Introduction to Digital Systems Laboratory 4a

Binary Adder

ABSTRACT

In this lab students will build a binary adder circuit using schematics in the Quartus II software.
The process of Active Serial Programming will also be covered.

Introduction
In this lab students will be building a 5-bit adder with an overflow bit. Ten switches will be used to input two 5-bit unsigned numbers (A and B). The resulting sum from the unsigned adder will be displayed in hexadecimal on two seven segment displays.

Implementation

Create a new project in Quartus named Lab4_Lastname in a folder called the same. (E.g. Lab4_Muhlbaier).

- Create a full adder using the schematic editor in Quartus II.
- Create a block from your full adder (File → Create / Update → Create symbol files for current file).
- Create a 5-bit adder from five full adder blocks.
- The carry output bit of the 5-bit adder will be the 6^{th} bit to display.
- Use two hexadecimal to seven segment code converters to display the 6-bit number on the DE0 in hexadecimal format.

Seven Segment Hexadecimal Display Code Converter
There shall be a 4-bit input and a 7-bit seven segment output where the least significant bit (LSb) of the output will represent segment **a** and the most significant bit (MSb) will represent segment **g**.

Dual Seven Segment Hex Display
There shall be a 6-bit input and two 7-bit seven segment outputs. This can be implemented using two of the Seven Segment Hexadecimal Display Code Converters described above.

If you already have a hexadecimal to seven segment code converter designed in Verilog you may use it by converting it to a symbol block (File → Create / Update → Create symbol files for current file).

Using the DE0 the 5-bit A input will be SW[4:0] and the 5-bit B input will be SW[9:5].

Active Serial (AS) Programming

In the previous labs we used a method of programming FPGAs known as JTAG programming. JTAG programming is a good way of programming devices for development and testing purposes; however, it is not a suitable way of programming devices for use or distribution.

FPGAs can be thought of much like RAM in a computer, that is they are referred to as volatile, much like RAM is referred to as volatile. In electronics, when a device is said to have volatile memory, it means if the power to the system is lost all

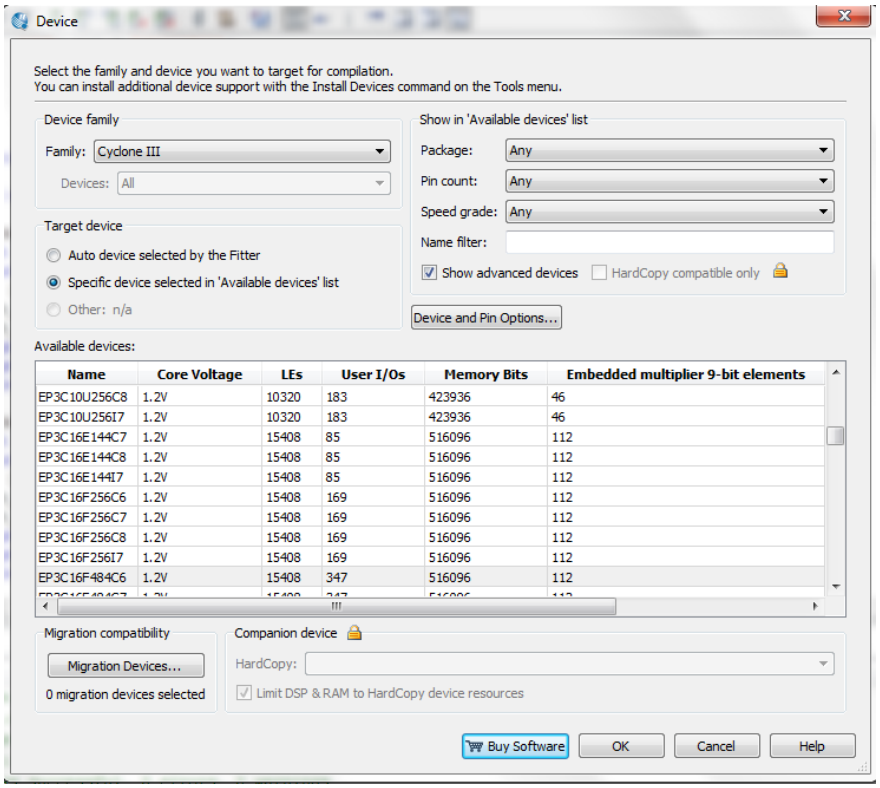
information stored in the device's memory is lost. The same is true with an FPGA, if power to the chip is interrupted; the configuration on the chip is lost. This is a major problem if an FPGA system is to be deployed.

There are many possible solutions to this problem; the most common is described as active programming on startup using a configuration device. In this setup, a chip known as a configuration device, which contains non-volatile memory, is placed on the circuit board along with the FPGA. The design is loaded into the configuration device instead of the FPGA. When the system is turned on, the configuration device immediately programs the FPGA, loading the design into the FPGAs memory.

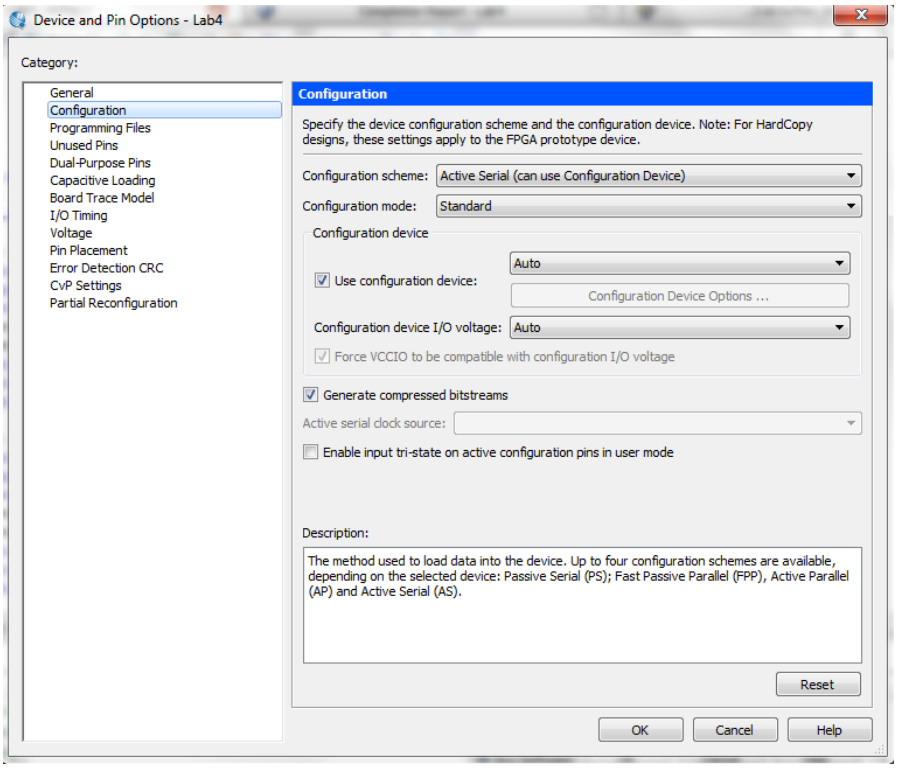
In this section, we will cover how to load a design into the configuration device on the DE0 or DE2 board and have that design loaded onto the FPGA after power up.

The configuration device on the DE2 board is identified as an EPCS16 and on the DE0 board EPCS4. To specify the configuration device perform the following steps.

1. Select **Assignments** -> **Device** and the following window will appear



2. Click on the **Device and Pin Options...** button, the following window will appear.



3. Select the Configuration category
    a. Check the checkbox **Use configuration device**.
    b. Select Auto as the configuration device (or select EPCS4 for DE0 or EPCS16 for DE2).
4. Click OK on all the configuration windows to close them.
5. Recompile your design.

The rest of the procedure is similar to JTAG mode.

Select **Tools** > **Programmer** to reach the programming window
In the **Mode** box select **Active Serial Programming**.

If you are changing the mode from the previously used JTAG mode, a pop-up box will appear, asking if you want to clear all devices. Click Yes.

In the Programmer window make sure the Hardware Setup indicates the USB-Blaster. Look at the file list in the window and check if the configuration file already listed. If it is not you must add it to the window.

Press Add File. A pop-up box will appear. Select the file "***.pof**" from your project output_files directory and click **Open**.

As a result, the configuration file "*.pof" will be listed in the window. This is a binary file produced by the Compiler's Assembler module, which contains the data to be

loaded into the EPCS4 or EPCS16 configuration device. The extension .pof stands for Programmer Object File.

In Programmer window, make sure the "**Program/Configure**" check box on the line of the .pof file is checked.

Flip the RUN/PROG switch on the DE0 or DE2 board to the PROG position. **Note: some computers will also require the DE0 to be power cycled before you press Start.** Press Start in the Programmer window. An LED on the board will light up when the configuration data has been downloaded successfully. Also, the Progress box will indicate when the configuration and programming process is completed.

Once you have downloaded the design to the configuration device, it's time to test the circuit. Flip the RUN/PROG switch to RUN position and the design should run. Test multiple combinations of the input switches to verify the functionality of the circuit.

Extra Credit

(+1 points) Implement the top level entity in Verilog.
(+2 points) Display the output value in decimal instead of hexadecimal.

Deliverable

Demonstration of adder functionality.

Grade Scale

Demonstration Shows:	Grade:
Nothing	0
Tried something	3
Designed but no functionality	6
Works	8
Works with AS Programming	10