

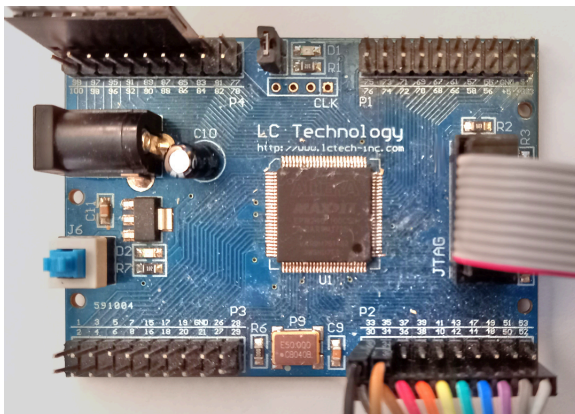
Practice Lab

Introduction

This lab will familiarize you with the CPLD board, the Quartus software, writing lab reports and creating video demonstrations. You are strongly encouraged to submit a lab report and video although the mark will not count towards your final lab mark.

Intel MAX II CPLD Board

This board contains an Intel (formerly Altera) EPM240T100C5 MAX II complex programmable logic device (CPLD). A CPLD is an IC that can be programmed to implement logic functions. This board will be used for most labs. The board's schematic is available on the course website.



The four 20-pin headers will be used to connect components such as an LED display and a keypad. The CPLD pin numbers connected to each header pin are marked on the PCB.

The FPGA's I/O pins use 3.3V logic levels. To avoid damaging the board, *never connect your circuits to an external power supply or use the on-board 5V supply.*

The board has one active-high LED connected to pin 77 and a 50 MHz clock oscillator connected to pin 15. We will use these on-board components for this lab.

The board is powered through a coaxial power connector which must be connected to a USB port or charger. *The USB-Blaster cannot power the board (even though it lights the power LED).*

The board is programmed through the JTAG connector which must be connected to the USB-Blaster from your ELEX 1117 parts kit.

Quartus Prime and ModelSim

Quartus is a logic synthesizer – software that converts a hardware description language (HDL) description of your circuit to a file that can program the CPLD over the JTAG port. We will use the System Verilog HDL in this course.

Procedure

Install the Software

Follow the instructions in the Software Installation and Use document to install Quartus, ModelSim, and MAX II device support. Install the USB-Blaster driver dated 2009-04-21 (version 2.4.16.0) from the course web site.

You can also use [Quartus Prime Lite 21.1](#) from [AppsAnywhere](#) although this requires a network connection when using the software¹.

Logic Synthesis

Create a project named **lab0** and add a System Verilog file named **lab0 sv** containing the following code:

```
// lab0 sv
// ELEX 2117 practice lab - LED blinker
// your name, date

module lab0 (
    input logic clk,
    output logic led ) ;

    logic [25:0] count ;
    always_ff @(posedge clk)
        count <= count + 1'b1 ;

    assign led = count[22] ;

endmodule
```

¹Avoid using the AppsAnywhere version if you have a locally installed version.

This design blinks the on-board LED to show that you can compile a System Verilog design and program the CPLD.

Follow the logic synthesis instructions to synthesize `lab0 sv`. Assign the `led` signal to pin 77 and the `clk` signal to pin 12. Program the FPGA. The jumper labelled J2 (next to LED) must be installed.

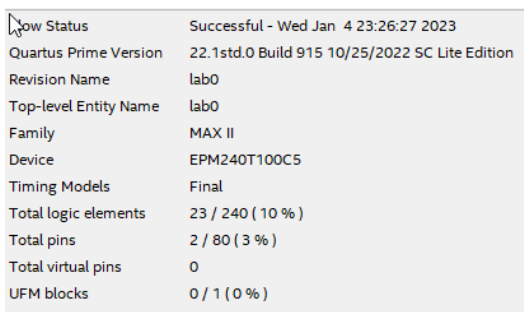
Submissions

Submit the following to the appropriate assignment folders on the course web site: (a) a report containing the items described above, and (b) a short video of your FPGA board showing the LED blinking. Orient the video so that the LED is at the top of the video.

Practice Lab Report

Follow the Report and Video Guidelines document to create a lab report that includes the following:

- a screen capture of the compilation report for `lab0` (**Window > Compilation Report**) similar to:



Row Status	Successful - Wed Jan 4 23:26:27 2023
Quartus Prime Version	22.1std.0 Build 915 10/25/2022 SC Lite Edition
Revision Name	lab0
Top-level Entity Name	lab0
Family	MAX II
Device	EPM240T100C5
Timing Models	Final
Total logic elements	23 / 240 (10 %)
Total pins	2 / 80 (3 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

- a block diagram corresponding to the following Verilog:

```
module sample
(
    input logic sel,
    input logic [3:0] a, b,
    output logic [3:0] y ) ;

    assign y = sel ? a : b ;

endmodule
```

- a corrected version of the following Verilog code². Ignore the content of this code.

```
// example
module sample (
    input wire clk, enable,
    output logic [7:0] count, ecount
) ;

    always @(posedge clk)
        count = !enable ?
            count : count + 1'b1 ;

endmodule
```

²The code shown violates four of the mandatory coding guidelines.