
ECE 436 Laboratory 4

TARGETING THE XESS PROTO-BOARD AND PHYSICAL TESTING WITH A LOGIC ANALYZER

Description:

For this laboratory, you will be learning how to constrain your design to FPGA pins, generate a bitstream configuration for your design, load your configuration onto the XESS FPGA prototyping board, and test your physical design using a Tektronix Logic Analyzer. These four processes, along with the processes covered in previous lab assignments, will enable you to complete the entire design flow of your processor. You will step through each of these processes with the TAs in your lab section.

Activity:

While you have been placing-and-routing your design for simulations, you have not been specifying which signals should be connect to which I/O pins. Given that your design will be targeting an FPGA board that has hardwired I/O connections, it is imperative that you constrain your place-and-route process using a .ucf (user constraints file) file, which is simply a netlist between your design's ports and the FPGA's pins. A basic .ucf file with the necessary port mappings will be posted on the Toolkit website, but you can add additional signals to the list for increased testability. Add the jackal.ucf file to your top-level design using the Project -> Add Source dropdown menu in the Project Navigator. The format for this file is as follows:

```
net clk_in loc=p88;      # Input clock
net reset loc=p93;     # Active-low reset from pushbutton

net PC<0> loc=p77;      #necessary for demos
net PC<1> loc=p79;
net PC<2> loc=p80;
net PC<3> loc=p83;
net PC<4> loc=p84;

net a loc=p49;         #7-segment LED mappings
net b loc=p46;
net c loc=p39;
net d loc=p67;
net e loc=p62;
net f loc=p57;
net g loc=p60;
```

For detailed information about the FPGA's pin numbers and pin descriptions, refer to the XESS Programmer's Models document.

Once the .ucf file is defined and added to your project, you are ready to place-and-route your design, followed by the bitstream generation (run Generate Programming File in the Processes for Current Source window of the Project Navigator with the top-level design selected). This will create a .bit file in your project directory, which has the configuration bitstream to configure the FPGA to implement your design.

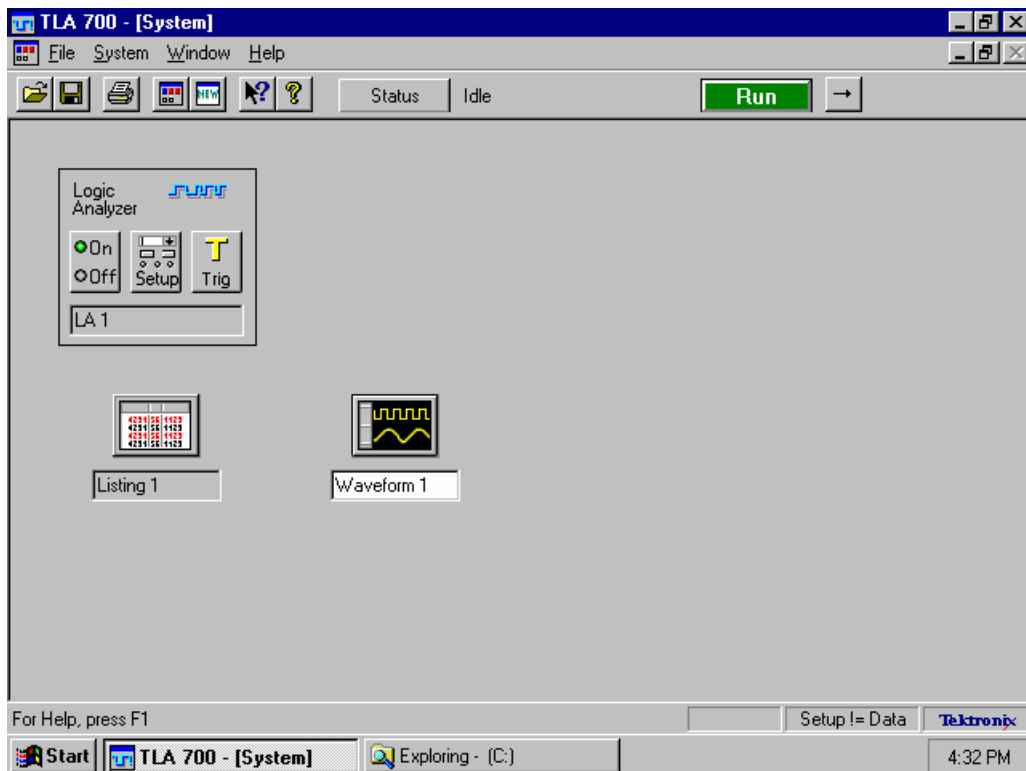
You will next load the design onto the XESS prototyping board. First, a word of advice about the equipment - The XESS board is a delicate piece of hardware. Please handle it with care. Remember to unplug the AC adapter from the power strips after each use in order to protect the board from inadvertent short-circuits. Now, before loading the .bit file onto the Xilinx FPGA, you will want to test the board, and then set the correct clock oscillator

frequency. Please refer to the XESS manual in the lab (pages 11 and 12) for more information about testing the board and setting of this oscillator frequency, especially the settings for the different jumpers (the procedure is quite arcane, be prepared to sneer). ATTENTION: The clock frequency has to be set to at least 25MHz for the interface to the SDRAM to work correctly! This means that the only available divisor values that you can use are: 1 (100MHz), 2 (50MHz), 3 (33MHz) and 4 (25MHz).

Next open the GXLOAD application that is referenced from the desktop or from Programs - > XSTOOLS -> GXLOAD. Drag and drop your .bit file into the FPGA/CPLD box and your .hex file (i.e. your assembled program) into the RAM box. Then click on the "Load" button to program your FPGA and RAM. Push the RESET button on the XESS board, and the program is executed.

While your design will certainly work perfectly the first time you put it on the board, it is still helpful to learn how to use a logic analyzer for testing and debugging. In the lab are several high-performance Tektronix Logic Analyzers (TLAs). These analyzers perform a similar function to waveform simulation, as the internal ISE simulator or ModelSim provide, but they are sampling the actual board outputs.

To prepare for testing, you must connect the TLA probes to the appropriate pins, matching the probes to the pin numbers defined in your .ucf file. The TAs will demonstrate this for you in lab.

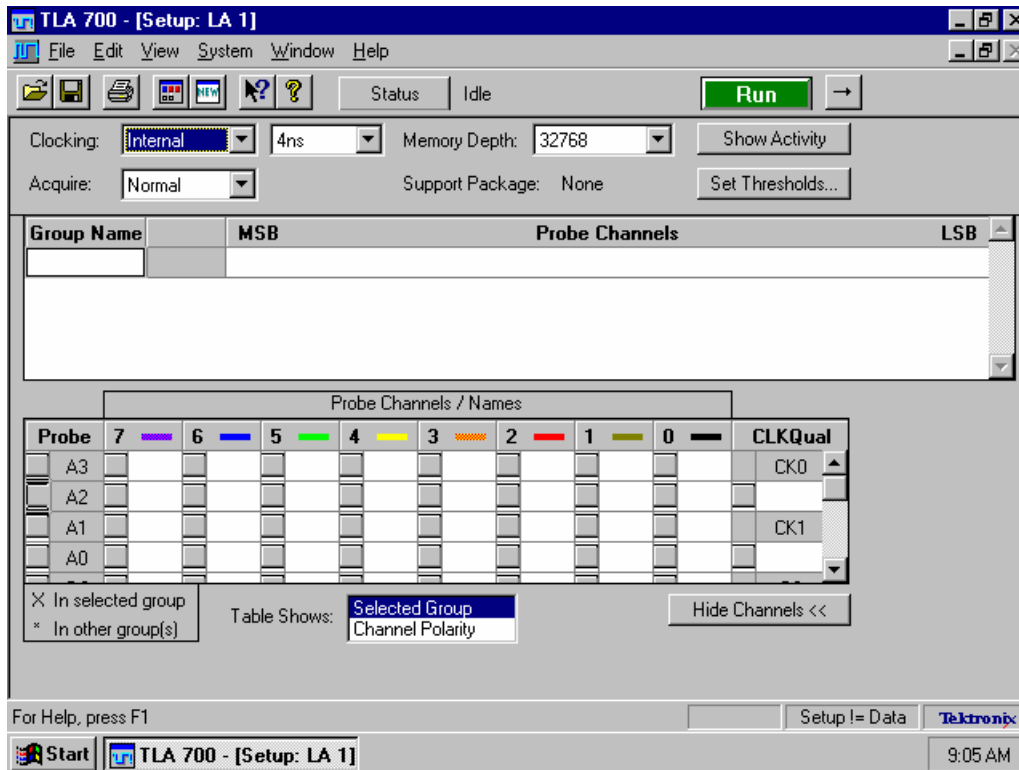


You are then ready to turn on the logic analyzer and begin setting it up. The TLA is actually a portable computer that runs the Windows operating system. The logic analyzer system is simply an application that runs under Windows. When the logic analyzer boots, the TLA Application will start. The easiest way to use the application is to make it full screen and

make all of the windows under it full screen. Then you can navigate through the windows in the TLA 700 Application using the Window menu. Use the Window menu now to go to the System window. When you do, you will see a display like the one above.

The System window shows the configuration of the logic analyzer and the application windows available. All of the application windows can be accessed from the System window as well as through the Window menu. Since the logic analyzer has not been activated, the Listing and Waveform windows do not contain any data. Before any data can be acquired, the analyzer must be configured. This process is performed in the Setup and Trigger windows. The Setup window should be configured first, followed by the Trigger window.

Clicking on the Setup icon in the system window brings up the default Setup window shown below:



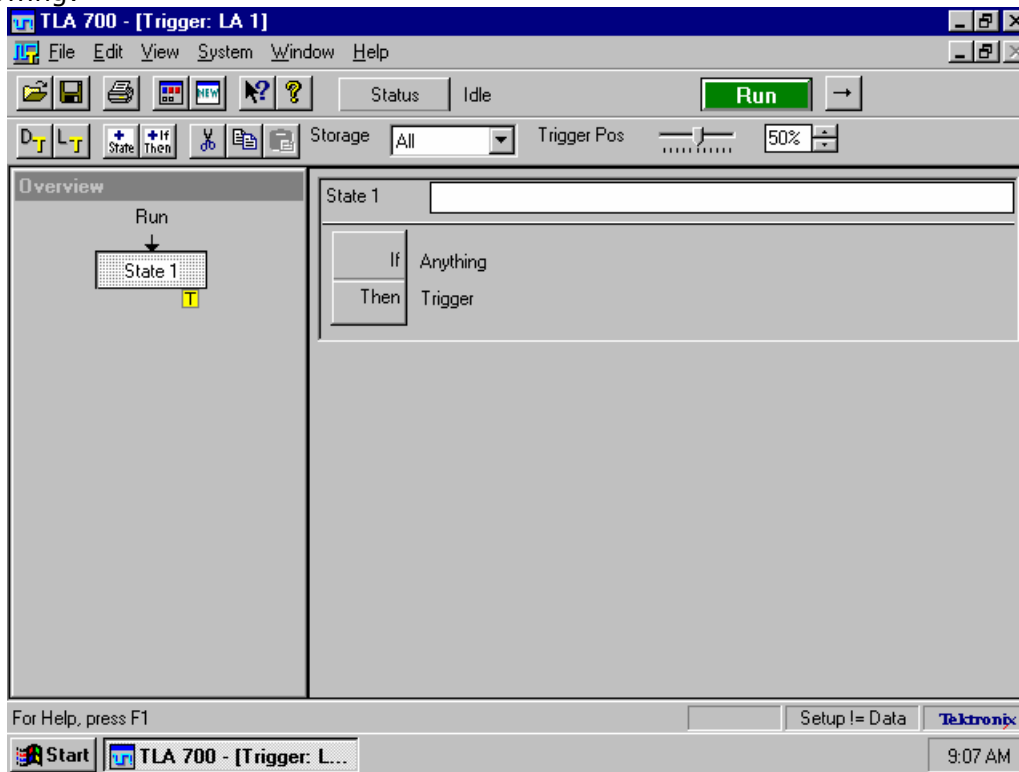
The Setup window is where the signals to be traced are selected and named, and the clocking/sampling scheme for the logic analyzer is defined. By clicking in the Group Name box, new signal groups, such as *reset* and *addr*, can be defined and the logic analyzer probes associated with them specified. The probes are grouped together according to their probe name (*A0*, *A1*, *B0*, *B1*, etc.) and the individual leads are color-coded. To define a group of multiple bits, you can use the following sample notation in the Probe Channels field: *A3(7-0)*. For such notation to be effective, it is imperative that the probes be connected to the board pins in the proper order (defined by the color scheme).

The logic analyzer can sample signals in a synchronous mode (*external clocking*) where samples are taken on a specified clock edge - the leading edge of *CK1* in this case. This mode is used for capturing data on the circuit-under-test's clock edge. Capturing data in this mode usually takes less of the logic analyzer's memory space for a given sampling time. The logic analyzer can also sample signals in an asynchronous mode (*internal clocking*). In this mode, data is sampled on the edge of an internal clock that is not

synchronized with the external clock. This mode is used for checking for data changes between the system clock (*hazards*) and for very fast switching of data signals that occur even between the logic analyzer's clock (*glitches*). Sampling in this way can take more storage area for a given sample time than the external clocking mode. Or, since the logic analyzer has a finite amount of data storage space, it can limit the sample time for which the logic analyzer can store data. We will primarily be using the internal clocking mode.

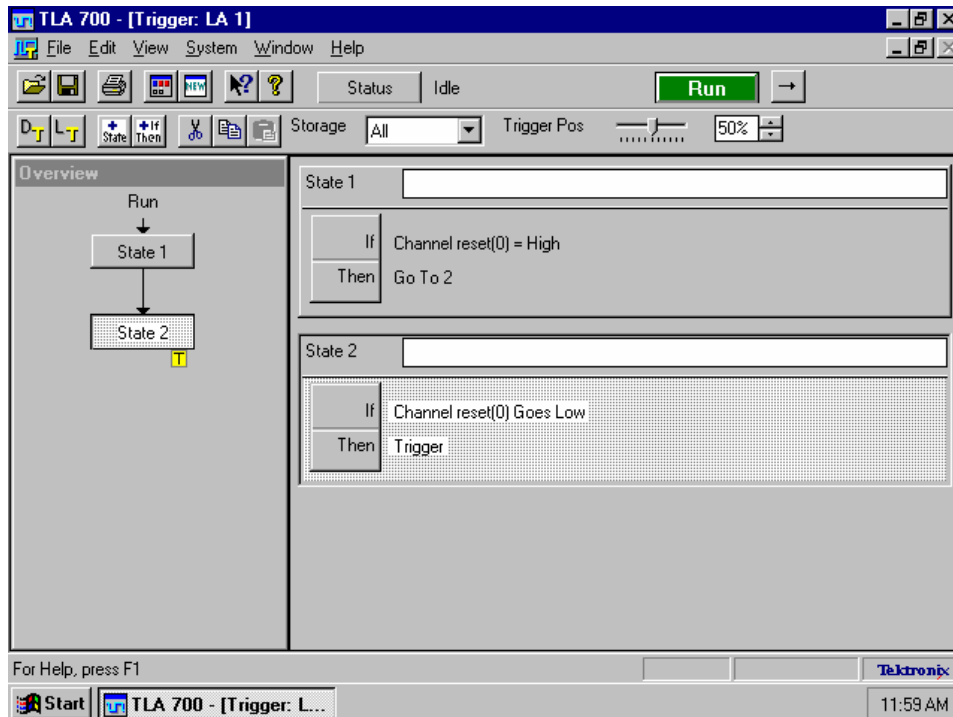
After the Setup window, the Trigger window must be configured. The Trigger window defines when the logic analyzer will begin storing data. For example, it is often desirable to begin capturing data after the system is reset. Therefore, a trigger condition that starts data capture when the *reset* signal goes from high to low must be setup. Obviously, it is easier to do this after the groups are defined in the Setup window so that the trigger can be setup on the *reset* signal by name.

The Trigger window can be accessed from the Window menu, or by going back to the System window and clicking on the Trigger icon. The default Trigger window will look like the following:

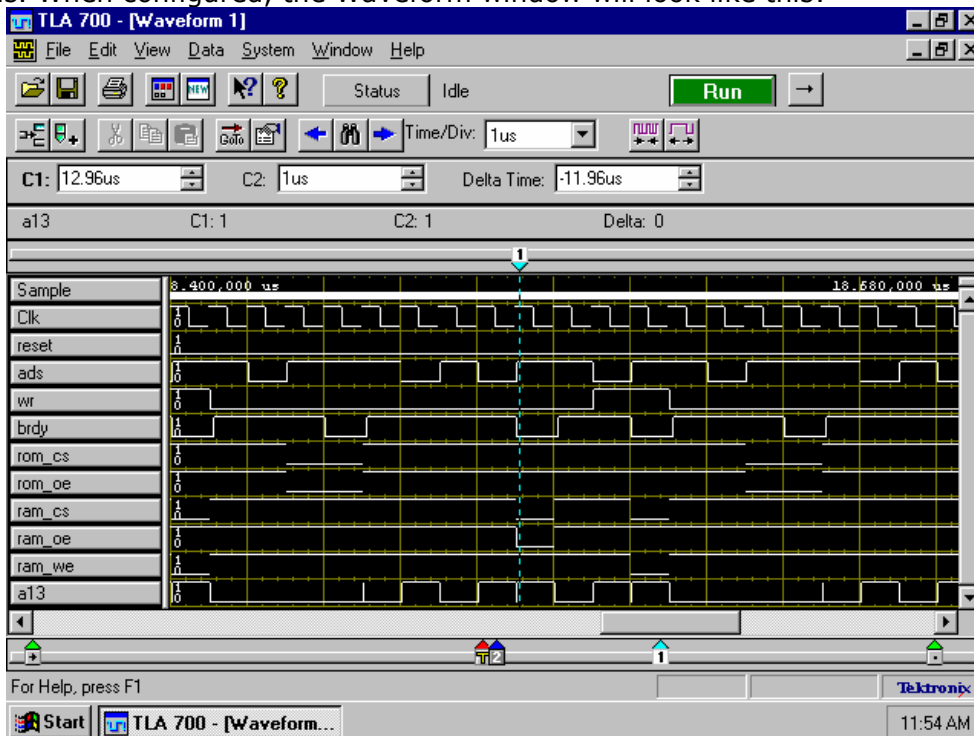


The TLA logic analyzer has the capability to define complex triggering conditions based on a state diagram model. In this case, since the logic analyzer must trigger when the *reset* signal goes from high to low, a two state diagram will be used. In the first state, the logic analyzer will be looking for the *reset* signal to go high (when the reset button is first pushed). At that point, it will transition to state 2 and wait for the *reset* signal to go low (when the reset button is released). The logic analyzer will then trigger and acquire data.

The Trigger window configured to trigger on the high to low transition of the *reset* signal is shown below:



Once the trigger is configured, the data acquisition can be started by pressing the Run button. Then, when the reset button on the test board is pressed and released, the system will trigger and acquire data. A Waveform window can be created using the Window->New Data Window... command and specifying a Waveform window. When the waveform window initially appears, every channel that was placed in a group will appear with its default name (A0(0), A0(1), etc.) These waveforms can be removed and replaced by the group named waveforms. When configured, the Waveform window will look like this:



Deliverable:

After the initial demo, you will be divided into groups of 3 or 4 and assigned to one of the five workstations in the lab. You will be provided with a bitstream that implements a version of a sample Jackal processor design. You will be asked to write a simple program in assembly for the Jackal processor, assemble and load the program together with the bitstream for the processor, run the program and diagnose the result of running the program by using the logic analyzer and other available outputs like the lowest four bits of Register0 at the LED.

The explicit deliverable for this laboratory is to be able to write and assemble a simple assembly program, diagnose the program output on the logic analyzer and explain your diagnosis to the TA.