otra forma de onda independiente)



Relaciono datos

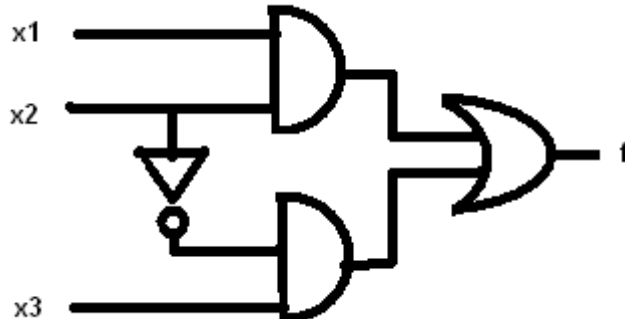


FIN

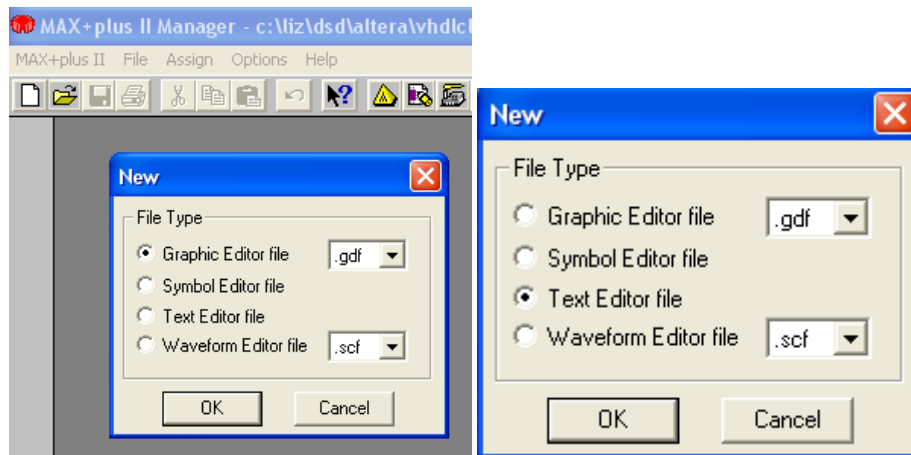
PRACTICA 2. VHDL CON ALTERA

Es posible realizar un bloque más grande con varias compuertas.

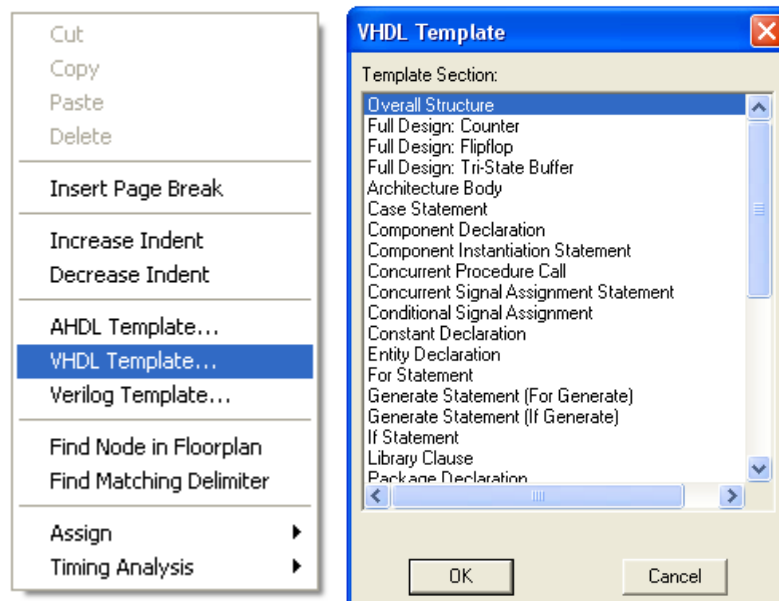
Cuando ya simplificaste tu tabla de verdad y te quedo así, entonces lo pasas a vhdL.



Creemos ahora uno nuevo. Seleccionemos Text Editor



Mouse derecho escogemos "template VHDL". Escogamos Entity declaration



Luego architecture body

```
Untitled3 - Text Editor
ENTITY __entity_name IS
  GENERIC(__parameter_name : string := __default_value;
          __parameter_name : integer:= __default_value);
  PORT(
    __input_name, __input_name      : IN   STD_LOGIC;
    __input_vector_name             : IN   STD_LOGIC_VECTOR(__high downto __low);
    __bidir_name, __bidir_name      : INOUT STD_LOGIC;
    __output_name, __output_name    : OUT  STD_LOGIC);
END __entity_name;

ARCHITECTURE a OF __entity_name IS
  SIGNAL __signal_name : STD_LOGIC;
  SIGNAL __signal_name : STD_LOGIC;
BEGIN
  -- Process Statement

  -- Concurrent Procedure Call

  -- Concurrent Signal Assignment

  -- Conditional Signal Assignment

  -- Selected Signal Assignment

  -- Component Instantiation Statement

  -- Generate Statement
END a;
```

VHDL Template

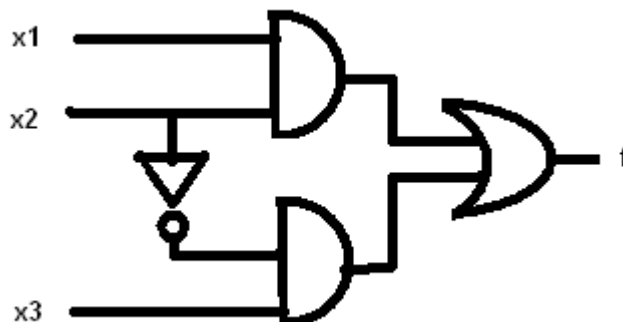
Template Section:

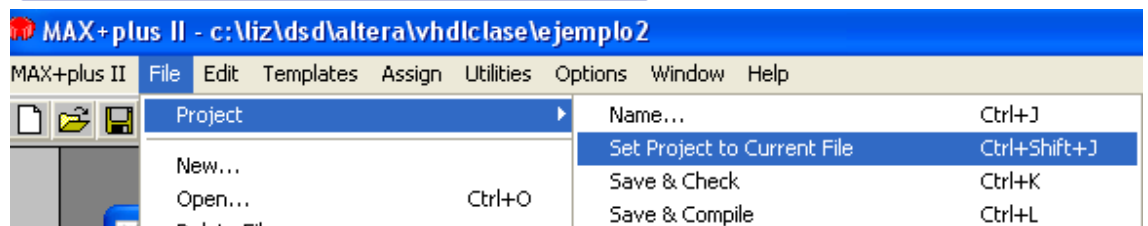
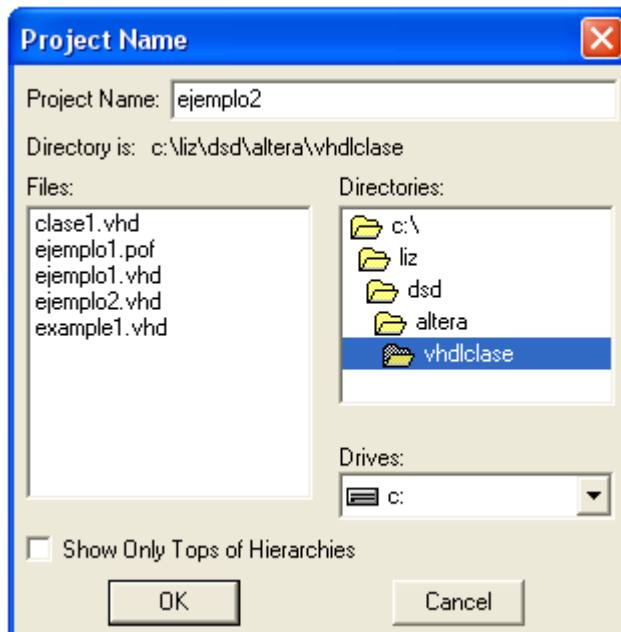
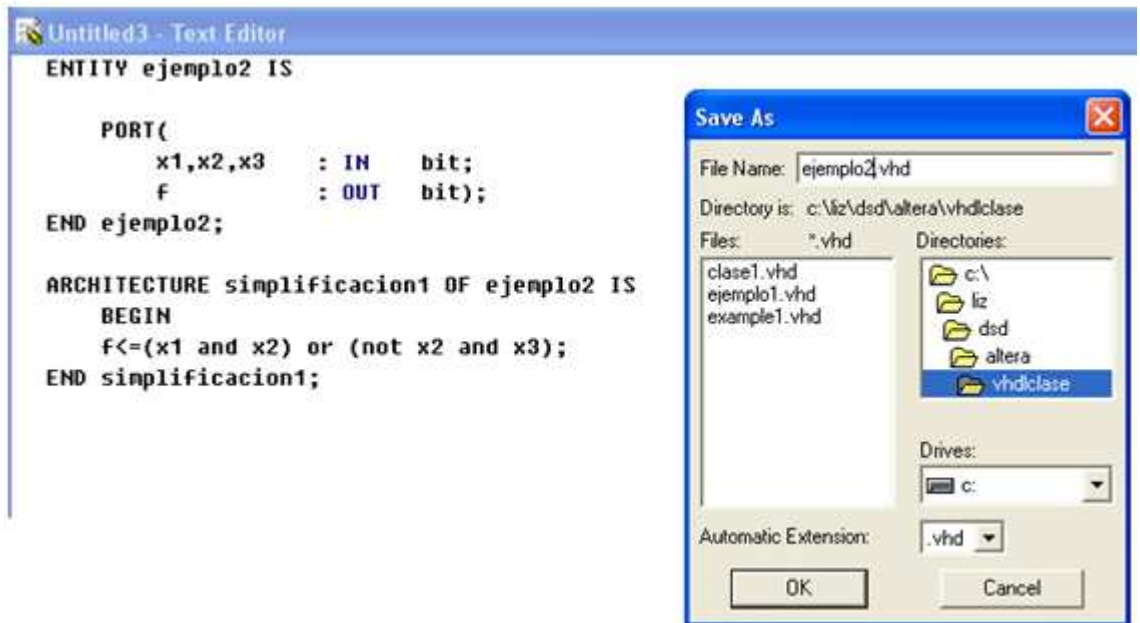
- Overall Structure
- Full Design: Counter
- Full Design: Flipflop
- Full Design: Tri-State Buffer
- Architecture Body
- Case Statement
- Component Declaration
- Component Instantiation Statement
- Concurrent Procedure Call
- Concurrent Signal Assignment Statement
- Conditional Signal Assignment
- Constant Declaration
- Entity Declaration
- For Statement
- Generate Statement (For Generate)
- Generate Statement (If Generate)
- If Statement
- Library Clause
- Package Declaration

OK Cancel

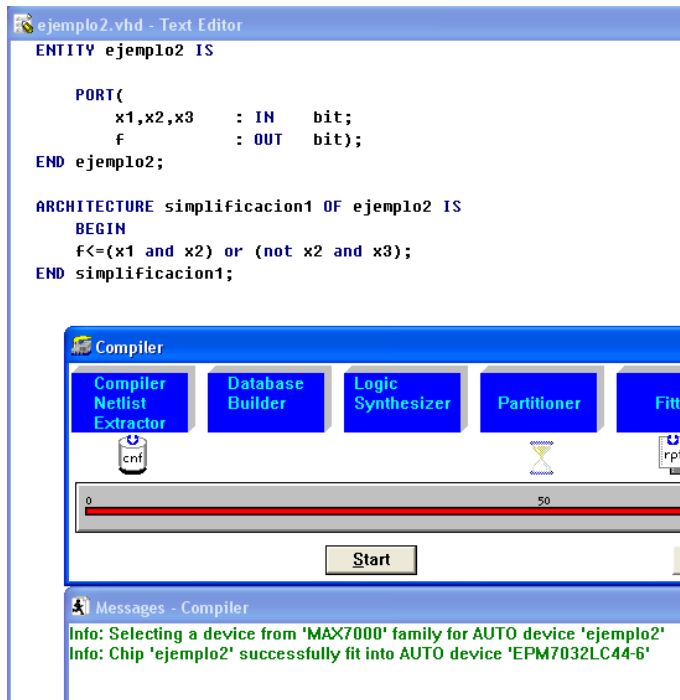
Line 10 Col 1 INS

Vamos a adaptar nuestro resultado al vhd, únicamente con tipos de datos BIT. Por lo tanto su comportamiento se escribirá con respecto a estos bits.

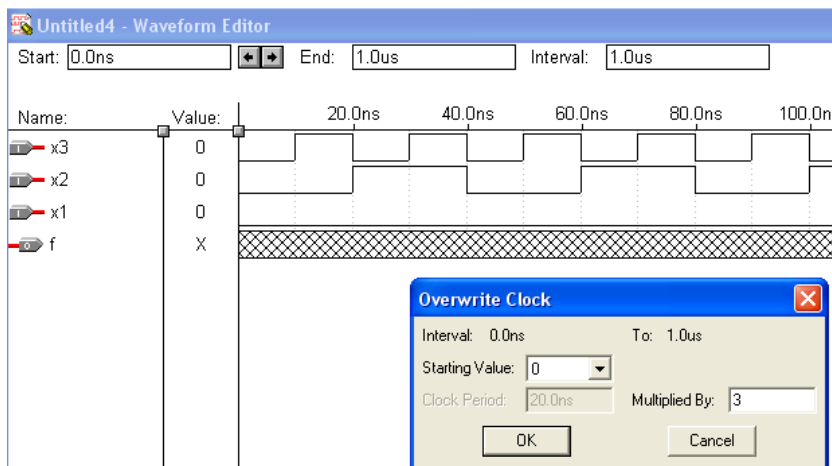




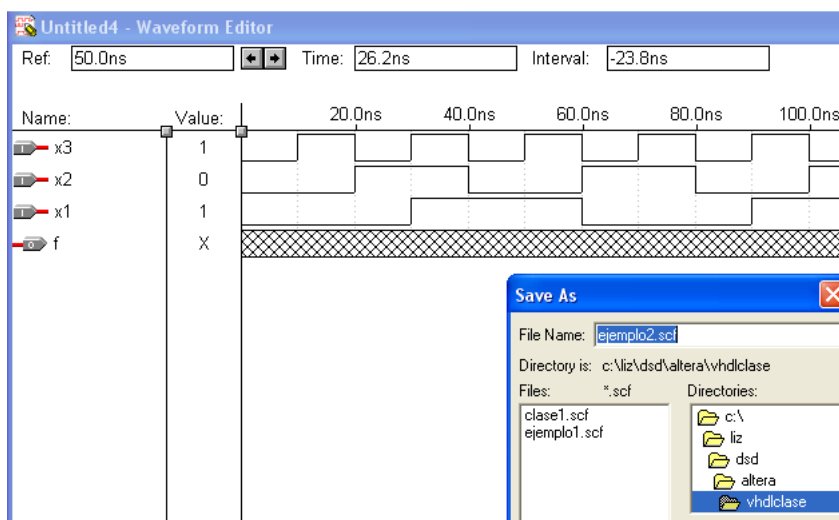
Compilo.



Inserto la forma de onda con datos de entrada



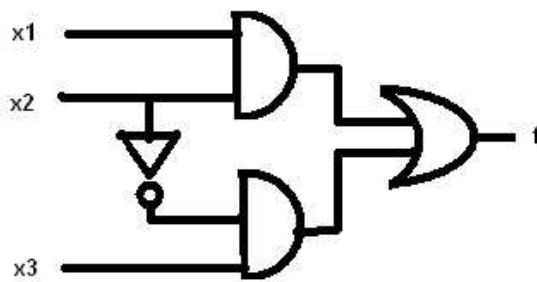
Quedándose así. Y guardo.



Nos resta simular

The image shows the simulation environment. The Waveform Editor displays the timing of signals x3, x2, x1, and f. The Simulator window shows the simulation input and timing parameters. The MAX+plus II - Simulator window shows a successful simulation message.

Información Relacionada.

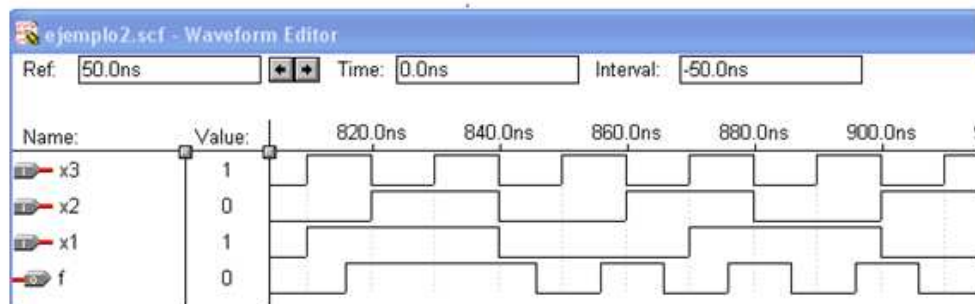


```

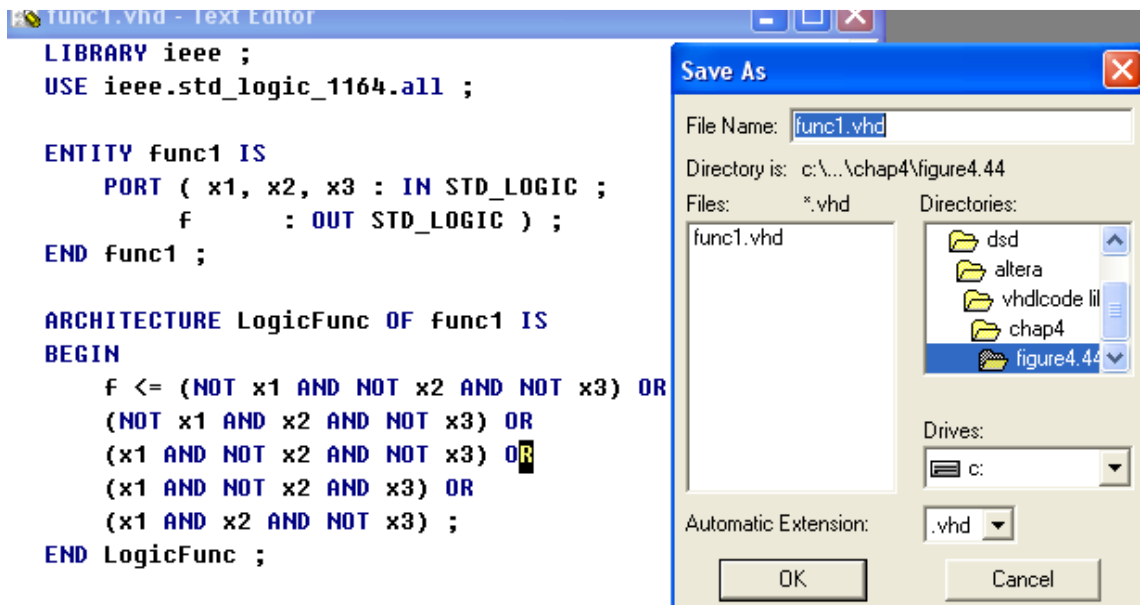
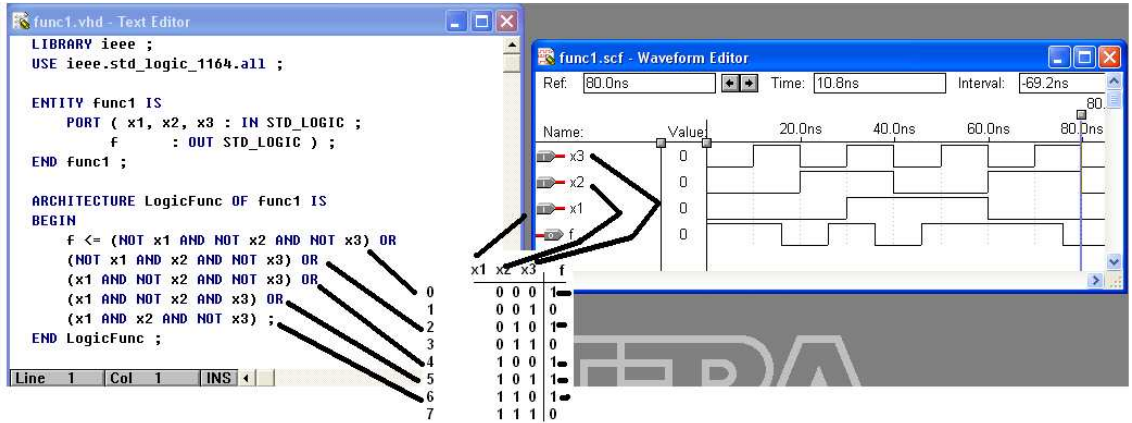
ENTITY ejemplo2 IS
    PORT(
        x1,x2,x3 : IN bit;
        f : OUT bit);
END ejemplo2;

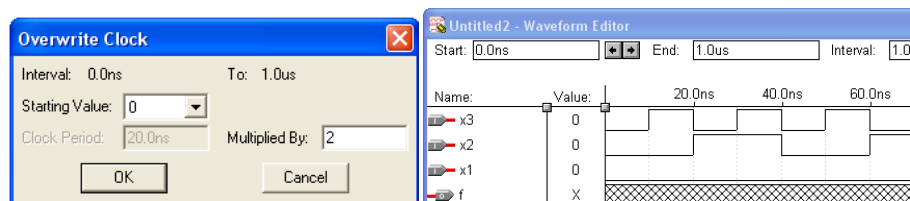
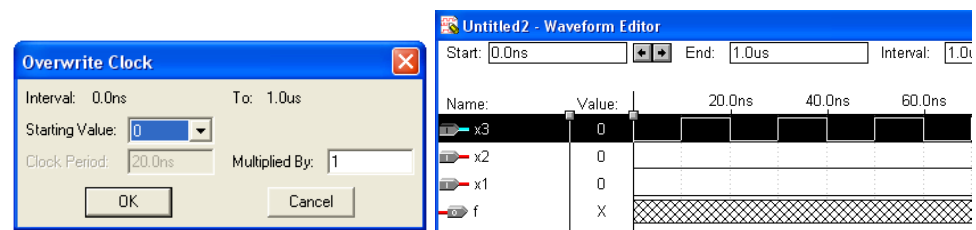
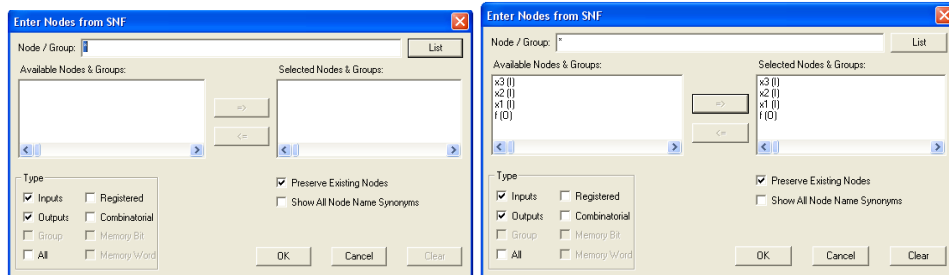
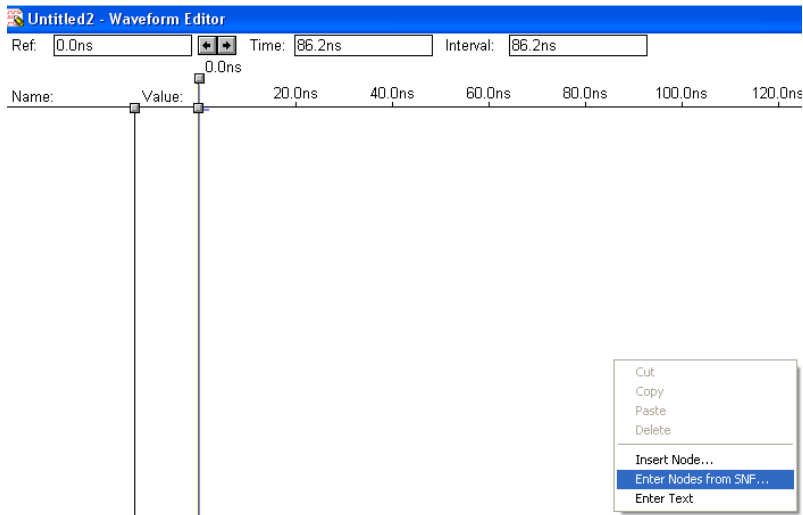
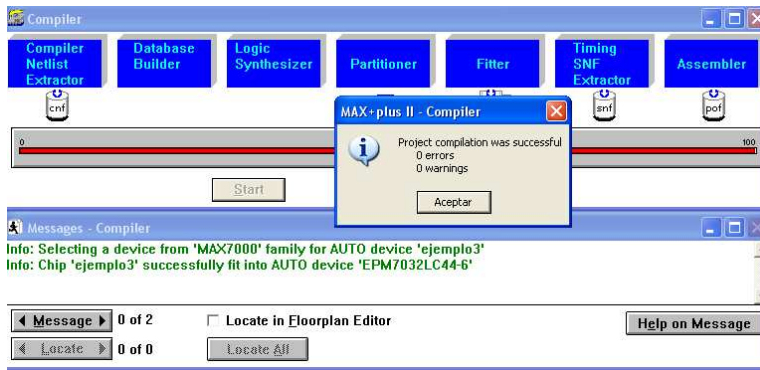
ARCHITECTURE simplificacion1 OF ejemplo2 IS
    BEGIN
        f<=(x1 and x2) or (not x2 and x3);
    END simplificacion1;

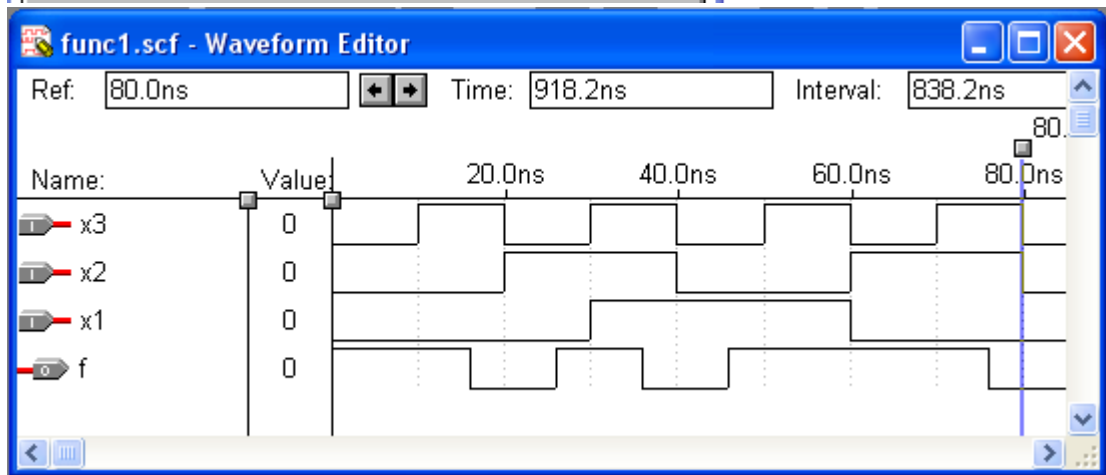
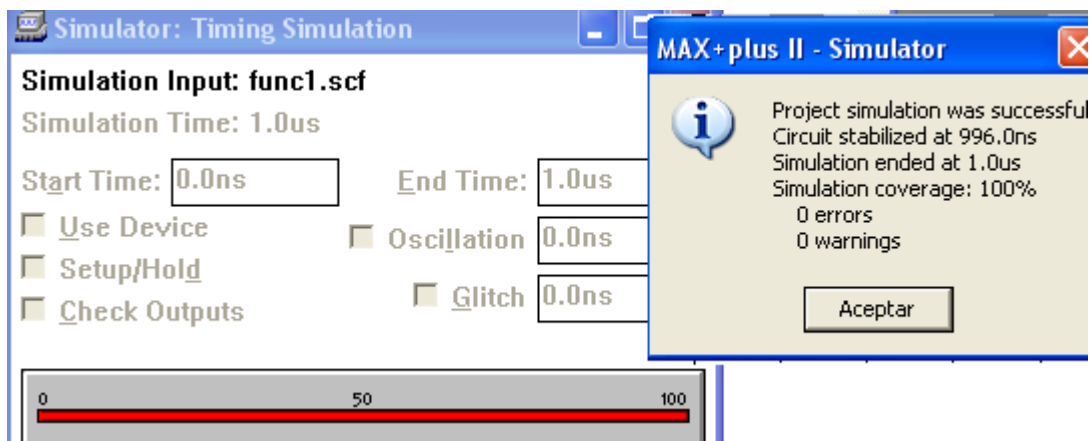
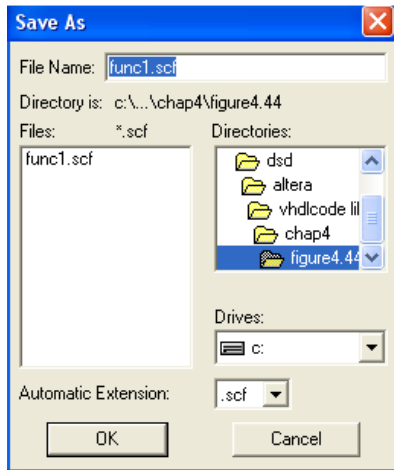
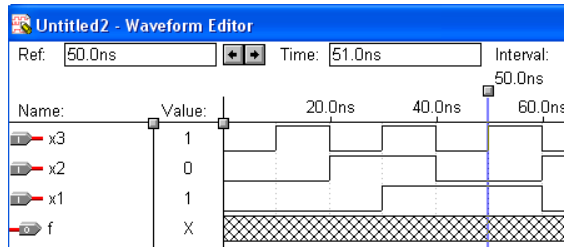
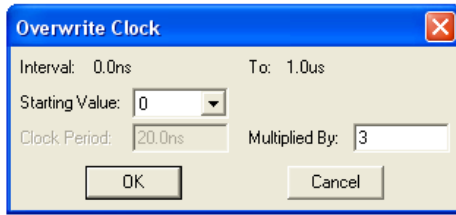
```



Practica3. Implementado una tabla de verdad cualquiera.







Relacionemos información

```

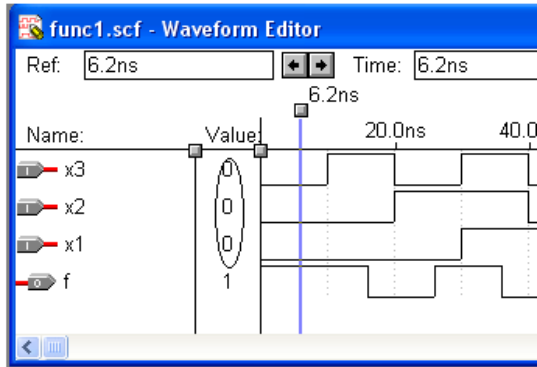
func1.vhd - Text Editor
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY func1 IS
    PORT ( x1, x2, x3 : IN STD_LOGIC ;
          f      : OUT STD_LOGIC ) ;
END func1 ;

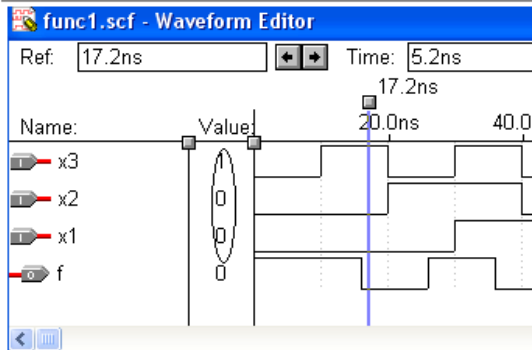
ARCHITECTURE LogicFunc OF func1 IS
BEGIN
    f <= (NOT x1 AND NOT x2 AND NOT x3) OR
        (NOT x1 AND x2 AND NOT x3) OR
        (x1 AND NOT x2 AND NOT x3) OR
        (x1 AND NOT x2 AND x3) OR
        (x1 AND x2 AND NOT x3) ;
END LogicFunc ;

```

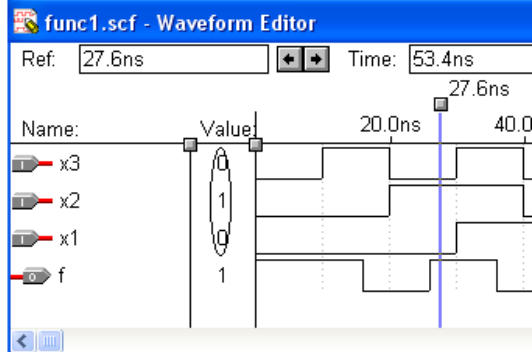
	x1	x2	x3	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



	x1	x2	x3	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



	x1	x2	x3	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



Etc.