Altera MAX+PLUS II and FPGA Chips

The Altera MAX+PLUS II software is a comprehensive tool for the design, compilation, and simulation of logic circuit designs, and for loading completed designs into an Altera CPLD chip. In the CS152A/EE116L laboratory we use the Altera UP-1 board, and all lab experiments are designed to be done with the EPM128SLC84-7 chip, which has about 2500 gates. These notes are designed to aid in getting to know the software package.

A tutorial of software can be found on the Altera Literature page (http://www.altera.com/literature/lit-mp2.html). The software also contains extensive HELP files.

A good introduction to the Altera UP-1 board and MAX+PLUS II is the book:

> James O. Hamblen & Michael D. Furman, "*Rapid Prototyping of Digital Systems*," Kluwer Academic Publishing, 2000

MAX+PLUS II is distributed on a CD with the textbook used in CS 51A:

> M. Ercegovac et al: "*Introduction to Digital Systems*," John Wiley, 1999

General Description

The MAX+PLUS II software may be used to design logic systems. There are three modes for preparing designs:

- Graphic Design Files (*.gdf);
- Text Design Files (*.tdf);
- VHDL files.

GDF is a way of drawing a whole logic system with predefined symbols, such as AND/OR/XOR gates, multiplexers, flip-flops, etc. The design will look similar to circuit diagrams that are in your Logic Design textbook.

TDF is a way of describing the logic of a design as text, such as boolean equations and state transition tables.

VHDL files use VHDL statements to describe designs. An introduction to VHDL is in Ercegovac's book.

In the Laboratory, we will use GDF and TDF files. These are easier to use than VHDL files. However, VHDL is a more widely used hardware documentation language.

We ask that you follow the following process. **Keep your designs on a floppy disk**. MAX+PLUS II works slowly with A: drive files. On the workstations Disk C: holds the MAX+PLUS II software. Do not make any subdirectories on Disk C. We may periodically revise the design software, and thus destroy whatever you put on C: Copy your files to the disk D, or copy them to your private subdirectory under "users

on the server" for improved security. The files your save on the server are usually safe, but still, the safest place for your files is on your floppy disk. You will need to save only the *.gdf, *.tdf and *.scf files; all your work can be reconstructed from these files.

Getting Started

1. Define the Project

A project is a set of files that define your design. You need to give a unique name to your project. Your designs will be *.gdf, or/and *.tdf files. You will also need *.scf simulation files after the design is completed, ready to be checked via simulation.

When you start MAX+PLUS II, select "File"; then "New" from the menu options. In the next menu choose either "Graphic Editor file" or "Text Editor file", depending on what design method you wish to use.

When a blank design screen or a empty text editing windows shows, define your file by selecting "File", then "Save As", then type in the file name.

To define your project select from the MAX+PLUS II menu, select "File", then "Project" then "Set Project to Current File". Your project name should now appear on the top line of the MAX+PLUS II Project Manager. You now have a project, and a file in which you may enter your design.

You can also open an existing design file, or set a project by type in the correct filename.

Note that the menus are context sensitive: appropriate messages will appear in different uses.

Your design may be a mixture of *.gdf, *.tdf files. The compiler automatically extracts connection and component information, and makes intermediate files from which the circuit is constructed. The first step is circuit connection extraction (*.cnf file) either from a *.gdf or a *.tdf design file. Same file name is used as the gdf or tdf file. Therefore you may NOT use the same name for a *.gdf and a *.tdf file in the same subdirectory. Make your *.gdf and *.tdf file names distinct.

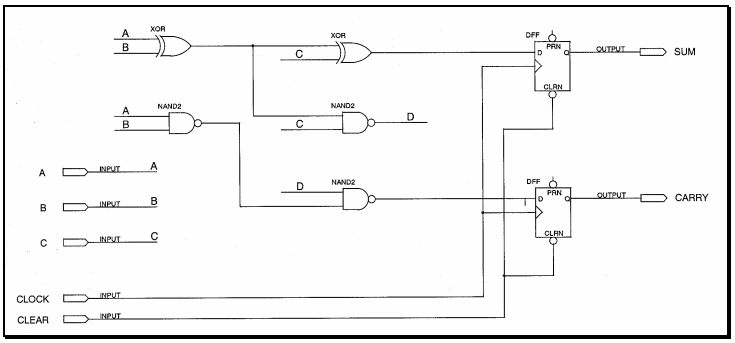
Your design can be constructed as a hierarchy of gdf or tdf files, with a design in one file being used as a component in a higher level file. You can check the structure using the "Hierarchy Display" tool. Your project should be set to the top level design file.

2. Do Your Design

A. Draw your design with the graphic tool

Use the graphic design tool to assemble and interconnect your circuit. This will involve:

- Placement of components (such as DFFs and basic gates, 74xx chips) by pointing at a location, double clicking, then enter the symbols;
- Interconnection of components is done with routing lines between component terminals/pins OR by labeled lines; Lines with same label are connected logically.
- Input/output connectors must be used and must be labeled with a unique name;
- Documentation is done by typing text to indicate information. You may use title from the components menu to get a standard drawing title block.



The design tool includes several libraries from which standard design components may be used (under "c:\maxplus2\max2lib"). You may select them form the list when you "enter a symbol". These include:

- **prim** files that define logic element primitives, such as basic gates (AND2, NOR3, NOT…), input/output terminals, flip-flops, and etc. Look at the "prim" subdirectory.
- **mf** files that have logic metafunctions such as registers, counters, and equivalents of many 74XX chips. Note that when you enter a mf component, such as 74164, you do not get that component, but a MAX7000S logical equivalent that has different dynamic behavior (rise/Fall/Delay times and input/output capabilities and requirements.)
- **mega_lpm** files that contain larger logic functions, such as dividers and FFTs.

The libraries contain graphical equivalents ("Symbols" for use in .GDF designs) and text equivalent ("include files" for use in .TDF designs). Use these library components; they are likely to have fewer errors than files/symbols you may create for the same functions. Help files exist for the components.

B. Write AHDL program

AHDL programs (tdf files) are plain text files; they can be edited with any text editor, such as the editor in MAX+PLUS II.

3. Compile your design

When your design is completed, you need to compile it for the chip on the Altera Board. First you need to set the proper device assignment. Open the compiler, select

"Assign" on menu, then 'Device', then select "**MAX7000S, EPM7128SLC84-7**".You may need to uncheck the "Show fastest speed grade only" box.

After you make the assignment, click "start" to compile. You may have to correct context errors in your source file (*.gdf or *.tdf). A successful compilation is the first step in verifying your design.

The compiler will generate a series of files for the project, including a *.rpt report files that describes the resource assignments for the project. You may want to check it for the pin assignments for the CPLD chip.

4. Simulate your design

A. Define a simulation waveform file (*.scf)

Once an error-free compilation is achieved, you need to step your design through a sequence of inputs to verify its design. Select "File", then "New", then "Waveform Editor file". A blank waveform display will appear. You may have multiple .scf files for one project, but then you have to link them to the project when you want to do simulation with them. A simple ways is to give the scf file same name as the project.

B. Define input and outputs

You need to enter inputs and outputs on the left side of the display. Double-click in the area where the names are to be typed; a new menu will appear. One by one you may type in the names for the input/output signals (and click on the I or O option). Alternately you may click on the "List" option: a list of the variables including inputs and outputs you have defined for your design will automatically appear in the lower left of the menu. Select all your signals. The appropriate Input or Output option will be checked.

C. Define your grid size

The simulator will allow you to define inputs (including CLOCK) on a grid interval. You may set this value over a wide range of values. Keep in mind the following:

- The devices you use have delay times indicated by the last number of the part number, following the "dash", in our case, 7 nanoseconds. You should allow at least that much time FOR EACH LEVEL of logic gating you use. Additional time (typically 2 ns) will be needed to propagate signals between internal logic gates. A grid size of 20ns is typical.

- When you use pre-defined parts (such as multiplexers and adders), it is not obvious how many levels of logic gating will be involved. In that case it is best to use the Timing Analyzer to find out. The maximum delay for combinational parts of the design will be calculated. Make your grid interval at least as much as the largest number calculated by the analyzer.

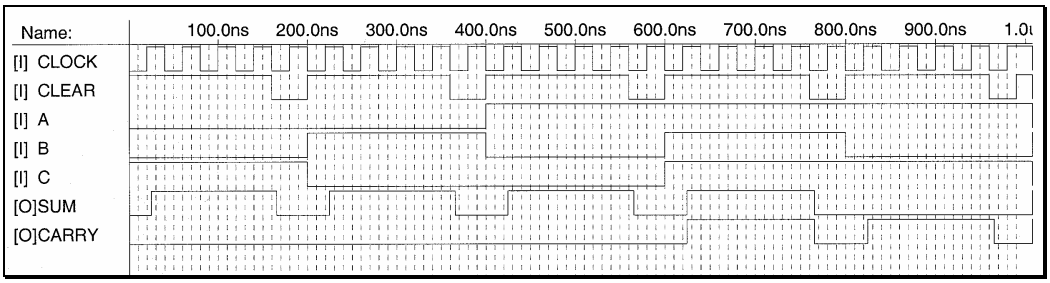
Set your Grid Size. Select “Options”, then “Grid Size”, then type in your grid size. Note that clock periods can not be made smaller than *TWICE* the grid size. Clock periods can be made integer multiples of twice the grid size. You can check for maximum circuit delays, including “register delays” using the Timing Analyzer.

D. Define the simulation interval

The default simulation interval is 1μs. If that is not enough, you can adjust it by selecting “File”, then “End Time”.

E. Run a Simulation

You need to define all inputs for your design, *including clock(s)*. Drag through a desired interval on a signal, you can then click the buttons on the left bar to set the binary value for the signal. Grouped signals can be viewed and set with representations other than binary values. Use the button with a clock icon for defining clock signal.



After your have defined all the inputs, run the simulation. The simulator will calculate the circuit response (including circuit delays) for the output pins. These will automatically be shown on the Waveform displays.

F. For larger designs (such as Lab6) you need to restrain your compiler option to work only with your chip (MAX7000 series). Select “Assign”, then “Global Project Logic Synthesis”, then “Multi-Level Synthesis for MAX5000/7000 Device”, then “OK”.

You may not need this Assignment until the final Lab Project.

5. Test on the CPLD chip

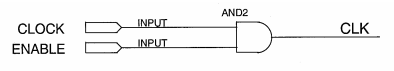
A. “CLOCK” pin

The Altera MAX7128SLC84-7 chip has an internal system clock, running at 25.175 MHz. This clock signal is hardwired to ping 83 of the chip. To use your own clean clock signal generated manually, you must avoid using pin 83 for the clock signals.

The compiler can automatically detect the global clock signal in your design and assign it to pin 83. To change this assignment, you can use the “Floorplan Editor”. If the clock signal is assigned to pin 83, click on it and change the pin assignment to

another pin (usually pin 2), and recompile the project. Now the clock signal is assigned to another pin.

Another method is to feed your clock input into a combinational circuit before connect it with the clock connection of each component. An AND gate and an "Enable" input signal will be enough:



The compiler then won't take "CLOCK" as a global clock signal and assign it to pin 83.

B. Download design to the chip

Connect the JTAG interface with the parallel port of the computer. Turn the power on. Open the "Programmer" to program the ELPD chip with your design. Make sure the correct project name shows on the Programmer window.

C. Connect inputs/outputs and test

Connect the input/output signals to the switches/LEDs on the UP-1 board so you can test the circuit. Check the rpt file for pin assignment. You only need to connect all the pins assigned to the input/output signals you have defined in your design.

Now your can test your circuit physically.

The FPGA Chip

A Field Programmable Gate Array basically is made up of a large number of gates, typically multiplexers and Flip-Flops and circuits that allow a nearly-arbitrary connection of these components. The actual circuit connections are in memory cells on the chip, and therefore the components on the chip can be reconnected to perform other functions as needed.

On the UP- I board there are two Altera FPGA chips: a MAX7128SLC84-7 and a FLEX EPFIOK20RC240-4 chip. The MAX chip has about 2,500 gates in it, and the FLEX chip has about 20,000 gates. Use of these chips is often called a "sea of gates" approach to hardware design. The device is a Complex Programmable Logic Device (CPLD). The gates are photolithographically put on a silicon wafer, less than ',4 inch square for the MAX chip. Every logic design can be implemented as a state machine using simple logic gates: basically AND/OR circuits feeding the control terminals of Flip-flops to set up the Next States and the Flip-flop outputs supplying the Present State information. Typically the AND/OR (or the equivalent OR/AND) gates are replaced by NAND/NAND circuits (or the equivalent NOR/NOR circuit.) One could use 7400 gates to implement the circuits, and indeed one could implement each gate by a set of transistors. The labor involved in the assembly of such circuits typically is too expensive, and is seldom done today. Usually low-production-volume State Machine circuits are fashioned with CPLDs.

NAND or NOR gates can be connected back-to-back to make a latch and with some additional simple gates a latch can be made into a clocked flip-flop. Hence, considerably complex state machines can be made from either the MAX or the FLEX chip. All lab exercises in the CS 152A/EE 116L course are designed to fit into the much smaller MAX chip; designs for Lab 6 can get close to the gate limitation.

The thousands of gates in the UP-1 MAX CPLD chip are organized into macrocells, each the equivalent of some five many-input NAND gates that feed another NOR gate, which can produce a Product-of-Sums (POS) function that feeds into a flip-flop, or can bypass the flip flop For the MAX chip on the UP-t board there are 128 such macrocells, organized into eight groups; the group is called a Logic Airay Block (LAB.) There are interconnect lines (busses) between the LABs, and inputs and outputs of the LABs are connected to the interconnect lines. Some of these interconnects can be used to reach any other LAB, some extend only to the nearest neighbors.

There are 84 pins on the MAX chip, brought out to 88 sockets aligned at the edge of the socket. Four of the sockets are not used; these are indicated with an X printed on the board next to them. There are a number of dedicated Power/Ground pins and several I/O voltage pins. The chip operates normally on Vcc = $+5V_{dt}$, but it can use and produce lower voltage input and output signals, such as the more recently popular 3.3 Volts. These are separately detected and produced with VCCIO voltages that may be separately supplied to the chip. The UP-l board uses +5 Volts for all these pins. The voltage and ground pins may not be used for signal pins. There are about 70 input and output pins that may be used for a design.

The pattern of interconnections is controlled by pass transistors whose control value (connect or not connect) is kept in a non volatile EPROM The content of the EPROM is set by an industry standard Joint Testing Advisory Group (JTAG) interface, which can be set from the Programmer software using the information in the .PF file for the project. The conversion of the design information to the .pof file is done by the Compiler using your TDF or GDF design files Pin placement information, as well as other device and compilation information is kept in the project's ACF file. It is possible to devise your own file for setting the connections in the CPLD chip, that is to make your own POF file, but it is too tedious to do so. Rely on the compiler to do this job.

The JTAG interface has four signal lines and it allows loading the control bits for the interconnections, and the verification of the downloaded signals by reading the completed interconnection matrix. The interface also has provisions for writing HIGH/LOW values to input lines and reading the HIGH/LOW values back into a software package. This is an industry-standard feature, Boundary Scan Test (BST) architecture. We will not use this feature for the Lab Exercises -- we will test the designs with values coming from input switches and wire the outputs to indicator lights.