FPGA programming

INTRODUCTION:

In a lot of digital designs (DAQ, Trigger, …) the FPGAs are used. The aim of this exercise is to show you a simple way to design logic in a FPGA. You will learn all the steps from the idea to the test of the design.

In this exercise you will:
- -discover how we can do parallel applications
- -program a FPGA from the design up to the implementation and the test

The boards used are ALTERA development kit (Figure 1) based on a small FPGA (CYCLONE) with multiple additional interface components like audio CODEC, switches, button, seven-segments display, LEDs, ….
and a home-made board (named detector in the following pages) connected to the development kit with a flat cable (figure 2)

The initial design is loaded into the board.
You will follow this example to understand the design flow. Three exercises are proposed to modify to original design functionality.

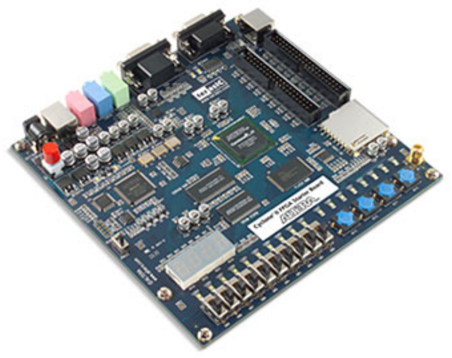


Figure 1: development kit

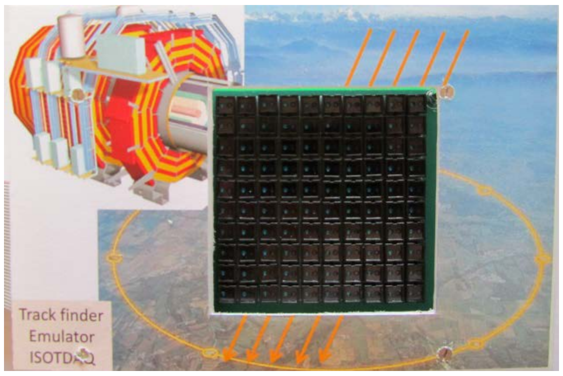


Figure 2: detector

QUICK START:

1) Start the PC, the instructor will give you the user name and password (student—Poland2012).
2) Programs used are: QUARTUS (FPGA tool), ModelSim (simulator), LabView.
3) Ask the tutor in case of problem.

EXERCISE (example)

When you switch on the kit, the initial design is loaded into the FPGA.
On the LabView window, you can see when the marker is passed over the detector (create a trace).
At the same time, you can see in the 7-segments display (Altera kit) the column and the line number over which the maker is positioned.
Ask the tutor for a demonstration.

DESIGN ENTRY

The design file is named "CII_Starter_Default.bdf" (for all exercises you should work with the same design file).
The design is divided in three parts:

a) A green rectangle which is used to transmit the information to the computer via the RS232 connection to display the trace on LabView.
b) A blue rectangle which generate the clock and the logic to control the detector (see Appendix A for detailed functionality).
c) A red rectangle, which contains the logic to detect the trace. You will change the logic in this rectangle in the following exercises.

The idea of all exercises is to detect a trace. As soon as the trace is detected the first 7-segment display blinks. Click on the key0 (Altera kit) to stop the blinking. Now generate another trace.
Spend some time to understand how this design works.

Do you understand it?

COMPILATION

This design is the entry of your logic, it should be compiled now; go to *Processing->Start Compilation*.

The design is compiled for the chosen component (Cyclone II).

The compiler executes multiple tasks: logic optimization generates the binary file to program the FPGA (memory array), extracts the timing between each logic elements (used for the simulation).

SIMULATION

After this, you can check the design with a simulator. To do this you will use ModelSim.

In the “Transcript” tab, type ‘source sim.tcl’, ENTER. The simulator opens the waveform, loads the signals, and starts the simulation.

At the end, stimuli and results are displayed in the wave window.
This simulation generates a trace starting from the top left and finishing at bottom right describing a straight line. (line0 :column0; line1 :column1;….; see figure 3)
Remember where the signal OK goes to “TRUE”.

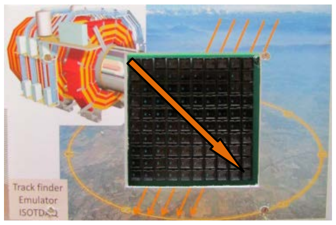


Figure 3: straight line

When you finished with the simulator type ‘quit –sim ‘ ENTER in the “Transcript” tab.

PROGRAM THE KIT

To download the design on the board, (QUARTUS program) go to on *Tools->Programmer* (Check that the Hardware is USB-Blaster, if not ask the instructor).
One file is shown in the window: it is your design. Click on Start .The programmer takes some seconds. At the end, a message appears to inform you that the programming is completed (or not successful: in this case usually the board is switched OFF, or the cable is not well connected).

TEST

Now, you are ready to do the other exercises yourself.

Good Luck!

EXERCISE I

The exercise above uses the graphic to describe the design. In this exercise, we want to do the same with a text design entry (VHDL).

In the QUARTUS design entry (file “CII_Starter_Default.bdf”), delete the line between inst134 and JKFF inst132 and connect the output ‘result’ of “track1”box to the JKFF inst132 with a line.
-Compile the design
-Simulate the design

Go to ModelSim, compile the file marked with a ? in the “Project” tab (click on the file to compile – Menu Compile-> Compile selected)
Type “quit –sim“ in the “Transcript” tab.
Type “source sim.tcl “ in the “Transcript” tab.
Find out the difference with the previous result (check where the signal OK goes to “TRUE”).

Can you explain the difference? Can you modify the file “track1.vhd” to have the same result as in the previous exercise?

-Download the design
-Test the design

EXERCISE II

In this exercise we want to detect a curved trace.

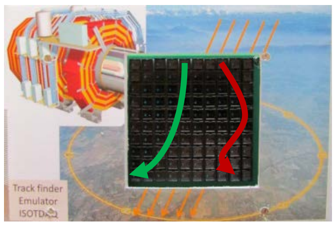


Figure 4

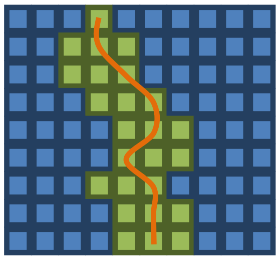


Figure 5: example of trace expected.

In the QUARTUS design entry (file “CII_Starter_Default.bdf”), delete the line between output ‘result’ of “track1” box to the JKFF inst132, and connect the output of the box “trck_fnd01” to JKFF inst132.
The “trck_fnd01” box logic detects only a straight trace. Compile the design and do a simulation (see below for simulation).
The exercise consists to modify the “trck_fnd01” box logic to detect any curve trace as in figure 4.
The trace should start at any pixel in the first line and goes to next line passing to a pixel adjacent to the pixel of the previous line and so forth (figure 5).

-Compile the design
-Simulate the design

Go to ModelSim, compile the file marked with a ? in the “Project” tab (click on the file to compile – Menu Compile-> Compile selected)

To simulate, type “quit –sim’ ENTER in “Transcript” tab to exist any running simulation.

Type ‘source sim2.tcl’ ENTER in “Transcript” tab to simulate in this exercise.

A signal OK becomes true if the logic detects the expected trace.

-Download the design

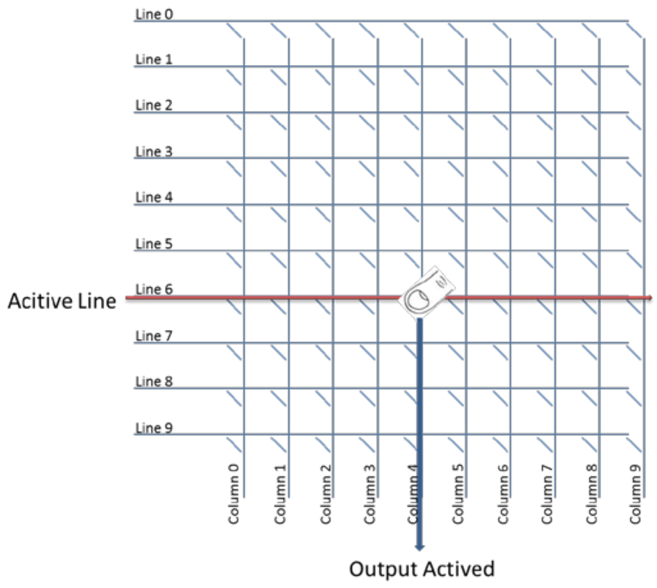
-Test the design

EXERCISE III

If you have time, you can modify the previous file to detect only the curve trace on right or left (not in zigzag like the red trace in figure 4).

APPENDIX A

The detector is a matrix of 10 lines and 10 columns (100 pixels). Only one line is activated at a time.



When a line is activated the result of each column indicates if the marker is over a pixel in the corresponding line. Each line is activated one after the other (0, 1, 2… 8,9,0,1, etc). Each line is activated during 4 clocks cycles. The detection logic checks the result (if pixel is masked by the marker) only during the third clock cycle (signal “check” in the design).