

Mentor Graphics Tutorial 4

Using SmartmodelTM Programmable Devices with Quicksim II

Created by: Dr. Khurram Waheed, San Diego State University

Last modified: Spring 2004

1 Objective

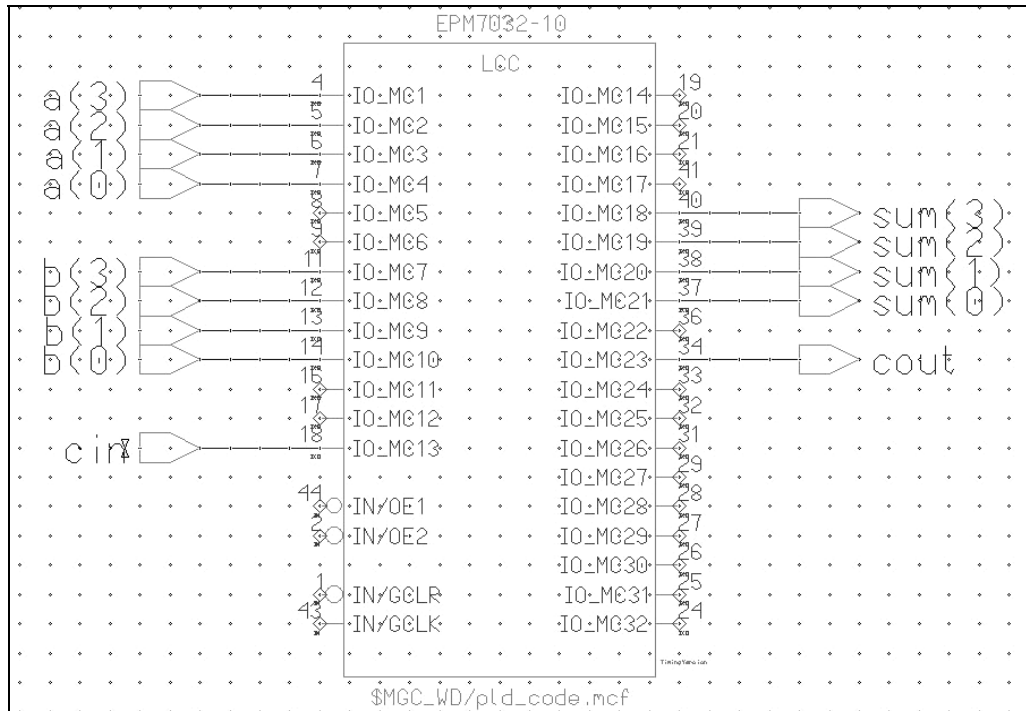
The purpose of this tutorial is to demonstrate the programmable logic device simulation capabilities of Mentor Graphics Quicksim II simulator using the Synopsys SmartmodelTM Libraries. Although the tutorial uses an Altera CPLD with the VHDL source synthesized using the Altera Max+PlusII software, the procedure is similar for other families (including Xilinx, Actel, etc.) of programmable logic (i.e., both CPLDs and FPGAs) provided the user can generate the right interface files required by the Smartmodel + Quicksim-II pair. In all cases, the smartmodel library PLD/CPLD/FPGA module is used in the Design Architect schematic as a regular digital device, where the programming file is specified via the Quicksim Interface file (either in .scf or .mcf format).

For this tutorial, we will employ an **ALTERA MAX7000** CPLD device, the EDIF 3.0.0 file will be generated using the Altera Max+PlusII software and the pins used in the Design Architect are applied as constraints. Note the tutorial will only describe the use of Altera software for the appropriate EDIF file generation. The VHDL code may be simulated using Modelsim, Xilinx ISE or Altera's software. For more info on these software see the class website.

2 Schematics Entry in Design Architect

1. Setup your \$MGC_WD environment label to your work directory.
2. Start Design Architect to create your schematics. You may do so via the Design manager, i.e., start Mentor Graphics' design Manager by typing "**dmgr &**". Now from the Tools window choose the Design Architect, or you may type "**da &**" at the command prompt to start the schematic entry directly.
3. In the design architect, choose *File* → *Open* → *Sheet*. In the pop-up window, enter an appropriate schematic name, e.g., **Tutorial4** in the component name window, do not change the sheet name which should be **sheet1**. Continue as earlier tutorials and draw the schematics shown below following the instructions outlined below.

Note: "**Click**" means use the left mouse button to select something.



4. To get the model for **EPM7032-10** do the following in Design Architect:
 - a. *Libraries* → *Logic Modeling Smart Model Library*
 - b. In the libraries palette on the right side, choose: *Programmable Logic* → *CPLD* → *Altera* → *7032 High-Performance 32-Macrocell Device* → *EPM7032-10* or you can choose another device with a different speed grade.
5. Once you have the model placed in your schematic, press F2 to make sure that everything is unselected, then click on the CPLD model to select it, then press the right mouse button to bring up a floating menu called Instance (use "Other menus" if necessary to find Instance). Choose *Properties* → *Modify...* to get the "Modify Properties" requester. Click on "SCFFile =" and click *OK*.
6. Another requester will come up. Fill in the "Value" box with the complete pathname where you have put your .jed file for the PAL20R6 that you are using. In this example, pld_code.mcf will be used, which we will place in the \$MGC_WD. So you may type *\$MGC_WD/pld_code.mcf*. Alternately you may give the complete path to where you will place this interface file.
7. Now you may connect the wires as shown above or you may **BUS Bundles**.
 - a. To use bus **bundles** in Design Architect, first set them up: *Setup* → *Ripper...* → *< set ripper mode to AUTO >* → *OK*
 - b. To lay down a bus do shift-F3. Busses should have a section which is perpendicular to the direction of connection to it. Click to start a bus; click to change direction (if desired); double-click to terminate the bus.
 - c. To make individual connections to a bus press F3 to lay down wire connections.

- d. Do not start from the bus. Start from a device pin. Click on most significant pin (e.g., if a bus is called $a(3:0)$, then $a3$ would be MSB), then double-click on the bus (the bus section and the wire you are laying down must be at 90 degrees to each other). A 45-degree “Ripper” connection is automatically placed for you connecting your wire to the bus, and a requester comes up asking you to name the bus and also to provide the index number for this wire.
 - e. For example, you will enter “ $a(3:0)$ ” for the bus name (don't type in the quotation marks). The “3” is the MSB, so enter a “3” for the Ripper Rule (this is the index number; do not try to enter $a3$ or $a(3)$ or anything like that -- it only wants the index number). If it does not accept the “3” it probably means that you forgot to set ripper mode to auto. When you click *OK*, you will find the “ $a(3:0)$ ” stuck to your mouse pointer. Move the mouse to where you want the “ $a(3:0)$ ” to be placed on the schematic (somewhere near the bus but not interfering with anything else), then click to place it.
 - f. Repeat the process by connecting the next pin to the bus. From now on the requester will only ask you for the index number.
8. Produce the input bus $b(3:0)$ and connect the output $sum(3:0)$ in the same way. Remember never start a wire connection to a bus by clicking first on the bus!. Complete the schematics as shown above. Verify its correctness using *Check* → *Schematics*. Now save the schematics using *File* → *Save (default registration)*.

3 Generating the PLD Programming File

1. For this tutorial we will assume that the VHDL file has been created, edited for any errors and simulated. We will place this file in a subfolder say *VHDL_src*.
2. Now navigate to this folder and type “**max2win &**”. The first time you start **max+plus II** in your account it will take sometime enumerating fonts. This is a one time event so be patients and let it finish.
3. Once the font-caching is done the max+plus II environment will come up.
4. Select *File* → *Open* and open your VHDL source file. The VHDL source for this tutorial (**full_add.vhd**) is given below

```

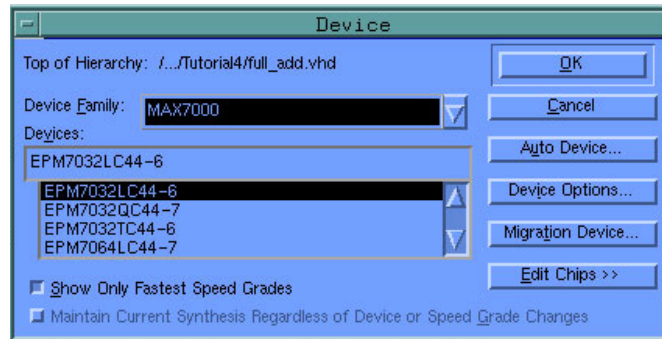
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY full_add IS
PORT(
  a: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  b: IN STD_LOGIC_VECTOR(3 DOWNTO 0);
  cin: IN STD_LOGIC;
  sum: OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
  cout : OUT STD_LOGIC
);
END full_add;

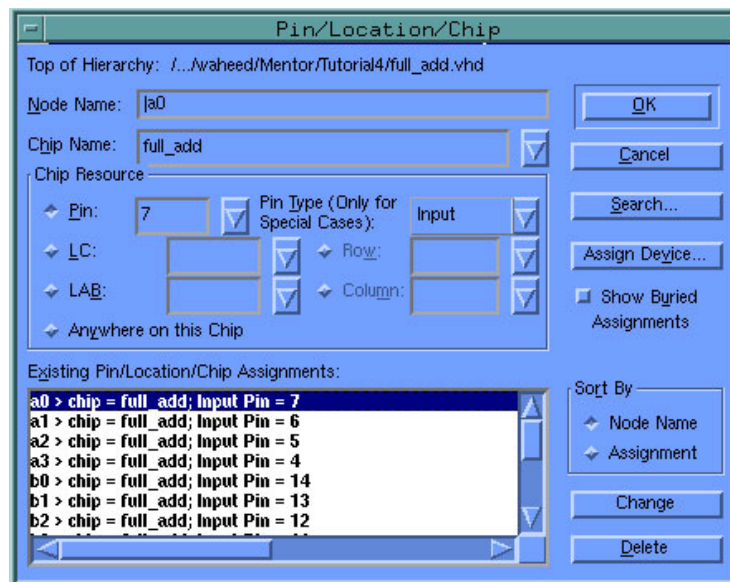
architecture behavior of full_add is
begin
  P1:process(a,b,cin)
  variable c: STD_LOGIC_VECTOR(4 DOWNTO 0);
  begin
    c(0) := cin;
    for j in 0 to 3 loop
      sum(j)<=(a(j) xor b(j) xor c(j));
      c(j+1):=(a(j) and b(j)) or (a(j) and c(j)) or (b(j) and c(j));
    end loop;
    cout<=c(4);
  end process P1;
end behavior;

```

- Now *Select File* → *Project* → *Set project to current file*. Note that for Altera's max+plus II software the name of the entity should be the same as your VHDL file.
- Now we need to select the appropriate device for our code.
- Select *Assign* → *Device*, and choose the device family **MAX7000** and the device **EPM7032LC44-6**.




- Now we need to constrain the pin locations for the device so that the signal assignment matches out schematics. In order to do this select *Assign* → *Pin/Location/Chip*. A new pop-up window will come up. Assign the appropriate package pins as shown below



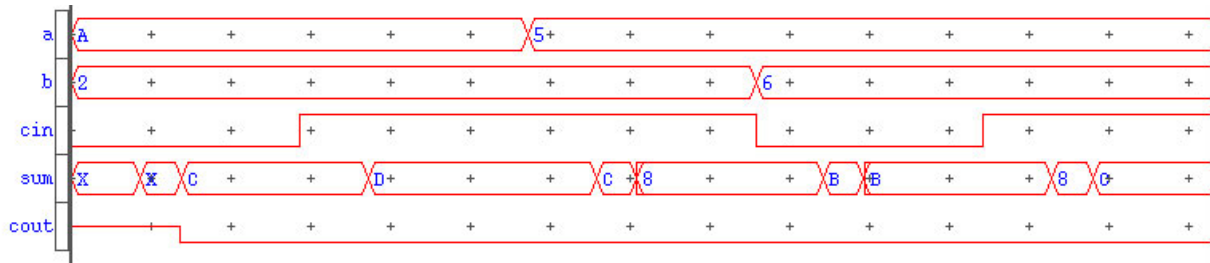
- All the pin assignments must match the schematics above. In case you need to correct any pin assignment, simply click on it in the *Existing pin/Location/Chip Assignments*, modify the contents and select **Change** to update the pin assignment.

Note that in general it is not a good idea to constrain the pins of the device before you actually fit your VHDL code into the device. Typically the procedure is first to write your programmable VHDL code, select a suitable CPLD/FPGA device and let the compiler/synthesizer decide on the best locations for the pin assignments. You can see the pin assignment in the report file generated by the fitter/synthesizer. This device assignment can be adopted for the schematic drawing if found satisfactory. Otherwise, some of the pins may be moved around for better schematic pin assignments. Sometimes, the design might fail to fit in a particular programmable device if the pin assignment becomes restrictive.

10. The design is now ready to compile, choose *Max+Plus II* → *Compile* or choose the compile icon, which is like a furnace on a chip . A Compile Window will pop-up and the menu options will change as well.
11. Click on the menu item *Interface* → *EDIF Netlist Writer Settings*, select *Vendor* to be **Synopsys** and the *EDIF version* to be **EDIF 3.0.0**, and select **OK**.
12. Now select the *Interface* → *EDIF Netlist Writer*, i.e., ensure that a checkmark appears next to it.
13. Now select **Start** in the Compiler Window. In case there are any error messages, fix them and run the process again. If everything is successful, you have generated the correct files needed. Click on the report file icon below **Fitter**. All the pin assignments should match your assignments in the earlier steps.
14. Your project directory should now have the compiled EDIF file named **full_add.edo**
15. Copy this file to your schematics directory.

4 Simulation using Quicksim II

1. Before starting Quicksim, we need to perform some preparation steps, such as
 - a. create the **p1d_code.mcf** file we specified in the schematics. To do this start a text editor of your choice and create the file with the following text.
Syntax: `load -source <source file> -g -m <model name> -f <device family name>`
load -source /home/student/..<your_path>/full_add.edo -g -m epm7032 -family altera
 - b. Now we need to compile the edif file to a format which is understandable to the Synopsys Smartmodel component we placed in our schematics. This is done using the Synopsys' **smartccn** compiler. At the unix command line type
Syntax: `smartccn <source file> -m <model name>`
smartccn full_add.edo -m epm7032
 - c. The above step, upon successful completion will create a file **full_add.ccn** file in your work directory.
 - d. Note that you can see the list of available components and models in Smartmodels by typing the "**sl_browser &**" at the unix prompt.
 - e. Now you are ready for the quicksim II simulation.
2. Start Quicksim II by either using the design manager by right clicking on the schematics in the design manager \$MGC_WD window and choosing Quicksim II or else you may type "**quicksim Tutorial4 &**" at the unix prompt.
3. While entering Quicksim II do the following: Check for the Model Messages box that will tell you if there were any errors. If there are, you will have to go back to Design Architect and correct the problem.
4. Using the force file apply the 4-bit inputs to the circuit and simulate the CPLD based circuit. A sample of simulated results are shown below

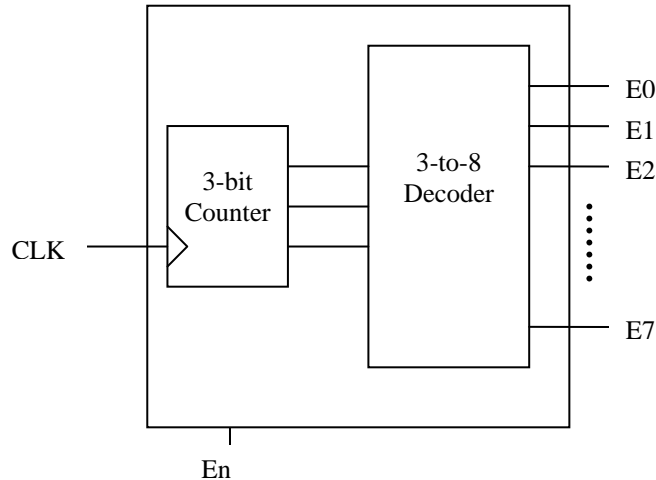


Important Information about Quicksim II Simulators

- Choose *Setup* → *Kernel* → *Analysis...* : In the requester, click on *Visible*, change *Timing Mode* to *MAX*, change *Constraint Mode* to *MESSAGES*. Also click on *Contention Check* and *Model Messages*, then *OK*. This enables a special feature of the Logic Modeling devices. During any simulation you will get messages whenever any setup or hold time, frequency limit, or pulse width specs are violated.
- To see this in action run a simulation with a very small CLK period, like 5-15 ns, and you will see the Model Messages box filled with complaints.
- When defining forces, select *Fixed* or *Wired* forces for bidirectional I/O lines, otherwise your logic level input will be seen by Quicksim II as clashing with the high impedance level programmed for an output.
- To define a *CLK* waveform using the menu *Setup* → *Force* do the following. Use *F2* to unselect all signals then click on *CLK* in the Trace window. In the palette area on the right side, click on *STIMULUS* to get this palette up, then click on the *ADD CLOCK* square button, which will bring up a requester. Enter the clock period in nanoseconds(ns) in the “*Period*” box, click *OK*. Proceed with the other forces as in the previous tutorials.
- While comparing the schematics and the PLD models don't worry that the inputs and outputs are named differently. The CPLD/FPGA program information in EDIF (.edo) file does not use any signal names. However, bus ripper index numbers in Design Architect must be numerical.

PROJECT 4

Using a Max 7000 series CPLD in Quicksim, design and simulate a digital circuit which performs the operation described in the block diagram below



- Implement the above circuitry in two different speed grades of the same CPLD device, e.g., use EPM7032-1 and EPM7032-15.
- Set the device delays to maximum. Determine the maximum CLK frequency at which the device works reliably.
- What is the relationship of the device speed grade to this maximum operable frequency? Please tabulate and answer in your report.

5 Grading

This tutorial and the project comprise of 50 points, i.e.,

Tutorial	25 points
Exercise	25 points

6 Report Format

Submit the completed tutorial and the exercise in the following format

1. Title Page
2. Introduction
3. Tutorial
 - Design Description
 - Schematics/VHDL design
 - Force File
 - Simulation Results
 - Comments
4. Exercise
 - Design Description
 - Schematics/VHDL design
 - Force File
 - Simulation Results (with maximum delays) for both speed grades
 - Comments & Observations
5. Conclusions