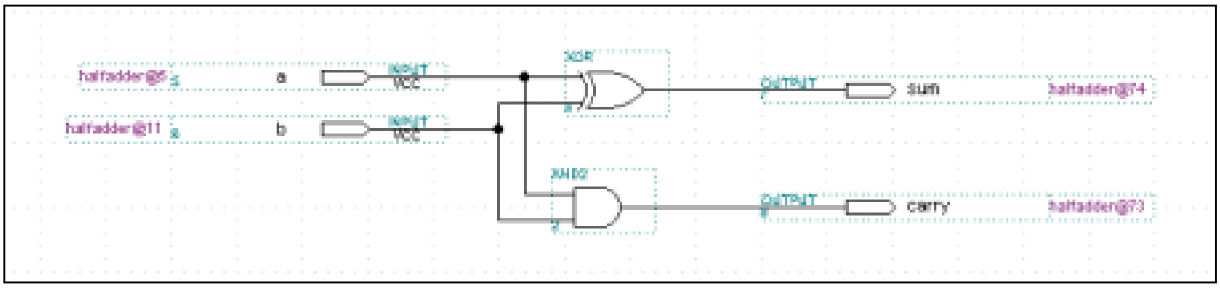
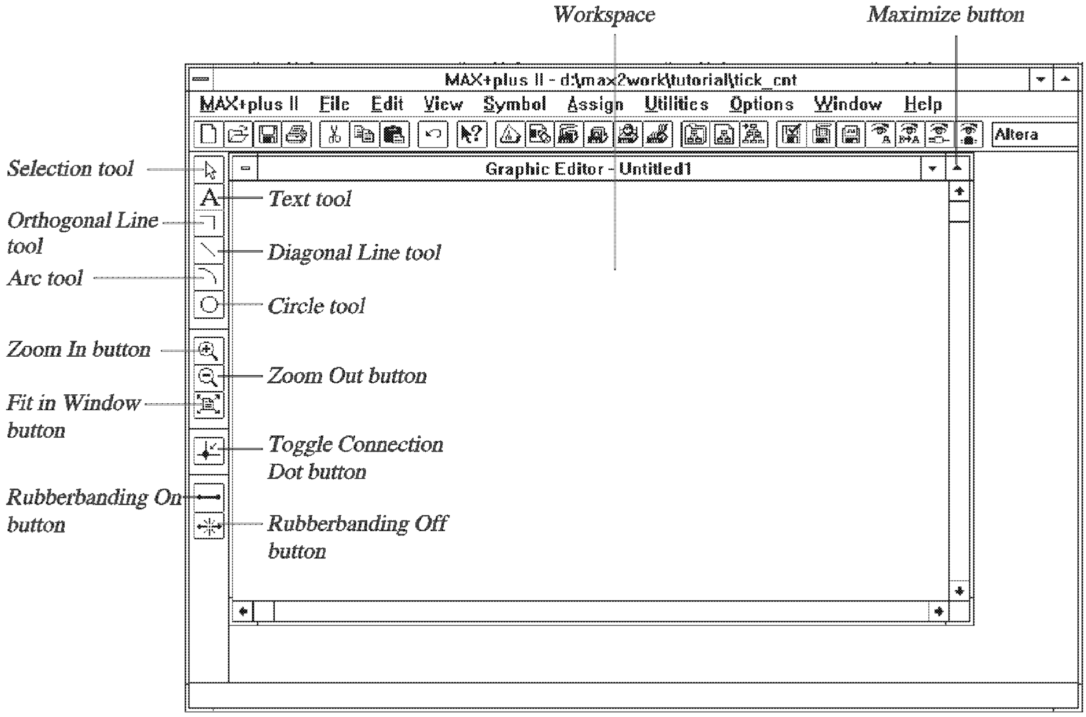
MAX+PLUS II Baseline Software Tutorial

This tutorial is designed to help you become acquainted with the most important functions and features of the MAX+PLUS II Baseline software that you will use during this course. The tutorial describes the steps involved with the design of a half-adder. By the end of this tutorial you will have designed, compiled, and simulated a half-adder. This tutorial will focus on the GDF file ("Graphic Design file") representation of a circuit like the illustration shown below. The original Altera documentation for this software is also available on the courses web site.



1 Creating a new GDF file

- Select NEW under the FILE menu.
- Select *Graphic Editor File*.
- Select *gdf* file type.
- Confirm by clicking *OK*.
- A *Graphic-Editor* window will open. It looks like the illustration on the next page.
- Save the file now by selecting *Save* under the *FILE* menu.
- Enter the desired name. In this case: *halfadder.gdf*.
- Confirm by clicking *OK*.



2 Specify the project name

Before you can compile or simulate your circuit, you must define the current project. In the case of the half-adder, select *Project to Current File* under the *File/Project* menu.

3 Adding functional blocks, components, or other library modules

MAX+PLUS II Baseline has a whole set of common predefined logic blocks that are available to you.

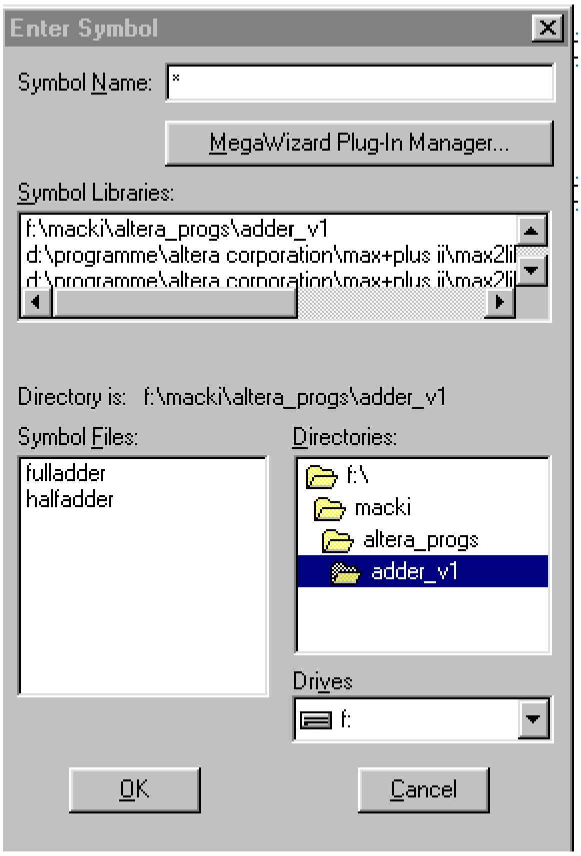
- Double click the left mouse button somewhere in the GDF Editor window where you want the new component to go (an AND2 gate in this case).

- In the window that appears (see following illustration) select and search through the symbol libraries until you find the AND2 gate. Then select it.

- Confirm by clicking *OK*.

- Repeat the previous steps in order to add an XOR gate to your design.

- Later, using this feature, hierarchical circuits can be designed. This is done by first creating a symbol for a completed design (section 10) and then by adding it to the current design as shown above.



Symbol Name: *

Symbol Libraries:
f:\macki\altera_progs\adder_v1
d:\programme\altera corporation\max+plus ii\max2li
d:\programme\altera corporation\max+plus ii\max2li

Directory is: f:\macki\altera_progs\adder_v1

Symbol Files:
fulladder
halfadder

Directories:
f:\
macki
altera_progs
adder_v1

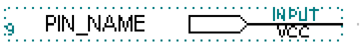
Drives
f:

4 Shift from symbol / gates

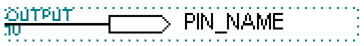
In order to move and align the AND2 gate, you select the symbol with the left mouse button and then click and hold to move it.

5 Adding input and output pins

The procedure is the same as when the AND2 and XOR symbol were added and placed. In one of the libraries there are components named *INPUT* and *OUTPUT*. Add 2 inputs and 2 outputs pins to your design.



INPUT-Symbol



OUTPUT-Symbol

6 Naming the Pins

- Double click on the *PIN_NAME* property of the input pin symbol. This will allow you to change the name.
- Now enter the desired name. In this case the inputs are called *A* and *B*.
- Now repeat the procedure for the output pins (named *SUM* and *CARRY*).

7 Connecting the gates with wires

- When the mouse pointer is over a pin or a terminal point of a design component a wire can be drawn. This is done by first moving the mouse pointer over the terminal point, then by left clicking and holding the mouse button down while you drag the mouse to the other terminal point you want to connect to.

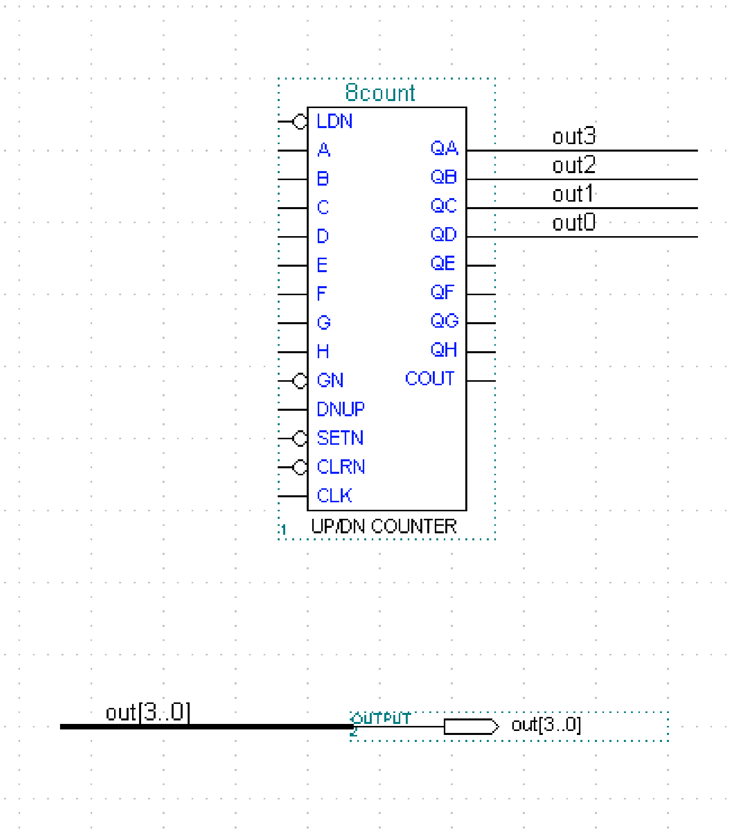
- Select from the *Options/Line Style* menu the highest line style (thinnest one). This represents a 1-bit bus line.

- Repeat the previous steps for all connection wires in order to wire your half-adder correctly.

8 Connecting pins and busses with the use of names

This step is not needed for the construction of the half-adder, however this does help simplify more complicated implementations by increasing the visual clarity of the design.

The following illustration serves only as an example and is not part of the half adder you are creating. It shows an 8-bit counter module that has a 4-bit output. As you can see in the drawing you do not need to connect all the outputs with separate wires. Instead you can give the wires a common name that explicitly implies that they are connected.



In these cases the procedure is as follows:

- Connect the outputs of the counter with short pieces of 1-bit skinny wires (see section 7).

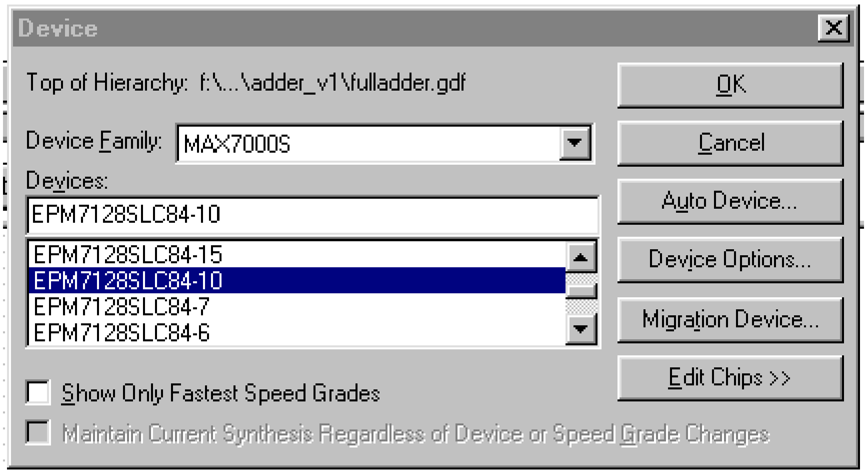
- Draw a second thicker “bus line” segment to the output pin component. The thicker wire is the second line style in the selection list (under *Options/Line Style*).

- Name the necessary pins in sequence by activating the line (left mouse button, activated element is red) and then rename it.

- When you name the busses, you give the name of the bus followed by the appropriate number of bits in square brackets (example: *out[3..0]* for a 4-bit bus with individual output names *out3*, *out2*, *out1* and *out0*). Note that you should indicate the highest order bit first (importantly for the simulation, see section 13).

9 Compile the project

- Before you compile your half-adder, you must select the device (FPGA) with which you would like it to be implemented on. Select the option *Device* under the *Assign* menu option. The following illustration shows the selection window. The FPGA for the course is Alteras EPM7128SLC84-15 (deactivate the *Show Only Fastest Speed Grades* option).



Top of Hierarchy: f:\...\adder_v1\fulladder.gdf

Device Family: MAX7000S

Devices: EPM7128SLC84-10

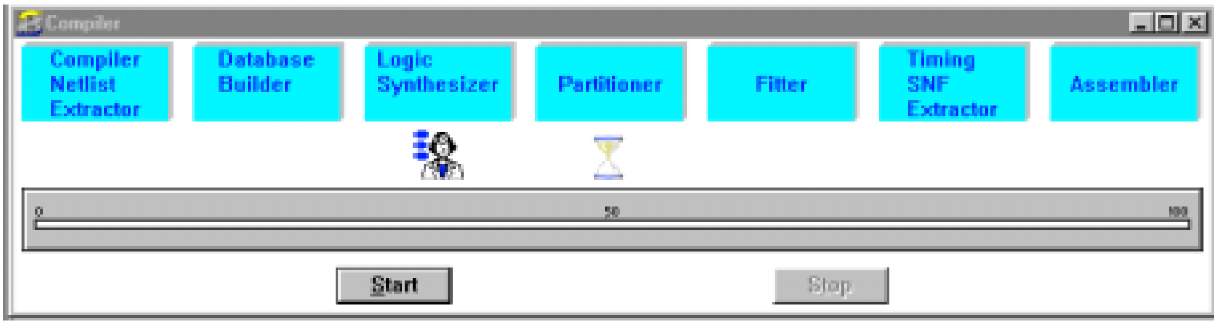
EPM7128SLC84-15
EPM7128SLC84-10
EPM7128SLC84-7
EPM7128SLC84-6

☐ Show Only Fastest Speed Grades

☐ Maintain Current Synthesis Regardless of Device or Speed Grade Changes

- Confirm your selection by clicking *OK*.

- Now select *Compiler* from the *MAX+PLUS II* menu.



- Press the start button to begin the compiling process. This process will also output many informative messages as well as warning or problem messages.

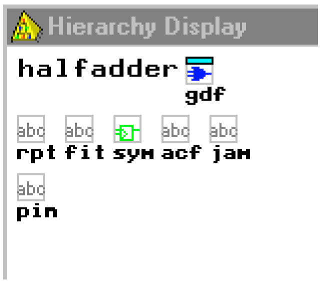
10 Creating a symbol for the design

In order for the circuit you have just compiled to be used again in other designs you must first create a symbol for it. This allows you to generate a hierarchy of circuit designs. This symbol can then be added like any other component in Step 3 of this tutorial.

- Select Option *Create Default Symbol* under the file menu. If a symbol already exists, you are asked whether you want to overwrite it. Normally, you should say *YES*.

11 Viewing the project in the *Hierarchy Display*

- With the *Hierarchy Display* in *MAX+PLUS II* you can see an overview of all the files that are part of this project. The *Hierarchy Display* for the half-adder looks approximately like:



- By double clicking the left mouse button on one of the small icons you can find out the appropriate information about that component.

12 Viewing the generated routing in the *Floorplan Editor*

The *Floorplan Editor* shows you the explicit implementation of your circuit within the selected FPGA and offers two possible opinions: *Device View* and *LAB View*. In the first case all the pins of the FPGA and their logical function are indicated. The second opinion shows the inside of the FPGA (consisting of macro cells, I/O cells, etc.).

- In order to open the *Floorplan Editor*, select it from the MAX+PLUS II menu.

- The two possible opinions discussed above can be selected in the layout menu.

13 Simulation of the Design

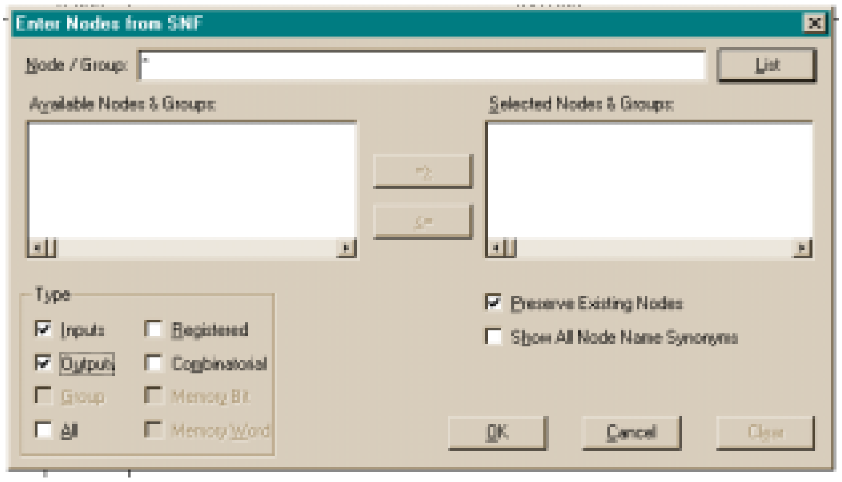
Before the compiled design is programmed into the FPGA, the logical behaviour of the design should be examined through simulation. For this, a so-called *Simulator Channel File* must be generated.

- Select *New* from the *File* menu. Then select *Waveform Editor File* (.scf file). Confirm by clicking *OK*.

- Enter the run length of the simulation by entering the *End Time*. This option is found under the *File* menu. For this test set it to 400ns.

- Under option you can set the *Grid Size*. For this test use 50ns.

- In order to select the inputs and outputs needed for the simulation, select *Enter Nodes from SNF* which is in the *Node* Menu.



Enter Nodes from SNF

Node / Group: *

Available Nodes & Groups:

Selected Nodes & Groups:

Type

☑ Inputs □ Registered

☑ Outputs □ Combinatorial

□ Group □ Memory Bit

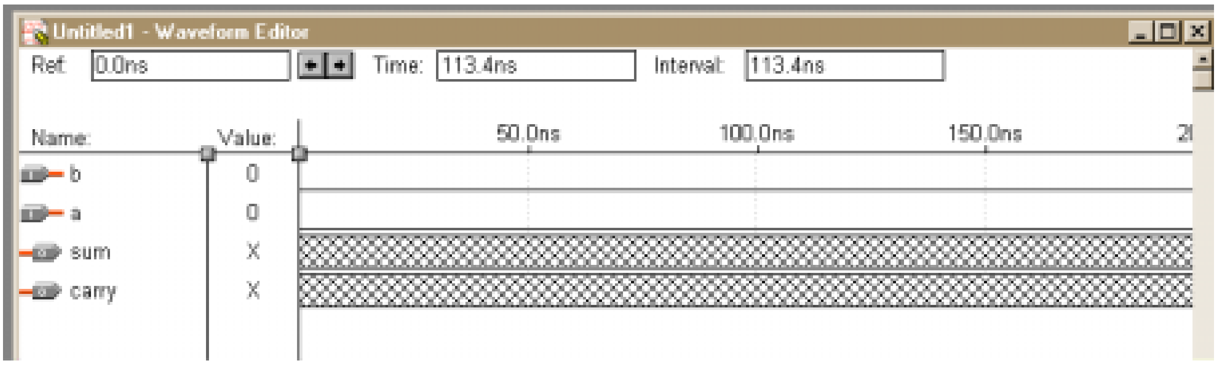
□ All □ Memory Word

☑ Preserve Existing Nodes

□ Show All Node Name Synonyms

- Press the *List* button and select all the pins of the half-adder.

- After clicking *OK* the SCF file opens and it should have the following appearance:



Untitled1 - Waveform Editor

Ref: 0.0ns Time: 113.4ns Interval: 113.4ns

Name:	Value:
b	0
a	0
sum	X
carry	X

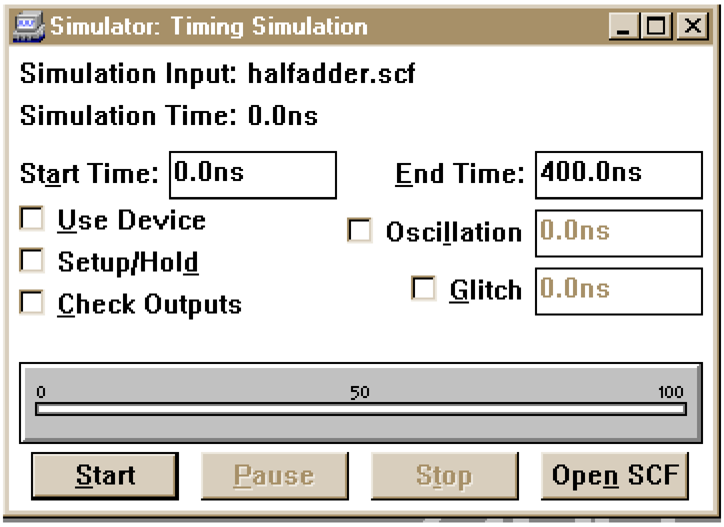
- Save the simulation file under the name *halfadder.scf*.

- In order to simulate the four possible combinations of the inputs *A* and *B* of the half-adder, proceed as follows:

  1. Select the time interval (for example 0 to 100ns) by using the left mouse button.
  2. Press the right mouse button and select under *Overwrite* the desired value of the signal, for example High (1).
  3. Continue the procedure, until all possible value combinations of the entrances are represented.

- Save the changes.

- Select *Simulator* from the *MAX+PLUS II* Menu.



Simulation Input: halfadder.scf

Simulation Time: 0.0ns

Start Time: 0.0ns

End Time: 400.0ns

☐ Use Device

☐ Setup/Hold

☐ Check Outputs

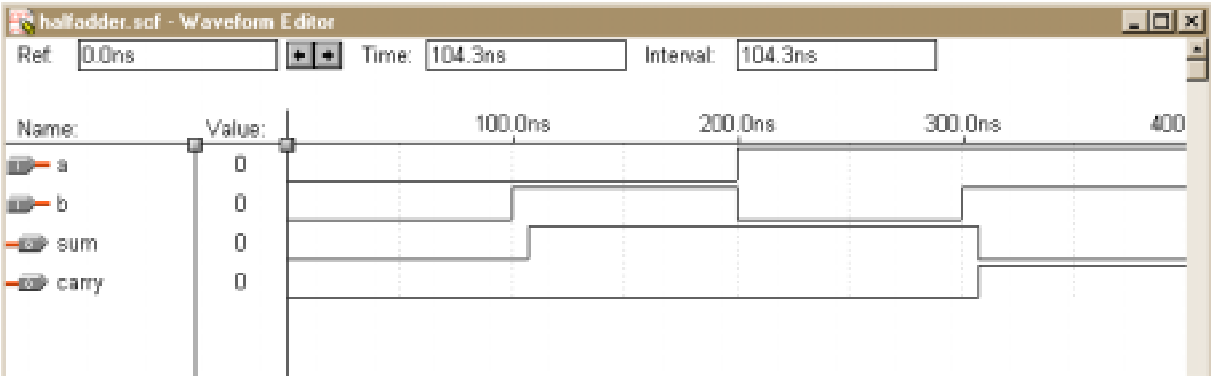
☐ Oscillation 0.0ns

☐ Glitch 0.0ns

- If you would like the simulation to consider the setup and hold times of the various gates you can set the *Setup/Hold* option.

- Start the simulation by pressing the *Start* button.

- If all is correct the simulation of the half-adder should provide results similar to those shown below.



Ref: 0.0ns

Time: 104.3ns

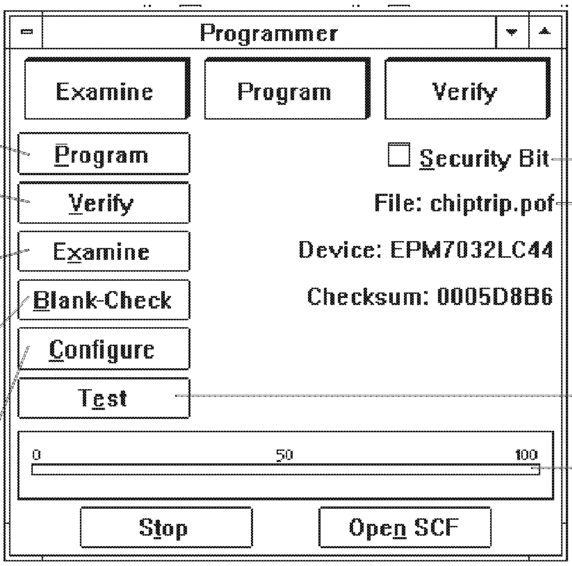
Interval: 104.3ns

Name:	Value:
a	0
b	0
sum	0
carry	0

14 Programming the FPGA

After successfully compiling and simulating your half-adder the selected FPGA can now be programmed.

- Select the *Programmer* Option under the MAX+PLUS II menu to start the Programmer. It should look like the illustration below.



□ Security Bit

File: chiptrip.pof

Device: EPM7032LC44

Checksum: 0005D8B6

Programs data from a programming file or examined device onto a blank device.

Verifies contents of a device against current programming data.

Examines a device and stores the data in a temporary buffer.

Examines a device to ensure it is blank.

Downloads configuration data into a FLEX device.

Turns on Security Bit option for current programming file.

Displays current programming file for the project.

Functionally tests a programmed device with input vectors from current SCF, Vector File, or programming file.

Progress bar