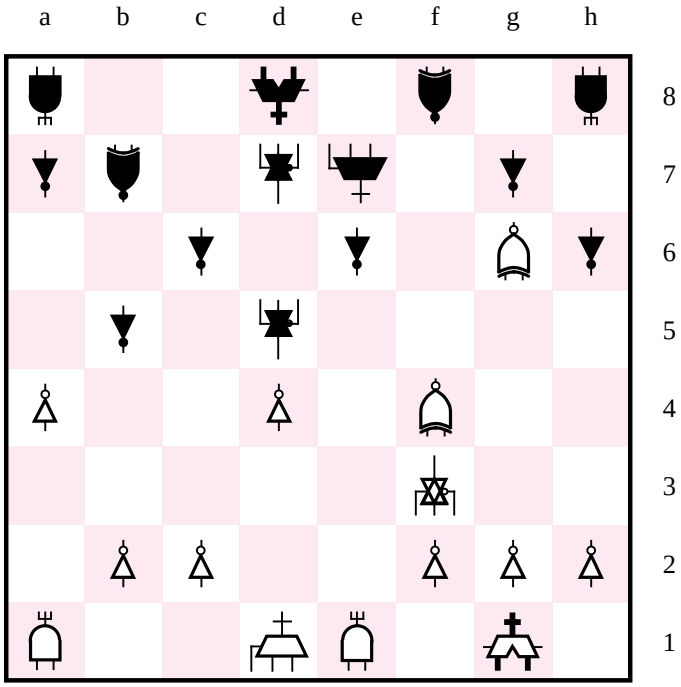
appendix

B

Tutorial 1



13. Rf1–e1, Nf6–d5

MAX+plusII is one of the most sophisticated and easiest to use CAD systems available on the market. In this tutorial we introduce the design of logic circuits using MAX+plusII. Step-by-step instructions are presented for performing design entry with three methods: using schematic capture, writing VHDL code, and using a truth table. The tutorial also illustrates functional simulation.

B.1 INTRODUCTION

This tutorial assumes that the reader has access to a computer on which MAX+plusII is installed. Instructions for installing the copy of MAX+plusII provided with the book are included with the CD-ROM. The MAX+plusII software will run on several different types of computer systems. For this tutorial a computer running a Microsoft operating systems (Windows95, Windows98, or WindowsNT) is assumed. Although MAX+plusII operates similarly on all of the supported types of computers, there are some minor differences. A reader who is not using a Microsoft Windows operating system may experience some slight discrepancies from this tutorial. Examples of potential differences are the locations of files in the computer's file system and the exact appearance of windows displayed by the software. All such discrepancies are minor and will not affect the reader's ability to follow the tutorial.

This tutorial does not describe how to use the operating system provided on the computer. We assume that the reader already knows how to perform actions such as running programs, operating a mouse, moving, resizing, minimizing and maximizing windows, creating directories (folders) and files, and the like. A reader who is not familiar with these procedures will need to learn how to use the computer's operating system before proceeding.

B.1.1 GETTING STARTED

Each logic circuit, or subcircuit, being designed in MAX+plusII is called a *project*. The software works on one project at a time and keeps all information for that project in a single directory in the file system (we use the traditional term *directory* for a location in the file system, but in Microsoft Windows the term *folder* is used). To begin a new logic circuit design, the first step is to create a directory to hold its files. As part of the installation of the MAX+plusII software, a few sample projects are placed into a directory called *max2work*. To hold the design files for this tutorial, we created the subdirectory *max2work**tutorial1*. The location and name of the directory is not important; hence the reader may use any valid directory.

To create a directory to work in, use the normal utilities provided by the computer's operating system. MAX+plusII is not involved in this step. After the directory has been created, start the MAX+plusII software. You should see a window similar to the one in Figure B.1. This window is called the *MAX+plusII Manager*. It provides access to all the features of MAX+plusII, which the user selects with the computer mouse.

Most of the commands provided by MAX+plusII are accessed by using a set of menus that are located in the Manager window below the title bar. For example, in Figure B.1

Figure B.1 The MAX+plusII Manager window.

clicking the left mouse button on the menu named File opens the menu shown in Figure B.2. Clicking the left mouse button on the entry Exit MAX+plusII Alt+F4 exits from MAX+plusII. In general, whenever the mouse is used to select something, the *left* button is used. Hence we will not normally specify which button to use. In the few cases when it is necessary to use the *right* mouse button, it will be specified explicitly. We should note that the Alt+F4 part of

Figure B.2 The File menu in the Manager window.

the menu item indicates a keyboard shortcut; instead of using the mouse, the command can alternatively be invoked by the holding down the Alt key on the keyboard and pressing the F4 function key. Keyboard shortcuts are available for a few of the MAX+plusII commands, but commands are usually invoked using the mouse. For some commands it is necessary to access two or more menus in sequence. We use the convention Menu1 | Menu2 | Item to indicate that to select the desired command the user should first click the left mouse button on Menu1, then within this menu click on Menu2, and then within Menu2 click on Item. For example, File | Exit MAX+plusII describes how to use the mouse to exit from the MAX+plusII system.

The MAX+plusII system includes 11 main software modules, called *applications*. They can be accessed in two different ways. First, all the applications can be invoked via the MAX+plusII menu in the Manager window, as illustrated in Figure B.3. Second, some of the applications can be invoked using the small icons that appear below the Manager title bar. (If no icons are visible under the Manager title bar, select Options | Preferences to open the Preferences dialog box. Then use the mouse to place a check mark beside the entry for Show Toolbar and click OK.) To see which applications in Figure B.3 a particular icon is associated with, place the mouse pointer on top of the icon; the Manager displays a message near the bottom of the window that gives the name of the application.

The applications introduced in this tutorial include the Graphic Editor, Text Editor, Waveform Editor, Compiler, Simulator, Message Processor, and Hierarchy Display. The others are introduced in Tutorial 2.

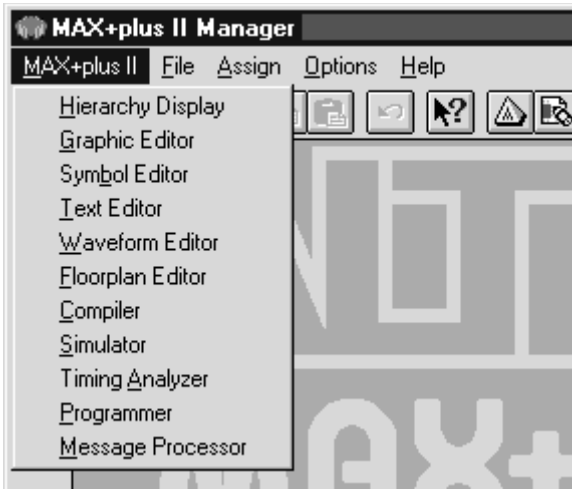


Figure B.3 The MAX+plus II menu in the Manager window.

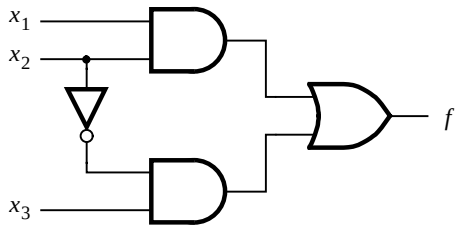
MAX+plusII On-Line Help

MAX+plusII provides comprehensive on-line documentation that answers most of the questions that may arise when using the software. The documentation is accessed from the Help menu in the Manager window. To get some idea of the extent of documentation provided, it is worthwhile for the reader to browse through the Help menu. For instance, selecting Help | MAX+plusII Table of Contents shows all the categories of documentation available.

The user can quickly search through the Help topics by selecting Help | Search for Help on, which opens a dialog box into which keywords can be entered. The available Help topics that match the keywords are automatically displayed. Two other methods are provided for quickly finding documentation for specific topics. First, while using any application, pressing the F1 function key on the keyboard opens a Help display that shows the commands available for that application. Second, in some instances holding down the Shift key and pressing the F1 key changes the mouse pointer into a *help* pointer. This feature is available when using the schematic capture tool provided in MAX+plusII. Clicking the help pointer on any circuit element in a schematic automatically displays any documentation that is available for that circuit element.

B.2 DESIGN ENTRY USING SCHEMATIC CAPTURE

In Chapter 2 we introduced three types of design entry methods: truth tables, schematic capture, and VHDL. This section illustrates the process of using the schematic capture tool provided in MAX+plusII, which is called the Graphic Editor. As a simple example, we will draw a schematic for the logic function $f = x_1x_2 + \overline{x}_2x_3$. A circuit diagram for f was shown in Figure 2.26 and is reproduced as Figure B.4*a*. The truth table for f is given in Figure B.4*b*. Chapter 2 also introduced functional simulation. After creating the schematic, we show how to use the functional simulator in MAX+plusII to verify the schematic's functionality.



(a) Circuit

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(b) Truth table

Figure B.4 The logic function of Figure 2.26.

B.2.1 SPECIFYING THE PROJECT NAME

As a first step we will specify the name of the design project. In the Manager window select File | Project | Name to open the pop-up box illustrated in Figure B.5. It is necessary to specify the location of the directory where MAX+plusII will store any files created for the project. For this example the directory used is named d:*max2work**tutorial1*. The disk drive designation, d:, is selected using the Drives pull-down menu shown in Figure B.5. The directory name is selected using the box labeled Directories. Use the mouse to double-click on the directory names displayed in the box until the proper directory is selected; the selected directory appears next to the words Directory is, as illustrated in the figure. In the box labeled Project Name, type *graphic1* as the name for this project and then click OK. Observe that the name of the project is displayed in the title bar of the Manager window.

B.2.2 USING THE GRAPHIC EDITOR

The next step is to draw the schematic. In the Manager window select MAX+plusII | Graphic Editor. The Graphic Editor window appears inside the Manager window. It may be helpful to move or resize the Graphic Editor window and to increase the size of the Manager window to provide more work space. In the screen capture in Figure B.6, the Graphic Editor window is maximized so that it fills the entire Manager window.

The title bar in Figure B.6 includes some menu names and icons that did not appear in Figure B.1. This is because the Manager window always indicates the features available in whatever application is currently being used. A number of icons that are used to invoke Graphic Editor features also appear along the left edge of the window. To see a description of the Graphic Editor feature associated with each icon, position the mouse on top of the icon; a message is displayed near the bottom of the window. Two of the most useful icons

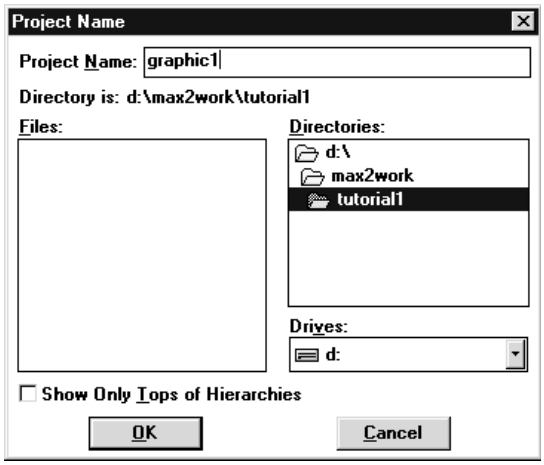


Figure B.5 Specifying the name and working directory for a project.

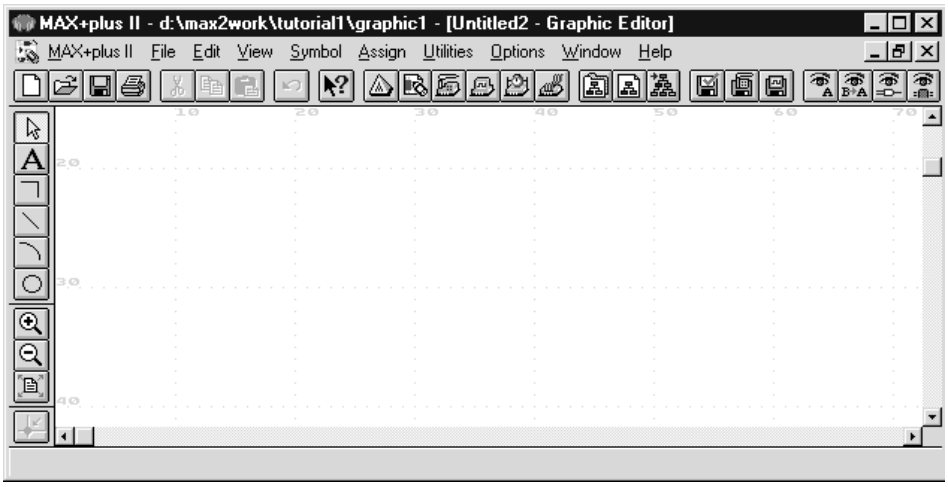


Figure B.6 The Graphic Editor display.

are the ones that look like a magnifying glass. These icons are used to see a larger or smaller view of the schematic.

Naming the Schematic

The schematic being created must be given a name. Select File | Save As to open the pop-up box depicted in Figure B.7. The directory that we chose for the project is already selected in the pop-up box. The Graphic Editor will create a separate file for the schematic and store it in the project's directory. In the box labeled File Name, type *graphic1.gdf*.

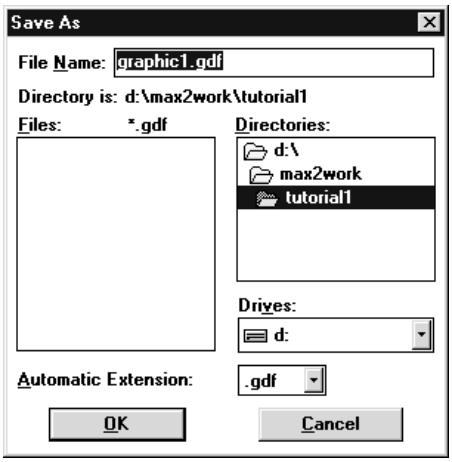


Figure B.7 Specifying the name of a schematic.

You must use exactly this name. The name *graphic1* must match the name of the project, and the filename extension *gdf*, which stands for *graphic design file*, must be used for all schematics. Click OK to return to the Graphic Editor.

Importing Logic-Gate Symbols

The Graphic Editor provides several libraries which contain circuit elements that can be imported into a schematic. For our simple example we will use a library called *Primitives*, which contains basic logic gates. To access the library, double-click on the blank space in the middle of the Graphic Editor display to open the pop-up box in Figure B.8 (another way to open this box is to select Symbol | Enter Symbol). The box labeled Symbol Libraries lists several available libraries, including the Primitives library. To open it, double-click on the line that ends with the word *prim*. A list of the logic gates in the library is automatically displayed in the Symbol Files box. Double-click on the *and2* symbol to import it into the schematic (you can alternatively click on *and2* and then click OK). A two-input AND-gate symbol now appears in the Graphic Editor window.

Any symbol in a schematic can be selected using the mouse. Position the mouse pointer on top of the AND-gate symbol in the schematic and click the mouse to select it. The symbol is highlighted in red. To move a symbol, select it and, while continuing to press the mouse button, drag the mouse to move the symbol. To make it easier to position the graphical symbols, a grid of guidelines can be displayed in the Graphic Editor window by selecting Options | Show Guidelines. Spacing between grid lines can be adjusted using Options | Guideline Spacing.

The logic function f requires a second two-input AND gate, a two-input OR gate, and a NOT gate. Use the following steps to import them into the schematic.

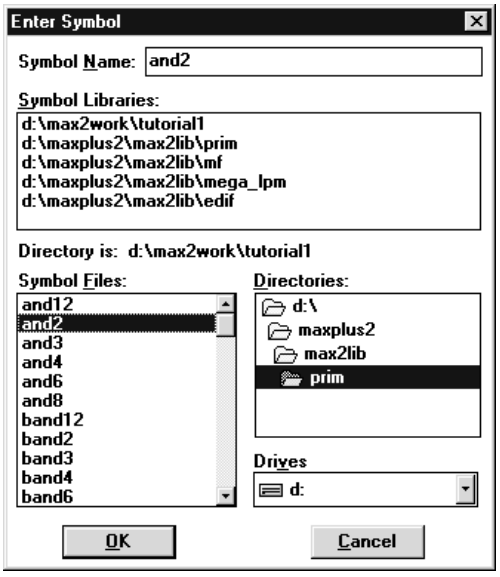


Figure B.8 Importing a logic gate from the Primitives library.

Position the mouse pointer over the AND-gate symbol that has already been imported. Press and hold down the Ctrl keyboard key and click and drag the mouse away from the AND-gate symbol. The Graphic Editor automatically imports a second instance of the AND-gate symbol. This shortcut procedure for making a copy of a circuit element is convenient when you need many instances of the same element in a schematic. Of course, an alternative approach is to import each instance of the symbol by opening the Primitives library as described above.

To import the OR-gate symbol, again double-click on a blank space in the Graphic Editor and then double-click on the Primitives library. In the box labeled Symbol Files, use the scroll bar to scroll down through the list of gates to find the symbol named *or2*. Import this symbol into the schematic. Next import the NOT gate using the same procedure. To orient the NOT gate so that it points downward, as depicted in Figure B.4*a*, select the NOT-gate symbol and then use the command Edit | Rotate | 270 to rotate the symbol 270 degrees counterclockwise. The symbols in the schematic can be moved by selecting them and dragging the mouse, as explained above. More than one symbol can be selected at the same time by clicking the mouse and dragging an outline around the symbols. The selected symbols are moved together by clicking on any one of them and moving it. Experiment with this procedure. Arrange the symbols so that the schematic appears similar to the one in Figure B.9.

Importing Input and Output Symbols

Now that the logic-gate symbols have been entered, it is necessary to import symbols to represent the input and output ports of the circuit. Open the Primitives library again. Click the mouse anywhere in the box labeled Symbol Files and then type the letter "i" to jump ahead in the list of symbols to those whose names begin with *i*. This shortcut can be used in addition to the scroll bars provided on the Symbol Files box. Import the symbol

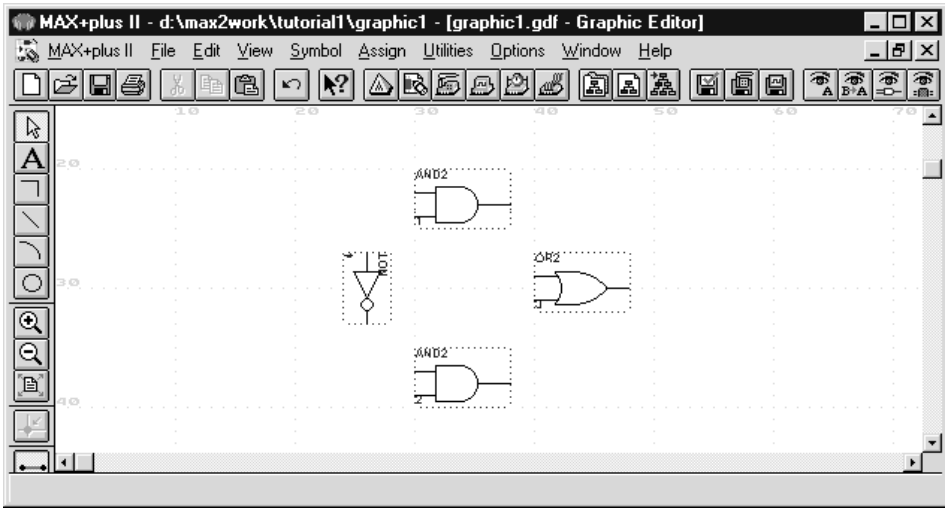


Figure B.9 A partially completed schematic for the circuit in Figure B.4.

named *input* into the schematic. Import two additional instances of the input symbol. To represent the output of the circuit, open the Primitives library and import the symbol named output. Arrange the symbols to appear as illustrated in Figure B.10.

Assigning Names to Input and Output Symbols

Point to the word PIN_NAME on the input pin symbol in the upper-left corner of the schematic and double-click the mouse. The pin name is selected, allowing a new pin name to be typed. Type *x*1 as the pin name. Hitting carriage return immediately after typing the pin name causes the mouse focus to move to the pin directly below the one currently being named. This method can be used to name any number of pins. Assign the names *x*2 and *x*3 to the middle and bottom input pins, respectively. Finally, assign the name *f* to the output pin.

Connecting Nodes with Wires

The next step is to draw lines (wires) to connect the symbols in the schematic together. Click on the icon that looks like an arrowhead along the left edge of the Manager window. This icon is called the Selection tool, and it allows the Graphic Editor to change automatically between the modes of selecting a symbol on the screen or drawing wires to interconnect symbols. The appropriate mode is chosen depending on where the mouse is pointing.

Move the mouse pointer on top of the *x*1 input symbol. The mouse pointer appears as an arrowhead when pointing anywhere on the symbol except at the right edge. The arrowhead means that the symbol will be selected if the mouse button is pressed. Move the mouse to point to the small line, called a *pinstub*, on the right edge of the *x*1 input symbol. The mouse pointer changes to a crosshair, which allows a wire to be drawn to connect the

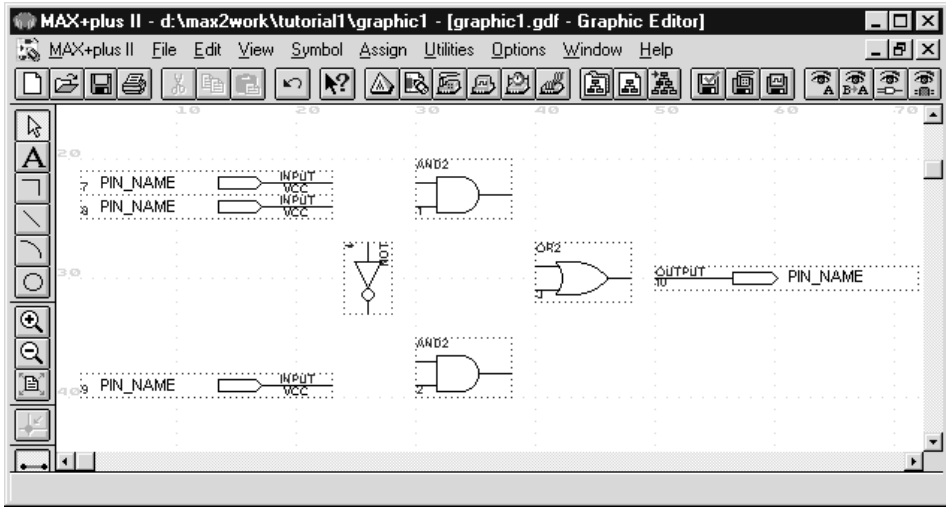


Figure B.10 Input and output symbols added to the schematic in Figure B.9.

pinstub to another location in the schematic. A connection between two or more pinstubs in a schematic is called a *node*. The name derives from electrical terminology, where the term *node* refers to any number of points in a circuit that are connected together by wires and thus have the same voltage.

Connect the input symbol for $x1$ to the AND gate at the top of the schematic as follows. While the mouse is pointing at the pinstub on the $x1$ symbol, click and hold the mouse button. Drag the mouse to the right until the line (wire) that is drawn reaches the pinstub on the top input of the AND gate; then release the button. The two pinstubs are now connected and represent a single node in the circuit.

Use the same procedure to draw a wire from the pinstub on the $x2$ input symbol to the other input on the AND gate. Then draw a wire from the pinstub on the input of the NOT gate upward until it reaches the wire connecting $x2$ to the AND gate. Release the mouse button and observe that a connecting dot is drawn automatically. The three pinstubs corresponding to the $x2$ input symbol, the AND-gate input, and the NOT-gate input now represent a single node in the circuit. Figure B.11 shows a magnified view of the part of the schematic that contains the connections drawn so far. To increase or decrease the portion of the schematic displayed on the screen, use the icons that look like magnifying glasses on the left side of the Manager window.

To complete the schematic, connect the output of the NOT gate to the lower AND gate and connect the input symbol for $x3$ to that AND gate as well. Connect the outputs of the two AND gates to the OR gate and connect the OR gate to the f output symbol. If any mistakes are made while connecting the symbols, erroneous wires can be selected with the mouse and then removed by pressing the Delete key or by selecting Edit | Delete. The finished schematic is depicted in Figure B.12. Save the schematic using File | Save.

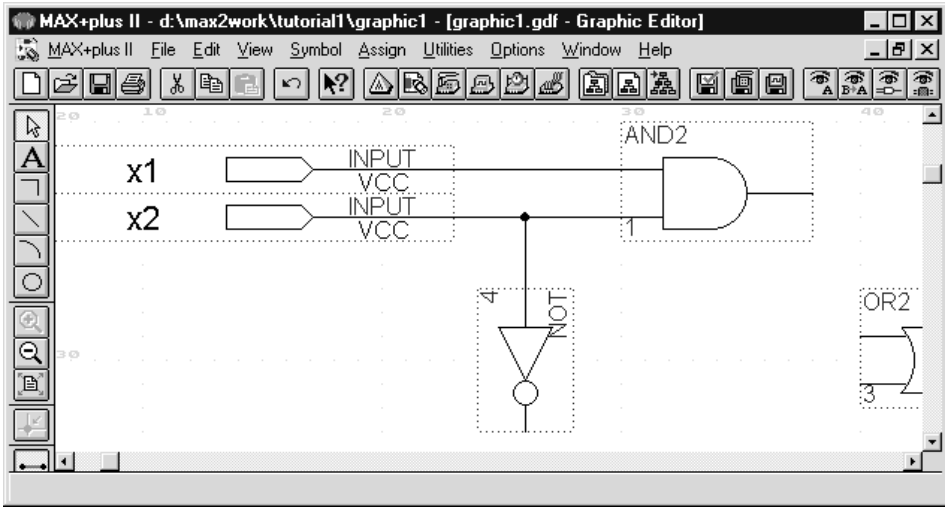


Figure B.11 Connecting the symbols in the schematic from Figure B.10.

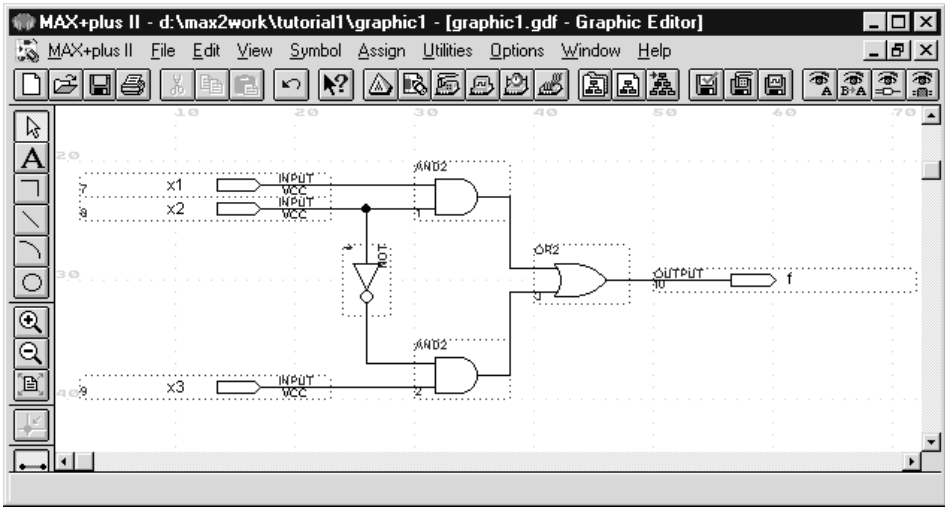


Figure B.12 The completed schematic for the circuit in Figure B.4.

Since our example schematic is quite simple, it is easy to draw all the wires in the circuit without producing a messy diagram. However, in larger schematics some nodes that have to be connected may be far apart, in which case it is awkward to draw wires between them. In such cases the nodes are connected by assigning labels to them, instead of drawing wires. We will illustrate this method of connecting nodes in section D.3.1.

B.2.3 SYNTHESIZING A CIRCUIT FROM THE SCHEMATIC

As we explained in section 2.8.2, after a schematic is entered into a CAD system, it is processed by initial synthesis tools. These tools analyze the schematic and generate a Boolean equation for each logic function in the circuit. In MAX+plusII the synthesis tools are controlled by the application program called the *Compiler*.

Using the Compiler

To open the Compiler window, click the mouse on the Compiler icon (it looks like a factory with a smoke stack) below the Manager window title bar or select MAX+plusII | Compiler.

For this tutorial we will use only the tools that are needed to allow us to perform a functional simulation of the schematic. To tell the Compiler to use these tools, select Processing | Functional SNF Extractor. The Compiler window should appear as shown in Figure B.13. The window shows three software modules that are invoked in sequence by the Compiler. The Compiler Netlist Extractor and Database Builder represent the initial synthesis tools. The module called Functional SNF Extractor creates a file, called a *simulator netlist file (SNF)*, which describes the functionality of the circuit and is used by the functional simulator.

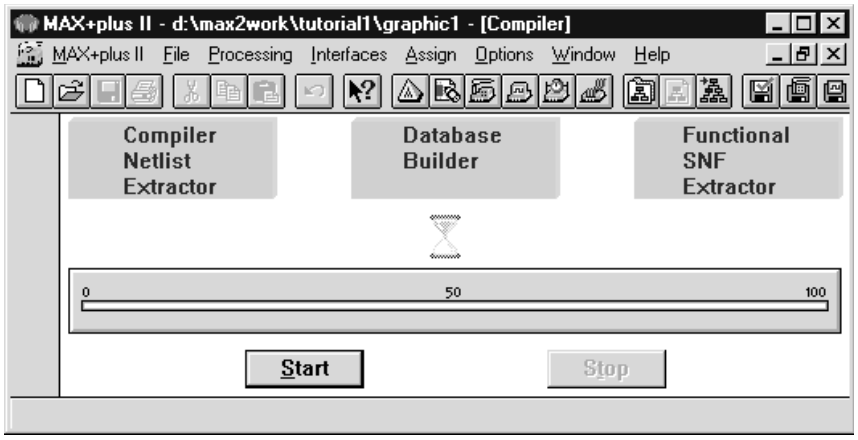


Figure B.13 The Compiler display.

Click the mouse on the Start button in the Compiler window. The Compiler indicates its progress by displaying a red progress bar and by placing an icon under each of the three software modules as they are executed. When the Compiler is finished, a window should be displayed that indicates zero warnings and zero errors. Click OK in this window to return to the Compiler window.

If the Compiler does not specify zero warnings and zero errors, then at least one mistake has been made when entering the schematic. In this case the Compiler opens a window called the Message Processor, which displays a message concerning each warning or error generated. An example showing how the Message Processor can be used to quickly locate and fix errors in a schematic is given in section B.2.5.

To close the Compiler window, use the *Close button* (it is an X) located in the top-right corner of its window.

B.2.4 PERFORMING FUNCTIONAL SIMULATION

Before the schematic can be simulated, it is necessary to create the desired waveforms, called *test vectors*, to represent the input signals. For this tutorial we will use the MAX+plusII Waveform Editor to draw test vectors, but it is also possible to use a text editor to create test vectors in a plain text (ASCII) file. Documentation pertaining to ASCII test vectors can be opened by selecting Help | MAX+plusII Table of Contents. Click on Simulator, then click on Basic Tools, and finally click on Vector File (.vec).

Using the Waveform Editor

Open the Waveform Editor window by selecting MAX+plusII | Waveform Editor. Because the Waveform Editor has many uses, it is necessary to indicate that we wish to enter test vectors for simulation purposes. Select File | Save As and type (if not already

there) *graphic1.scf* in the box labeled File Name. A file with *scf* extension stores the waveforms that will be used as simulation test vectors.

Select Node | Enter Nodes from SNF to open the pop-up box shown in Figure B.14. Click on the List button in the upper-right corner of this box to display the names of the nodes in the current project in the box labeled Available Nodes & Groups. Click the mouse on the name *x*3 to highlight it. Click on the button labeled => to copy *x*3 into the box labeled Selected Nodes & Groups. Use the same procedure to select each of the other signals and copy them into the Selected Nodes & Groups box. It is also possible to select multiple nodes at the same time, by dragging the mouse upward or downward inside the Available Nodes & Groups box. Click OK to return to the Waveform Editor. The nodes *x*1, *x*2, *x*3, and *f* are now shown in the waveform display.

We will now specify the logic values to be used for the input signals during functional simulation. The logic values at the output *f* will be generated automatically by the simulator.

Select File | End Time to specify the total amount of time for which the circuit will be simulated. In the box labeled Time, type *160ns* to set the total simulation time to 160 nanoseconds. This amount of time is rather arbitrary because functional simulation does not include any timing delays, as discussed in section 2.8.3. The concept of *simulation time* will become more significant in Tutorial 2 when timing simulation is introduced. Click OK to return to the Waveform Editor. Select View | Fit in Window so that the entire time range from 0 to 160 ns is visible in the Waveform Editor display. In the Options menu make sure that Show Grid has a check mark next to it so that the Waveform Editor displays light vertical guidelines in the waveform area of the display. The guidelines provide a visual aid for positioning the mouse when drawing waveforms. Select Options | Grid Size and type *20ns* in the box labeled Grid Size. Click the mouse when pointing to any of the guidelines and observe that a vertical reference line is drawn at that point. We will use the reference line in Tutorial 2. Figure B.15 shows how the Waveform Editor window should look at this

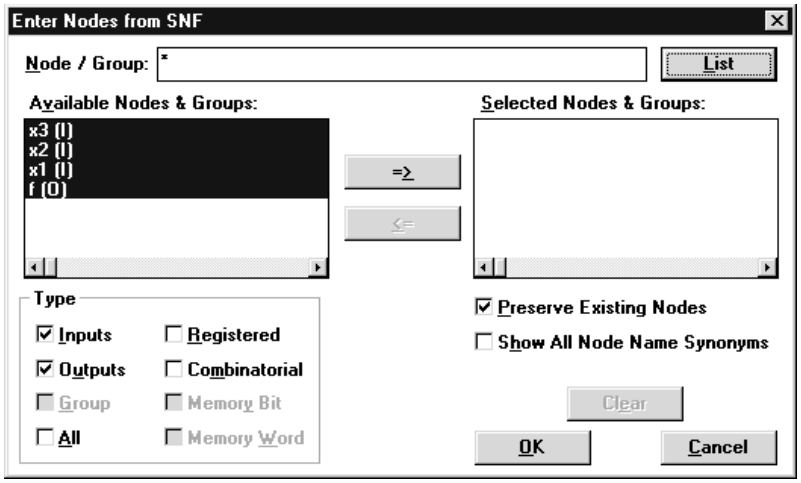


Figure B.14 Selecting nodes for simulation.

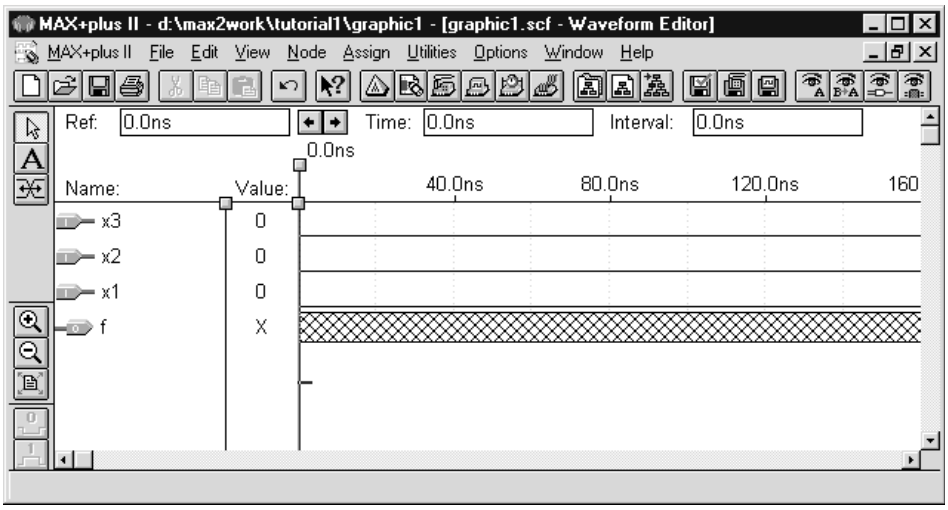


Figure B.15 The Waveform Editor display.

point. The input waveforms are set to logic value 0, and the output is shown as a hashed-line pattern that indicates that the logic value has not yet been determined.

To thoroughly test the circuit during simulation, it is desirable to use as many different values of the input signals as possible. For our small example, there are only eight different valuations, and so it is easy to include all of them. To make all eight valuations fit in the 160 ns simulation time, the signal valuations have to change every 20 ns. To create the waveforms for the input signals, do the following.

Activate the *Waveform Editing* tool by pressing its icon on the left edge of the window. The icon is shown in the top-left corner of Figure B.16; it looks like two arrows pointing left and right. Position the mouse pointer over the waveform for input $x3$ at the 20 ns grid line. Press and drag the mouse to the right to highlight the section of the $x3$ waveform from 20 ns to 40 ns, as illustrated in Figure B.16. The Waveform Editing Tool automatically

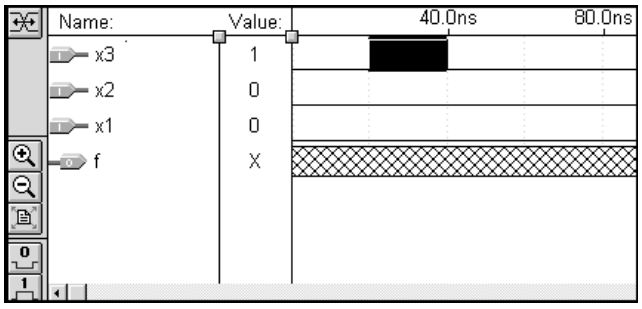


Figure B.16 Editing the waveform for $x3$ from Figure B.15.

changes the selected portion of the waveform from its present value 0 to the value 1. Next select the section of the waveform for $x3$ between 60 ns and 80 ns to set it to 1. Continue in this manner to set every second 20 ns section of $x3$ to 1.

An alternative way to draw waveforms is to use the Selection tool, which is activated by selecting the icon that looks like an arrowhead along the left edge of the window. Using the Selection tool, the procedure for drawing a waveform is to first select a section of the waveform by dragging the mouse over it. The highlighted section can be set to 1 by selecting Edit | Overwrite | High. The highlighted section can also be changed by using the buttons labeled 0 or 1 along the left edge of the window.

Use the Waveform Editing tool to set the waveform for $x2$ to 1 in the range from 40 ns to 80 ns, as well as from 120 ns to 160 ns. Also, set the waveform for $x1$ to 1 in the range from 80 ns to 160 ns. The waveforms drawn, as illustrated in Figure B.17, now include all eight input valuations. Select File | Save to save the waveforms in the *graphic1.scf* file.

Performing the Simulation

To open the Simulator window, shown in Figure B.18, click on its icon (it looks like a computer with a waveform on the screen) or Select MAX+plusII | Simulator. MAX+plusII provides both functional simulation and timing simulation. The type of simulation used by the Simulator application is determined automatically by the settings used in the Compiler application. The Simulator will perform a functional simulation in this case because we instructed the Compiler to generate information for functional simulation, as discussed for Figure B.13.

Observe in Figure B.18 that the Simulator specifies that it will use the file called *graphic1.scf* as the simulator input and will perform the simulation for the time range from 0 to 160 ns. Click the Start button to perform the simulation. The Simulator displays a message indicating that no errors were generated. Click OK to return to the Simulator window. The simulator stores the results of the simulation in the *graphic1.scf* file. To view the file, click on the Open SCF button in the simulator window, which automatically opens the Waveform Editor window and displays the file. As illustrated in Figure B.19, the Simulator creates a waveform for the output *f*. The reader should verify that the generated waveform corresponds to the truth table for *f* given in Figure B.4*b*. The Waveform Editor and Simulator windows can now be closed.

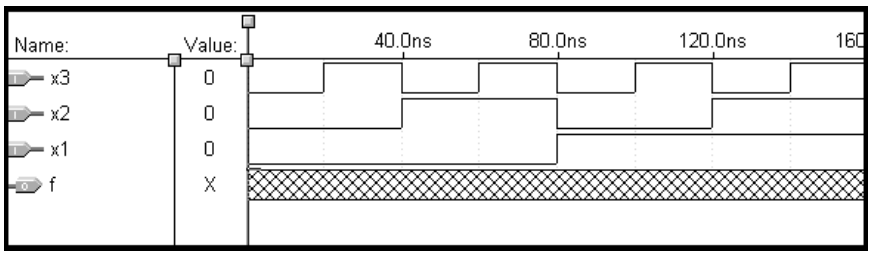


Figure B.17 The completed waveforms for $x1$, $x2$, and $x3$.

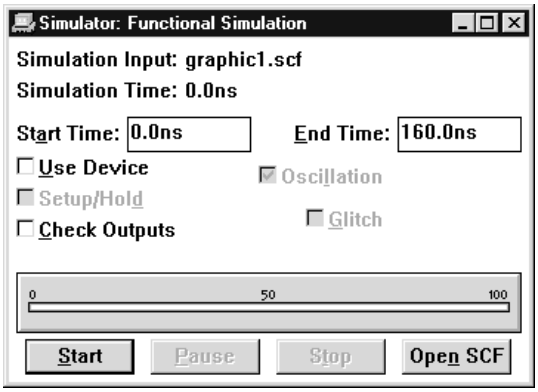


Figure B.18 The Simulator display.

B.2.5 USING THE MESSAGE PROCESSOR TO LOCATE AND FIX ERRORS

In the description in section B.2.3 of how the Compiler is used to synthesize a circuit from the schematic, we said that the Compiler should produce a message stating that no warnings or errors were generated. In this section we illustrate what happens when there is an error in the schematic. To insert an error in the schematic created for f, reopen the schematic by selecting File | Open to open the pop-up box shown in Figure B.20. In the box labeled Show in Files List, click on Graphic Editor Files. Then in the box labeled Files, click on the name

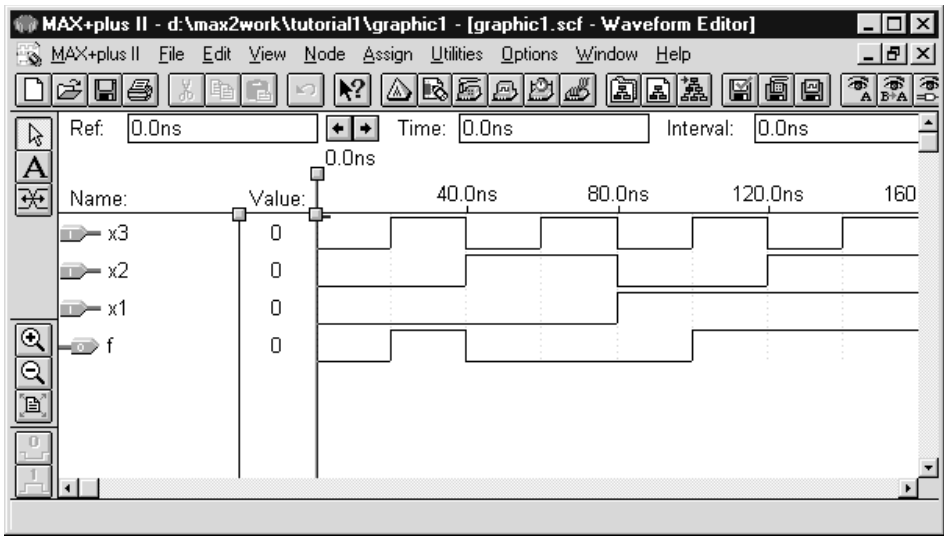


Figure B.19 Functional simulation results for the waveforms in Figure B.17.

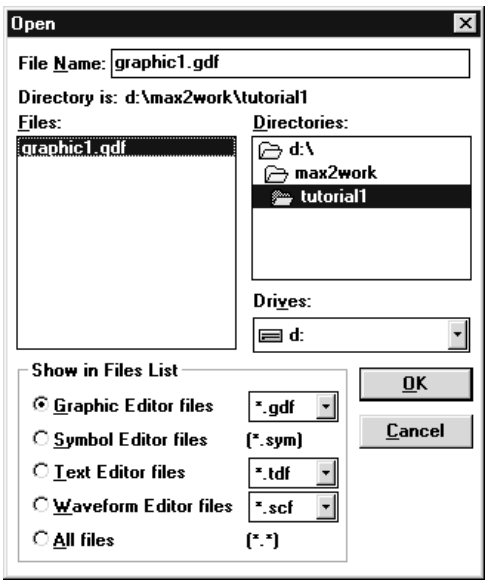


Figure B.20 The dialog box used to reopen the schematic.

graphic1.gdf to put this name in the box labeled File Name. Alternatively, *graphic1.gdf* can be typed into the box rather than using the mouse to select it from the list of files. Click OK to open the file inside the Graphic Editor.

Use the mouse to select the wire that connects the output of the OR gate to the *f* output symbol. Delete the wire by pressing the Delete key; then save the schematic file. Open the Compiler window and run the synthesis tools again. The Compiler should produce a message stating that one warning and one error were found. Click OK. A window, called the Message Processor, is automatically opened to display the messages generated by the Compiler, as illustrated in Figure B.21. If the Message Processor window is obscured by some other window, select MAX+plusII | Message Processor to bring the Message Processor window to the foreground.

The warning message is produced because the OR-gate output is not connected to any other node in the schematic. The error message states that the *f* output symbol is

Figure B.21 The Message Processor display.

not connected to anything. Although it is clear how to fix the error, since we created it purposely, in general some of the messages displayed by the Compiler when synthesizing larger circuits may not be obvious. In such cases it is possible to select a message with the mouse and then click on the Help on Message button in the Message Processor window; documentation that explains the message is automatically opened. Experiment with this feature for both the warning and error messages in Figure B.21.

Another convenient feature of the Message Processor is the Locate button in the lower-left corner of the window. It can be used to automatically display the section of the schematic where the error exists. Select the warning message and then click the Locate button. Observe that the Graphic Editor is automatically displayed with the OR gate highlighted. Next select the error message in the Message Processor window and then click the Locate button again. The f output symbol becomes highlighted in the Graphic Editor.

Use the Graphic Editor to redraw the missing wire between the OR-gate output and the f output symbol. Save the schematic and then use the Compiler to run the synthesis tools to see that the error is fixed. We have now completed our introduction to design using schematic capture. If any application windows are still open, close them to return to the Manager window.

B.3 DESIGN ENTRY USING VHDL

This section illustrates the process of using MAX+plusII to implement logic functions by writing VHDL code. We will implement the function f from section B.2, where we used schematic capture. After typing the VHDL code, it will be simulated with the Functional Simulator.

B.3.1 SPECIFYING THE PROJECT NAME

We need a new project name for the VHDL design. In the Manager window select File | Project | Name. We will store the design files for the project in the same directory that we used for the schematic capture design created earlier. In the box labeled Project Name, type *example1* as the name for the project and then click OK. The name of the project is displayed in the title bar of the Manager window.

B.3.2 USING THE TEXT EDITOR

MAX+plusII provides a text editor that can be used for typing VHDL code. Open the Text Editor window by selecting MAX+plusII | Text Editor. The first step is to specify a name for the file that will be created. Select File | Save As to open the pop-up box depicted in Figure B.22. Type *example1.vhd* in the box labeled File Name. You must use exactly this name. The name *example1* must match the name of the project, and the filename extension *vhd* must be used for all files that contain VHDL code. When File | Save As is selected, the Text Editor places the default name *example1.tdf* in the File Name box. The *tdf* extension stands

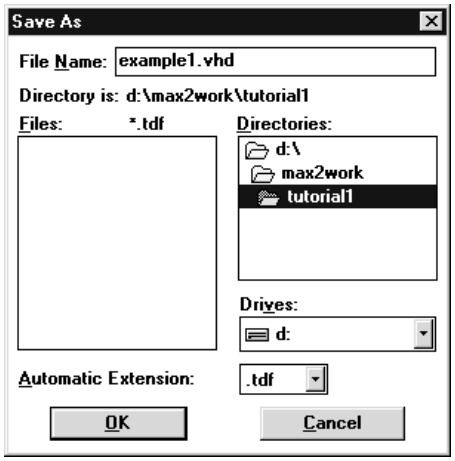


Figure B.22 Specifying a name for the VHDL design file.

for text design file. It is used for files that contain source code written in the Altera Hardware Description Language (AHDL), which is another language supported by the MAX+plusII system. Make sure to change the filename extension from *tdf* to *vhd*. We should mention that it is not necessary to use the Text Editor provided in MAX+plusII. Any text editor can be used to create the file named *example1.vhd*, as long as the text editor can generate a plain text (ASCII) file.

The VHDL code for this example is shown in Figure 2.29. Type the code into the Text Editor to obtain the display in Figure B.23. Most of the commands available in the Text Editor are self-explanatory. Text is entered at the *insertion point*, which is indicated by a thin vertical line. The insertion point can be moved either by using the keyboard arrow keys

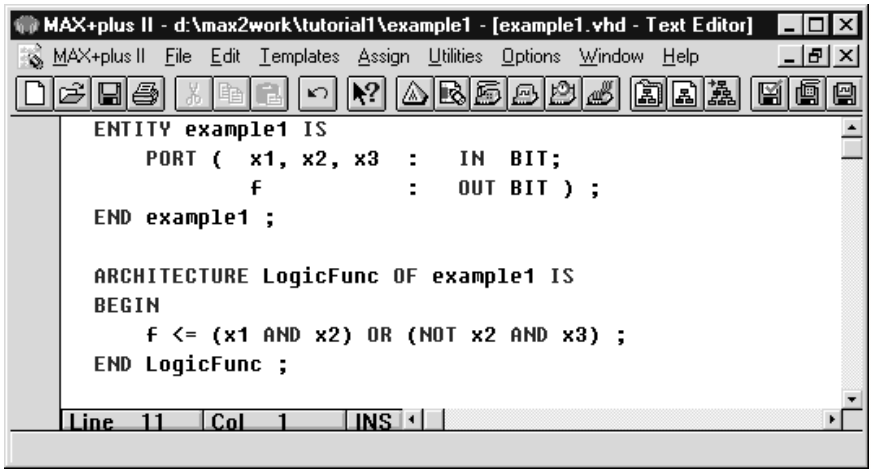


Figure B.23 The Text Editor display showing the VHDL code from Figure 2.29.

or by using the mouse. Two features of the Text Editor are especially convenient for typing VHDL code. First, the editor can optionally display different types of VHDL statements in different colors. To turn on this option, open the Options menu and place a check mark next to the item named Syntax Coloring. Second, the editor can automatically indent the text on a new line so that it matches the previous line. To turn on this option, place a check mark beside Options | Auto-indent. Save the file.

Using VHDL Templates

The syntax of VHDL code is sometimes difficult for a designer to remember. To help with this issue, the Text Editor provides a collection of *VHDL templates*. The templates provide examples of various types of VHDL statements, such as an entity declaration, an architecture, and a signal assignment statement. It is worthwhile to browse through the templates by selecting Templates | VHDL Template to become familiar with this resource.

B.3.3 SYNTHESIZING A CIRCUIT FROM THE VHDL CODE

In section 2.8.2 we said that a VHDL compiler generates a logic circuit from VHDL code. The VHDL compiler provided by MAX+plusII is controlled by the Compiler application.

Using the Compiler

Open the Compiler window. As described for the design created with schematic capture earlier, select Processing | Functional SNF Extractor so that the Compiler will generate the information needed to perform functional simulation. Press the Start button in the Compiler window. If the VHDL code has been typed correctly, the Compiler will display a message that says that no errors or warnings were generated.

If the Compiler does not specify zero warnings and zero errors, then at least one mistake was made when typing the VHDL code. In this case the Message Processor window is opened, and it displays a message corresponding to each warning or error found. An example showing how the Message Processor can be used to quickly locate and fix errors in VHDL code is given in section B.3.5. The Compiler window can now be closed.

B.3.4 PERFORMING FUNCTIONAL SIMULATION

Functional simulation of the VHDL code is done in exactly the same way as the simulation described earlier for the design created with schematic capture. Open the Waveform Editor and select File | Save As to save the file with the name *example1.scf*. Following the procedure given in section B.2.4, select Node | Enter Nodes from SNF and import the nodes in the project into the Waveform Editor. Draw the waveforms for inputs $x1$, $x2$, and $x3$ shown in Figure B.17. It is also possible to open the previously drawn waveform file *graphic1.scf* and then "copy and paste" the waveforms for $x1$, $x2$, and $x3$. The procedure for copying waveforms is described in Help | MAX+plusII Table of Contents | Waveform Editor | Procedures | Copying, Cutting & Pasting Nodes and Groups. Open the Simulator and click on the Start button. The waveform generated by the Simulator for the output f should be the same as the waveform in Figure B.19.

B.3.5 USING THE MESSAGE PROCESSOR TO DEBUG VHDL CODE

In section B.2.5 we showed that the Message Processor application can be used to quickly locate and fix errors in a schematic. A similar procedure is available for finding errors in VHDL code. To illustrate this, open the *example1.vhd* file with the Text Editor. In the fourth line, which reads "END example1 ;" delete the semicolon at the end of the statement. Save the *example1.vhd* file and then run the Compiler again. The Compiler generates one error, and the Message Processor window is opened, as illustrated in Figure B.24. The error message specifies that the problem was identified when processing line 6 in the VHDL source code file. Select the error message in the Message Processor window and then click the Locate button. The Text Editor window is automatically displayed with the insertion point at line 6.

Fix the error by reinserting the missing semicolon; then save the file and run the synthesis tools again, to confirm that the error is fixed. We have now completed the introduction to design using VHDL code. Close any open application windows to return to the Manager window.

B.4 DESIGN ENTRY USING TRUTH TABLES

This section describes the process of designing a logic circuit using a truth table. We will implement the truth table shown in Figure B.25. It will be entered into the CAD system by drawing a timing diagram with the Waveform Editor. We discuss the equivalence of truth tables and timing diagrams in section 2.4.1.

We need to specify a new project name for the truth table design. Using File | Project | Name, follow the procedure described in section B.3.1 to assign the name *timing1* to the project. Use the same directory as for the projects designed in the previous sections.

B.4.1 USING THE WAVEFORM EDITOR

Open the Waveform Editor window by selecting MAX+plusII Waveform Editor. The Waveform Editor can be used for multiple purposes. In section B.2.4 the editor was used to create

Figure B.24 The Message Processor window displaying an error in VHDL code.

x_1	x_2	x_3	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Figure B.25 A three-variable function.

input files for simulation. In this section the Waveform Editor will be used to create a different type of file, called a *waveform design file*. To specify the type of file to be created, select File | Save As. In the box labeled File Name, type the name *timing1.wdf*. You must use exactly this name. The name *timing1* must match the name of the project, and the filename extension *wdf* indicates that the waveforms will be used to describe a logic function, instead of being used as simulation input.

B.4.2 CREATING THE TIMING DIAGRAM

To create a timing diagram, it is first necessary to specify the input and output signals for the circuit. Select Node | Insert Node to open the pop-up box shown in Figure B.26. In the box labeled Node Name in Figure B.26, type *x*1. Since *x*1 is an input to the circuit, make

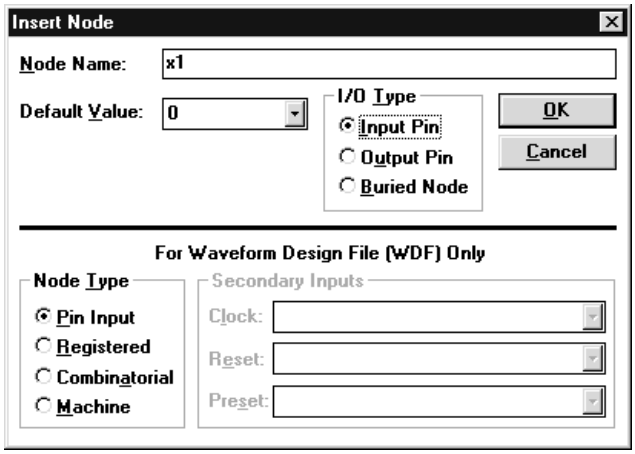


Figure B.26 Inserting a node into the Waveform Editor.

sure that Input Pin is selected in the box labeled I/O Type. Click OK. The input $x1$ appears in the Waveform Editor display. Use the same procedure to insert inputs $x2$ and $x3$ into the Waveform Editor display. Next, select Node | Insert Node again and type f in the Node Name box. Since f is the output for the circuit, make sure that Output Pin is selected in the box labeled I/O Type and then click OK. An alternative way to open the Insert Node pop-up box used above is to double-click in the Waveform Editor display in a blank space under the column labeled Name. The inserted node will be placed in the Waveform Editor window at the location where the mouse was double-clicked.

Having inserted the waveforms into the Waveform Editor, we will now draw a timing diagram to represent the truth table in Figure B.25. Since the truth table has eight rows, we will need to draw eight valuations of the inputs $x1$, $x2$, and $x3$. In section B.2.4 we set the size of the grid displayed in the Waveform Editor to 20 ns. If this same grid size is used, then the total time range needed in the Waveform Display is 160 ns. Select File | End Time and specify *160ns* as the total simulation time. To make the entire time range visible in the waveform display, select View | Fit in Window or type the shortcut command Ctrl+w (while holding down the Ctrl key, press the w key). The Waveform Editor window should now appear as shown in Figure B.27.

Following the procedure described in section B.2.4, modify the waveform for signal $x3$ so that it is 1 for every second 20 ns time range. Also, edit the waveform for $x2$ so that it is 1 for the time ranges from 40 ns to 80 ns and from 120 ns to 160 ns. Finally, set the waveform for $x1$ to 1 in the time range from 80 ns to 160 ns. Previously, when using the Waveform Editor, we did not specify a waveform for the output of the circuit, because the output waveform was generated by the simulator. However, in this case we need to specify a waveform for output f that corresponds to its truth table. In Figure B.25 the function is

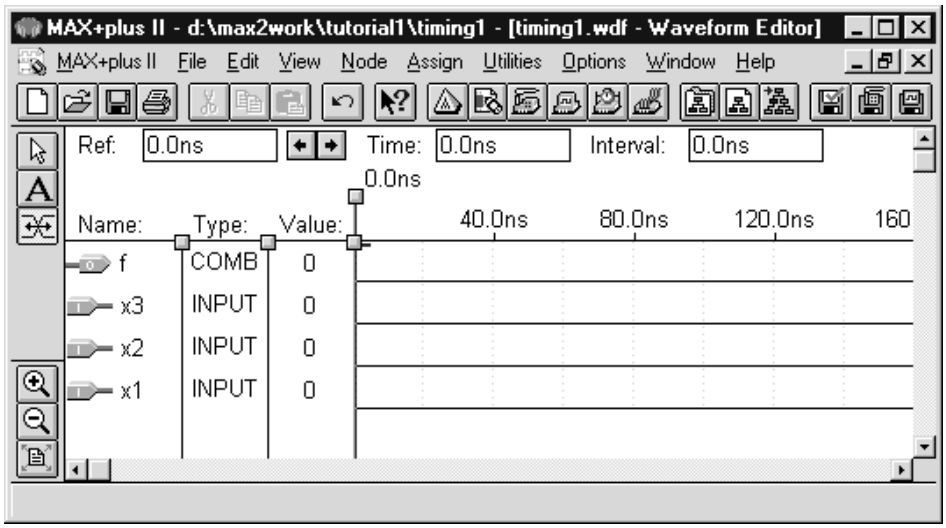


Figure B.27 The Waveform Editor display for the truth-table design.

1 in the rows where $x1$, $x2$, and $x3$ have the valuations 001, 011, 101, 110, and 111. Use the Graphic Editor to change the waveform for f to 1 for the appropriate time ranges. For instance, f should be set to 1 in the time range from 20 ns to 40 ns because this represents the input valuation 001. After completing the waveform for f, the waveform display should appear as shown in Figure B.28. Notice that we have rearranged the waveforms, by moving f to the bottom, in comparison to Figure B.27. Waveforms can be moved by pointing the mouse at the small symbol, called the *node handle*, to the left of the signal name in the waveform display and then dragging the waveform upward or downward. Select File | Save to save the timing diagram in the *timing1.wdf* file.

B.4.3 SYNTHESIZING A CIRCUIT FROM THE WAVEFORMS

The next step is to use the MAX+plusII Compiler to perform the initial synthesis steps for the circuit. The Compiler will generate a Boolean expression to represent f, according to the truth table given by the timing diagram.

Use the same procedure described for the designs created with schematic capture and VHDL code. Open the Compiler window and select Processing | Functional SNF Extractor. Press the Start button in the Compiler window and then click OK in response to the Compiler message that says that no warnings or errors were found.

For the circuits designed in the previous sections, after logic synthesis was completed, the next step performed was functional simulation. It does not make sense to perform the functional simulation for the circuit designed in this section, because the waveforms that would be used as inputs to the simulator would be the same waveforms used to design the

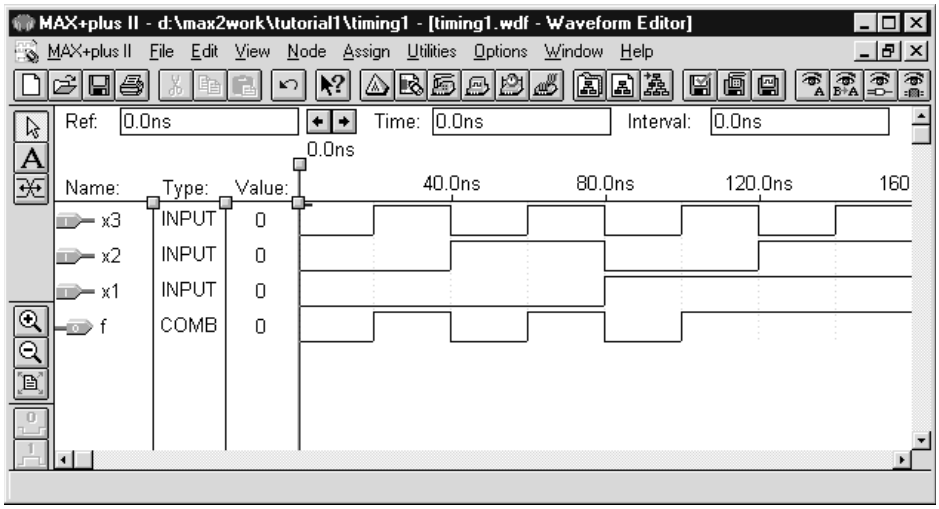


Figure B.28 The timing diagram representing the truth table in Figure B.25.

circuit! In the next section we will use the circuit synthesized from the timing diagram in this section as part of a larger circuit, and we will simulate the operation of the larger circuit.

The tutorial on design with truth tables is now complete, so close any open application windows to return to the Manager window.

B.5 MIXING DESIGN-ENTRY METHODS

It is possible to design a logic circuit using a mixture of design-entry methods. As an example, in this section we will create a schematic that includes the circuit designed using the truth table in the previous section.

We need to specify a new project name for the mixed design. Select File | Project | Name and assign the name *mixed1* to the project. Use the same directory as for the projects designed in the previous sections.

B.5.1 CREATING A SCHEMATIC THAT INCLUDES A TRUTH TABLE

Open the Graphic Editor by selecting MAX+plusII | Graphic Editor. Select File | Save As and, if not already there, type the name *mixed1.gdf* in the File Name box. Make sure to use exactly this name.

Double-click the blank space in the Graphic Editor to open the Enter Symbol pop-up box, as shown in Figure B.29. In the box labeled Symbol Name, type the name *timing1*, which is the name of the circuit designed using a truth table in the previous section. Click OK to import a graphical symbol for the *timing1* circuit into the Graphic Editor. Once

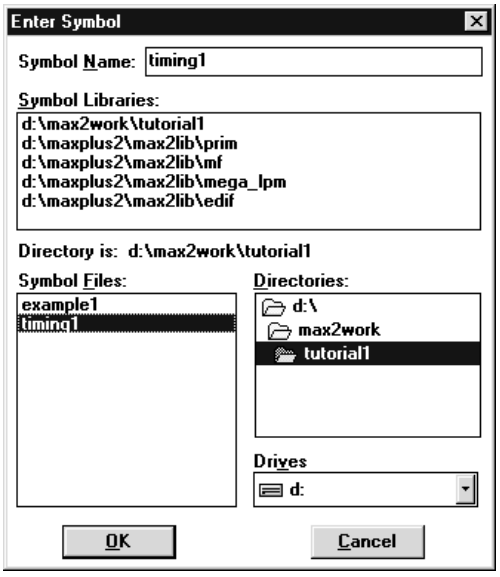


Figure B.29 Importing the truth-table design into the Graphic Editor.

the *timing1* symbol is imported into the Graphic Editor, double-clicking on the symbol automatically opens the Waveform Editor and displays the waveforms that were used to design the circuit. When the Waveform Editor is closed, the Graphic Editor is automatically reopened. This ability to move quickly from one design-entry tool to another is convenient when it is necessary to make changes to a schematic or the subcircuits in it.

Following the procedure described in section B.2.2, import a two-input AND-gate symbol and a NOT gate from the Primitives library into the Graphic Editor. Also from the Primitives library, import three input symbols and an output symbol. Arrange the symbols in the schematic as illustrated in Figure B.30. As described in section B.2.2, assign the names $x1$, $x2$, and $x3$ to the input symbols and assign the name f to the output symbol. The reader will observe that the name $x3$ is used twice in this design project: as an input to the *timing1* subcircuit and as an input to the mixed schematic. The MAX+plusII compiler treats these two nodes named $x3$ as separate nodes because they appear in different levels of the design project hierarchy. Connect the symbols in the schematic together as shown in Figure B.31. Because a wire drawn with the Graphic Editor can be either straight or have a single bend, it is necessary to draw more than one wire for the connection shown in the figure from the AND-gate output to the input labeled $x3$ on the *timing1* subcircuit. Start drawing each wire so that it touches the end of the previously drawn wire; wires that touch are automatically connected by the Graphic Editor. Save the schematic.

B.5.2 SYNTHESIZING AND SIMULATING A CIRCUIT FROM THE SCHEMATIC

Use the procedure described for the designs created in the previous sections to synthesize a circuit from the schematic. The synthesis tools will create a single logic circuit by merging the *timing1* subcircuit with the other logic gates in the schematic. Open the Compiler window, select Processing | Functional SNF Extractor, and then run the Compiler.

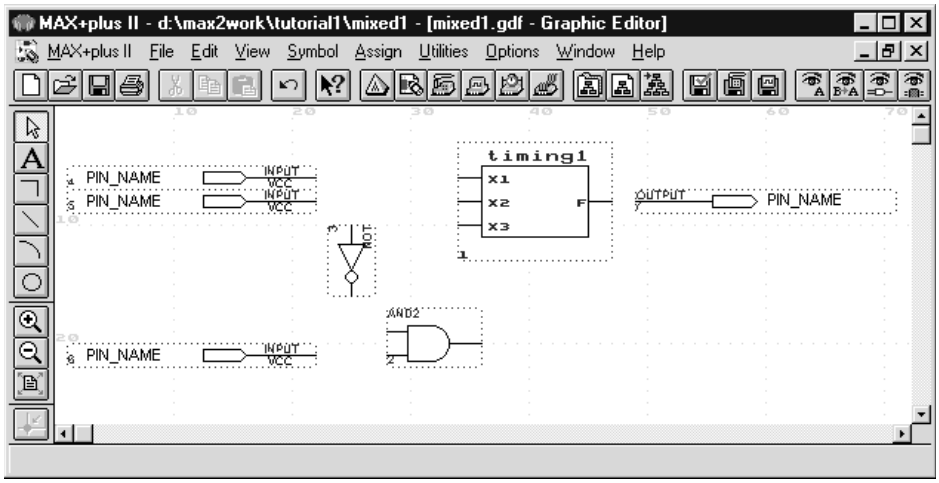


Figure B.30 A schematic including a truth table and logic gates.

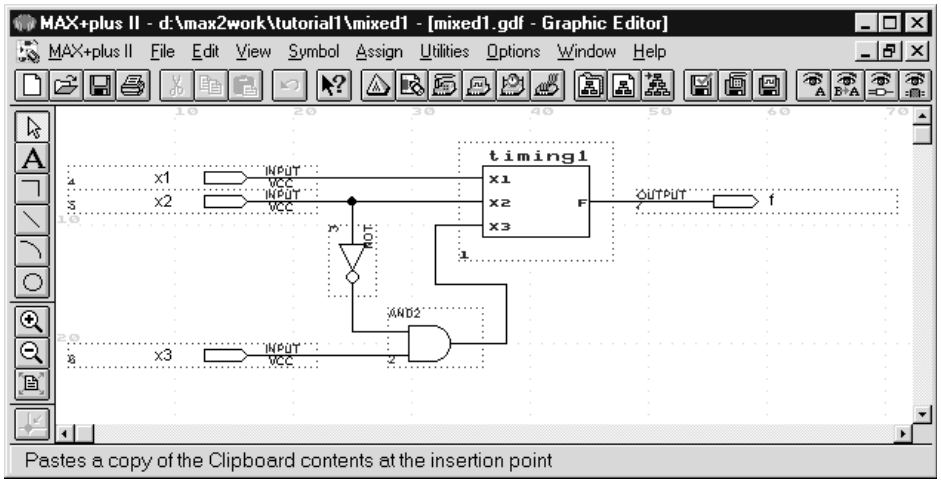


Figure B.31 The completed schematic corresponding to Figure B.30.

Simulation of the *mixed1* project is done in exactly the same way as for the other projects created in this tutorial. Open the Waveform Editor and select File | Save As to create a new file named *mixed1.scf*. Following the procedure given in section B.2.2, import the input and output nodes $x1$, $x2$, $x3$, and f into the Waveform Editor. Draw the waveforms for inputs $x1$, $x2$, and $x3$ that are shown in Figure B.17. Open the Simulator and click on the Start button; then select Open SCF to see the results of the simulation. The waveform generated by the Simulator for the output f should be exactly the same as the waveform shown in Figure B.19. The *mixed1* schematic represents the logic function $f = x_1x_2 + \bar{x}_2x_3$ that was designed using both schematic capture and VHDL code in this tutorial. Techniques that can be used to synthesize the expression for f from the *mixed1* schematic are covered in Chapter 4.

In practice a designer would not use a mixture of design-entry methods for a circuit as simple as our example. The reason that we have created the *mixed1* schematic is simply to illustrate that MAX+plusII allows design-entry methods to be combined in a hierarchical manner. It is also possible, although not shown here, to create a schematic that includes a subcircuit designed using VHDL code. MAX+plusII provides a convenient feature, called the Hierarchy Display, for working with hierarchical design projects.

B.5.3 USING THE HIERARCHY DISPLAY

Select MAX+PlusII | Hierarchy Display to open the Hierarchy Display window shown in Figure B.32. The display shows that the design project consists of two hierarchical levels, with *mixed1* at the higher level and *timing1* at the lower level. The *mixed1* design project has an icon next to it, labeled *gdf*. It can be double-clicked to automatically open the *mixed1.gdf* file in the Graphic Editor. Similarly, *timing1* has an icon next to it, labeled *wdf*. If this icon

Figure B.32 The Hierarchy Display window for the *mixed1* design project.

is double-clicked, the file *timing1.wdf* is opened in the Waveform Editor. Experiment with this method of opening design files. Figure B.32 also shows a small icon labeled *acf*, which represents the *assignment & configuration file* for the project. The file contains settings for a large number of optional features of MAX+plusII that affect the way the design files are processed. These settings are saved automatically in the assignment & configuration file, and so we will not need to modify them manually. Although it is not necessary, the *acf* file can be opened in the Text Editor by double-clicking on its icon in the Hierarchy Display.

B.5.4 CONCLUDING REMARKS

This tutorial has introduced the basic use of the MAX+plusII CAD system. We have shown how to perform design entry by drawing a schematic, writing VHDL code, and drawing a timing diagram that represents a truth table. Each design was processed by the initial synthesis tools and then simulated with the functional simulator.

In the next tutorial we will show how the logic synthesis and physical design tools are used to implement circuits in PLDs. The timing characteristics of the implemented circuits will be examined using timing simulation.