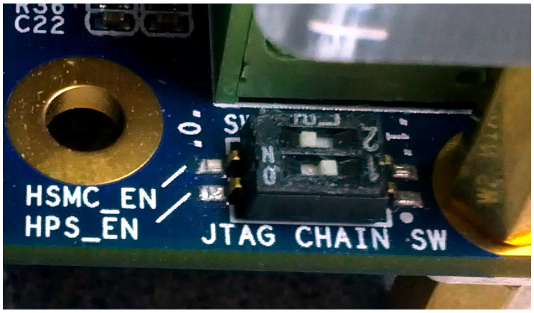
CSEE 4840
Embedded System Design Lab 1: Using the FPGA

Stephen A. Edwards, Columbia University

Spring 2016

Learn how to compile and download an FPGA-only project to the SoCKit board. You will add functionality that allows a user to display and edit the contents of a 16 × 8 bit RAM.

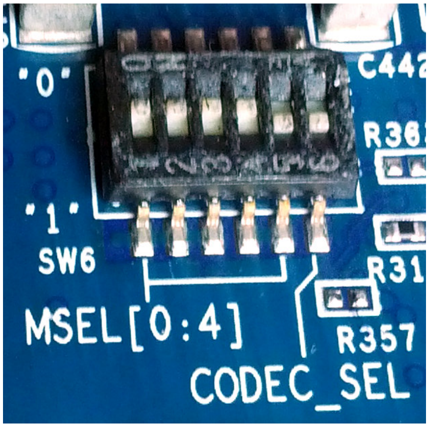
1 Configure the Board



Set the JTAG scan chain to include the HPS but not the HSMC by setting SW4.1 to "1" (off) and SW4.2 to "0" (on) (upper right corner of the top of the board).

JTAG is the serial protocol we use to configure and debug the FPGA.

The HPS refers to the ARM processor and its peripherals; the HSMC is the high-speed mezzanine connector on the board, which we will not use in the labs.



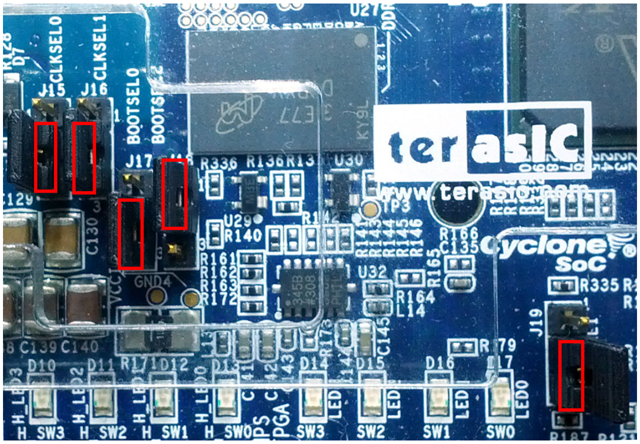
Set the FPGA configuration mode switches (SW6, on the bottom of the board) to 000011.

The MSEL switches control how the FPGA loads its configuration when it is powered on.

Set jumpers J15–J19 such that BOOTSEL[2:0] is 100 and CLKSEL[1:0] is 00.

BOOTSEL controls how the ARM processor boots; 100 sets it to boot from the FPGA. This effectively disables the processor from booting—what we want for this lab.

CLKSEL controls the speed of the HPS peripherals; 00 is slowest.



2 Download and Unpack the Lab 1 files

Download *lab1.tar.gz* from the class website and extract it by typing *tar zxf lab1.tar.gz*. This will create a *lab1* directory containing the files listed below.

Name	Contents
lab1.qpf	Quartus Project File. Select this when opening a project.
lab1.qsf	Quartus Settings File: lists FPGA type, pin names, project files
lab1.sdc	Synopsys Design Constraints: clock pins and frequencies for the timing analyzer
VGA_LED_Emulator.sv	A module that emulates 8 seven-segment LEDs on a VGA monitor connected to the SoCKit.
lab1.sv	Skeleton lab 1 code: the memory, a seven-segment decoder, the controller, and a top-level module that connects these. **Your assignment: modify this file.**
SoCKit_Top.sv	SystemVerilog top-level module for the SoCKit board. Top-level pins and default outputs. Instantiates the LED emulator module and the *lab1* module.
Makefile	Instructions for building *lab1.tar.gz* and for cleaning up unneeded files.

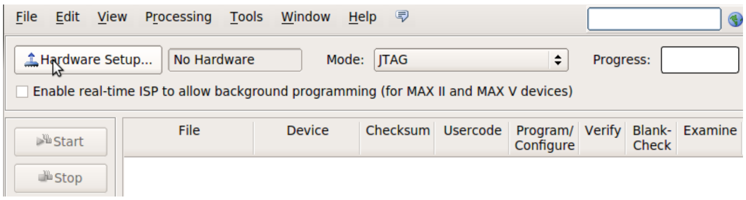
3 Compile and Download the Project

Start Quartus (type *quartus*). Select *lab1.qpf* with File→Open Project....

Compile the design by selecting Processing→Start Compilation. This will take a while. If all goes well, you should see *Quartus II Compilation was successful.*

Download the generated file to the SoCKit board. Select Tools→Programmer.

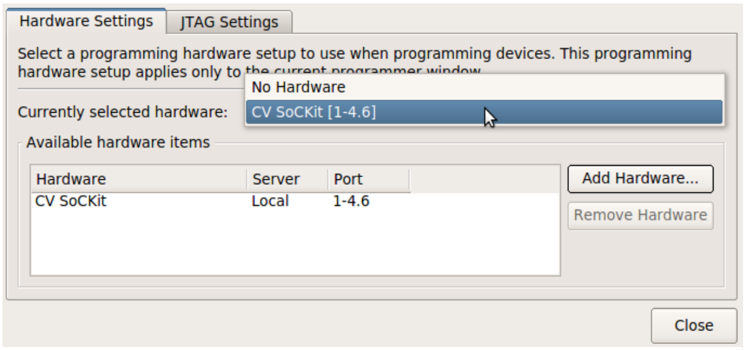
If "No Hardware" appears, **turn on the SoCKit board** and click on Hardware Setup...



Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine

and select "CV SoCKit."



Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: CV SoCKit [1-4.6]

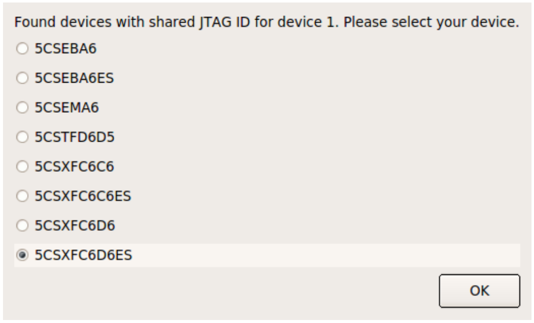
No Hardware
CV SoCKit [1-4.6]

Available hardware items

Hardware	Server	Port
CV SoCKit	Local	1-4.6

Click on "Auto Detect." It should report that it found devices with a shared JTAG ID. Select "5CSXFC6D6ES" and click OK.

Answer "yes' if it asks to update the Programmer's device list.

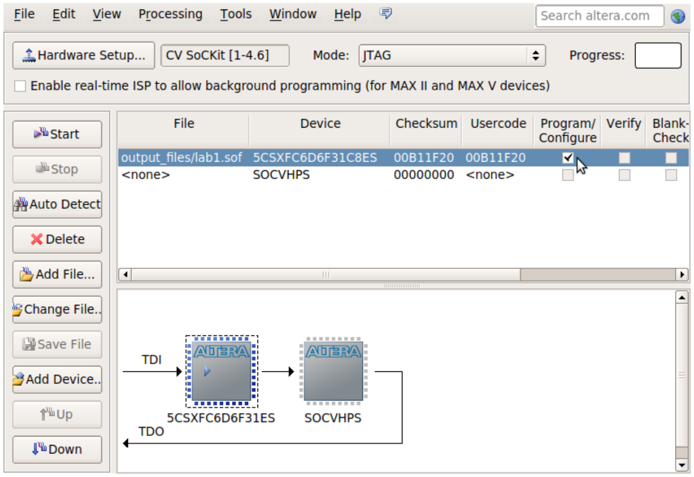


Found devices with shared JTAG ID for device 1. Please select your device.

- ○ 5CSEBA6
- ○ 5CSEBA6ES
- ○ 5CSEMA6
- ○ 5CSTFD6D5
- ○ 5CSXFC6C6
- ○ 5CSXFC6C6ES
- ○ 5CSXFC6D6
- ● 5CSXFC6D6ES

Click on 5CSX… device in the JTAG chain to highlight the first line, then click on "Change File…," enter the *output_files* directory and select *lab1.sof*, which the compilation process should have generated.

Click the "Program/Configure" checkbox for the 5CSXF… device: see the image on the right.



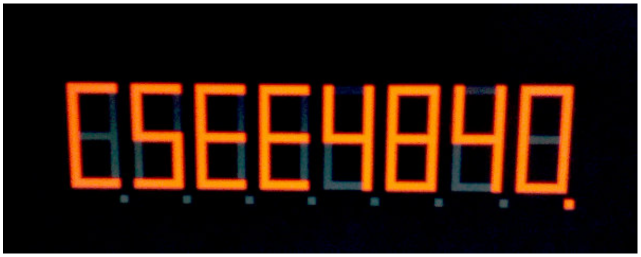
CV SoCKit [1-4.6]   Mode: JTAG

☐ Enable real-time ISP to allow background programming (for MAX II and MAX V devices)

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check
output_files/lab1.sof	5CSXFC6D6F31C8ES	00B11F20	00B11F20	✓	☐	☐
<none>	SOCVHPS	00000000	<none>	☐	☐	☐

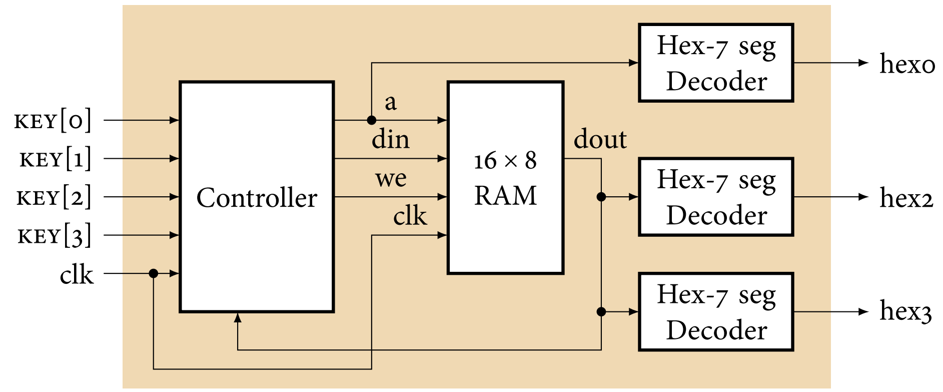
Click "Start." It should take a couple of seconds: the progress bar should go from left to right and finally announce *Success*.

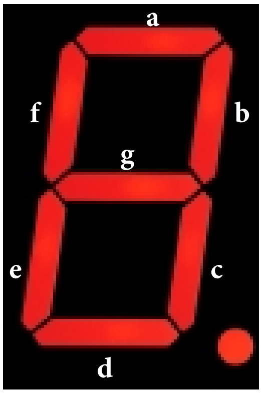
Turn on sw[0] (rightmost on the board) by pushing it up. The VGA screen should now display CSEE4840 in a red-on-black seven-segment style.



Now, switch off sw[0] and try pressing the four keys on the right of the SoCKit board. The display should change.

4 The Lab 1 Design





Implement a memory display and modification circuit according to the block diagram above. The circuit should always display the address and contents of one of 16 byte-wide memory locations.

Have KEY3 and KEY2 increment and decrement the address and KEY1 and KEY0 modify its contents. The KEY inputs are active-low signals from the four pushbuttons on the right side of the SoCKit board. See the *SoCKit User Manual* for details.

The VGA LED emulator displays the *hex* signals on the screen. hex0[0] controls the “a” segment of the leftmost digit, hex0[1] is the “b” segment of the leftmost digit, hex7[2] is the “c” segment of the rightmost digit, etc.

Modify the code in *lab1.sv* to implement your lab. Put your names and unis in the comments.

Turn off SW[0] when you are developing your code so you can see its output.

Submit your modified lab1.sv file on Courseworks.