PROJECT #3 GUIDE

UCB ECEN 5863 FALL 2023 PROJECT #3

TABLE OF CONTENTS

INTRODUCTION

CYCLONE V SOCS: LOWEST SYSTEM COST AND POWER

Altera's Cyclone® V SoCs provide the industry's lowest system cost and power. The SoCs' high-performance levels are ideal for differentiating high-volume applications, such as industrial motor control drives, protocol bridging, video converter and capture cards, and handheld devices. SoCs come in a wide range of programmable logic densities with many system-level functions hardened in silicon — a dual-core ARM® Cortex®-A9 hard processor system (HPS), embedded peripherals, multiport memory controllers, serial transceivers, and PCI Express® (PCIe®) ports.

ARM-BASED HPS

The Cyclone V SoC HPS consists of a dual-core ARM Cortex-A9 MPCore™ processor, a rich set of peripherals, and a multiport memory controller shared with logic in the FPGA, giving you the flexibility of programmable logic and the cost savings of hard intellectual property (IP) due to:

- Single- or dual-core processor with up to 925 MHz maximum operating frequency
- Hardened embedded peripherals eliminate the need to implement these functions in programmable logic, leaving more FPGA resources for application-specific custom logic and reducing power consumption
- Hardened multiport memory controller, shared by the processor and FPGA logic, supports DDR2, DDR3, and LPDDR2 devices with integrated error correction code (ECC) support for high-reliability and safety-critical applications

HIGH-BANDWIDTH INTERCONNECT

High-throughput datapaths between the HPS and FPGA fabric provide interconnect performance not possible in two-chip solutions. This tight integration provides:

- Over 100 Gbps peak bandwidth
- Integrated data coherency
- Significant system power savings by eliminating the external I/O paths between the processor and the FPGA

FLEXIBLE FPGA FABRIC

The FPGA logic fabric lets you differentiate your system by implementing custom IP or off-the-shelf preconfigured IP from Altera or its partners into your designs. This allows you to:

- Adapt quickly to varying or changing interface and protocol standards
- Add custom hardware in the FPGA to accelerate time-critical algorithms and create a compelling competitive edge
- Quickly deploy a custom ARM processor without the extensive design, verification, and non-recurring engineering (NRE) costs required in ASICs

ARCHITECTURE MATTERS

Because Cyclone V SoCs integrate many hard IP blocks, you can lower your overall system cost, power, and design time. SoCs are more than the sum or their parts. How the processor and FPGA systems work together matters greatly to your system's performance, reliability, and flexibility. Altera SoCs are designed to:

- Preserve the flexibility of processor boot or FPGA configuration sequence, system response to processor reset, and independent memory interfaces of a two-chip solution
- Maintain data integrity and reliability with integrated ECC
- Protect DRAM memory shared by the processor and FPGA with an integrated memory protection unit
- Enable system-level debug with Altera's FPGA-adaptive debugging for unmatched visibility and control of the whole device

DE1-SOC BOARD

- The DE1-SoC Development Kit presents a robust hardware design platform built around the Altera System-on-Chip (SoC) FPGA, which combines the latest dual-core Cortex-A9 embedded cores with industry-leading programmable logic for ultimate design flexibility. Users can now leverage the power of tremendous re-configurability paired with a high-performance, low-power processor system. Altera's SoC integrates an ARM-based hard processor system (HPS) consisting of processor, peripherals and memory interfaces tied seamlessly with the FPGA fabric using a high-bandwidth interconnect backbone. The DE1-SoC development board includes hardware such as high-speed DDR3 memory,

video and audio capabilities, Ethernet networking, and much more.
The DE1-SOC Development Kit contains all components needed to use the board in conjunction with a computer that runs the Microsoft Windows 7 or later (*64-bit OS and Quartus Prime 64-bit are required to compile projects for DE1-SoC*).

LEARNING OBJECTIVES

For this project, the objective is for students to:

- Become familiar with the Quartus and SoC EDS development flow, particularly in the case of a SoC with software development flow included.
- Appreciate the capabilities of the Altera Cyclone V System on a Chip, and compare and contrast it to the Xilinx Zynq SoC.
- Learn how to build hardware on a large-scale FPGA.
- Learn how to integrate software with hardware in the same device using both the Altera Monitor Program and the SoC EDS (Embedded Design Suite).
- Design and build a several hardware examples using the Altera Cyclone V SoC.
- Design and build a several similar functions in software using the Altera Cyclone V SoC.
- Consider hardware and software tradeoff possibilities available in the Cyclone V SoC system architecture.

Do not be afraid to ask questions as you work on this project. It involves many processes and actions that may seem complicated and confusing at first, but will become clearer as you work through the modules.

The modules will get progressively more difficult and take more time. Try to finish Modules 1 in the first week, so that you will have the second week to complete Modules 2 and 3; and the last few weeks for Module 4 and to write the report.

I. PROCEDURE

MODULE I: LIGHT THE LIGHTS, HIT THE HEIGHTS

1. From Canvas, download the pdf documents lab1vhdl.pdf and lab5vhdl.pdf, **or** the Verilog versions lab1verilog.pdf and lab5verilog.pdf. Also download the source files found in lab1vhdl_design_files.zip **or** lab1ver_design_files.zip.
2. Follow the instructions in lab1 Parts I – IV (skip V and VI) and record your observations.
   a. If your results look different than the lab guide, do not be alarmed, as there are some differences between the tools used in the guide and current version of the tools.
   b. Do not be surprised when the initial and even subsequent compiles have errors. This will point you to the additional work you need to do.
3. Follow the instructions in lab5 parts I – III (skip IV) and record your observations.
4. In step 2 do the displays work as expected?
5. In step 3, is your timer accurate? How would you confirm this?
6. Record the fMAX for each lab.
7. Estimate the % utilization of the FPGA logic for each lab at completion.

8. Record your observations of the board behavior once the FPGA is programmed. Does it behave as you expected?
   Note: Build each small part (lab 1, lab5) as a separate project to facilitate availability of each Verilog/VHDL file for submission and a .sof file for burning during DEMO.

MODULE II: EMBEDDED SOFTWARE DEVELOPMENT ON AN SOC USING C

1. From Canvas, download the pdf document lab12vhdl.pdf, **or** the Verilog version lab12verilog.pdf. Also download the source files found in lab12vhdl.zip **or** lab12verilog.zip.
2. Follow the instructions in lab12 Parts I and II (skip III) and record your observations.
   a. If your results look different than the lab guide, do not be alarmed, as there are some differences between the tools used in the guide and current version of the tools.
   b. Do not be surprised when the initial and even subsequent compiles have errors. This will point you to the additional work you need to do.
   c. Record the Fmax and %utilization for each Part.

3. This next part is an exercise in using C code with the ARM Cortex-A9 processor. We will use the Altera Monitor Program software to compile, load, and run application programs written in the C language. In this exercise you have to be familiar with both the C language and the ARM assembly language. You should read the parts of the Monitor Program tutorial that discuss the use of C code. This tutorial can be accessed from Altera's University Program website, or by selecting Help > Tutorial within the Monitor Program software. You also need to be familiar with a number of I/O ports in the DE1-SoC Computer, including the parallel ports connected to the red LEDs, 7-segment displays, and pushbutton switches, as well as the A9 Private Timer port. These I/O ports are described in the documentation for the DE1-SoC Computer.
   Create a C program file called part1.c that finds the largest number in a list of at least 7 32-bit integers that is stored in the memory. Make a new Monitor Program project for this part of the exercise. In the Monitor Program screen shown in Figure 2 select C Program in the Program Type dropdown menu, and on the screen that follows select your part1.c file. In the screen of Figure 3 set the Terminal device to Semihosting. This setting causes the output of the printf library function to appear in the Terminal window of the Monitor Program graphical user interface.

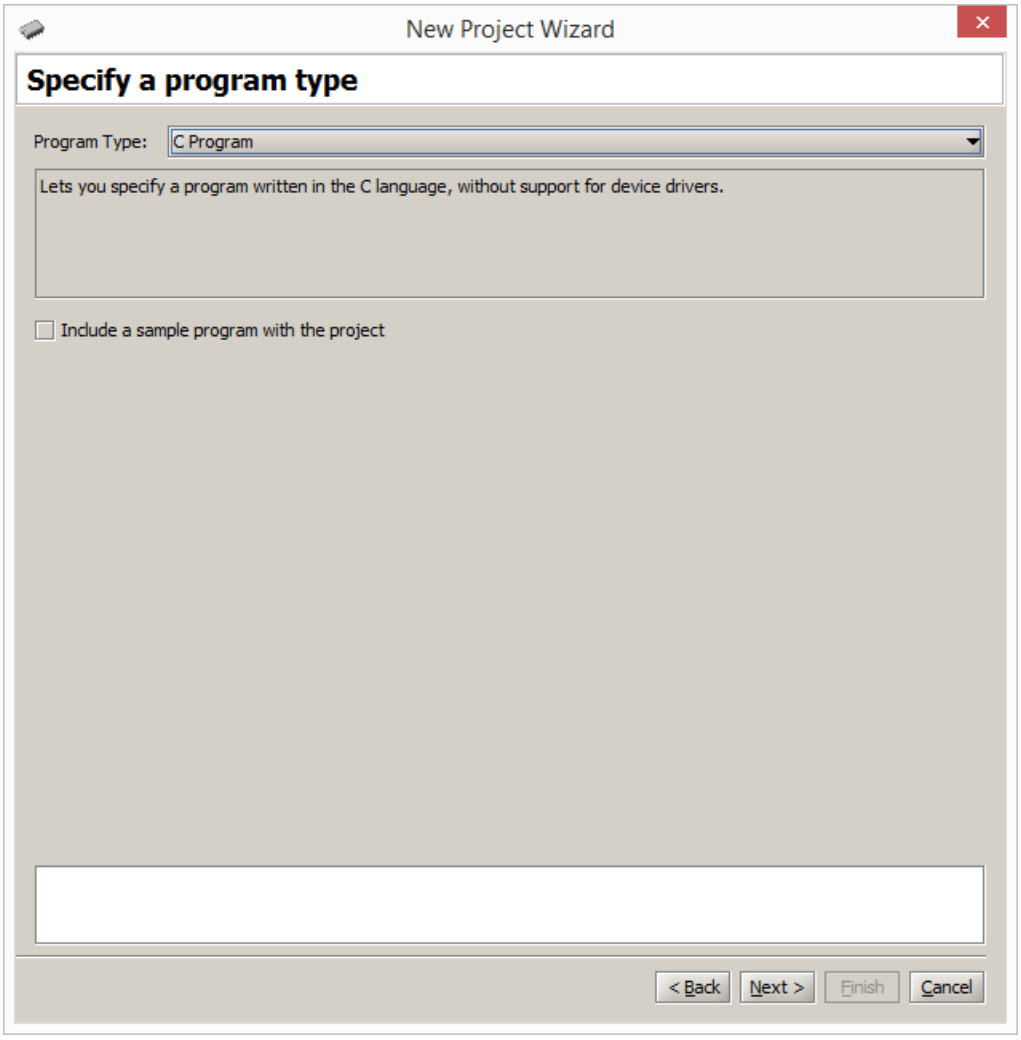

Figure 2: Setting the program type.

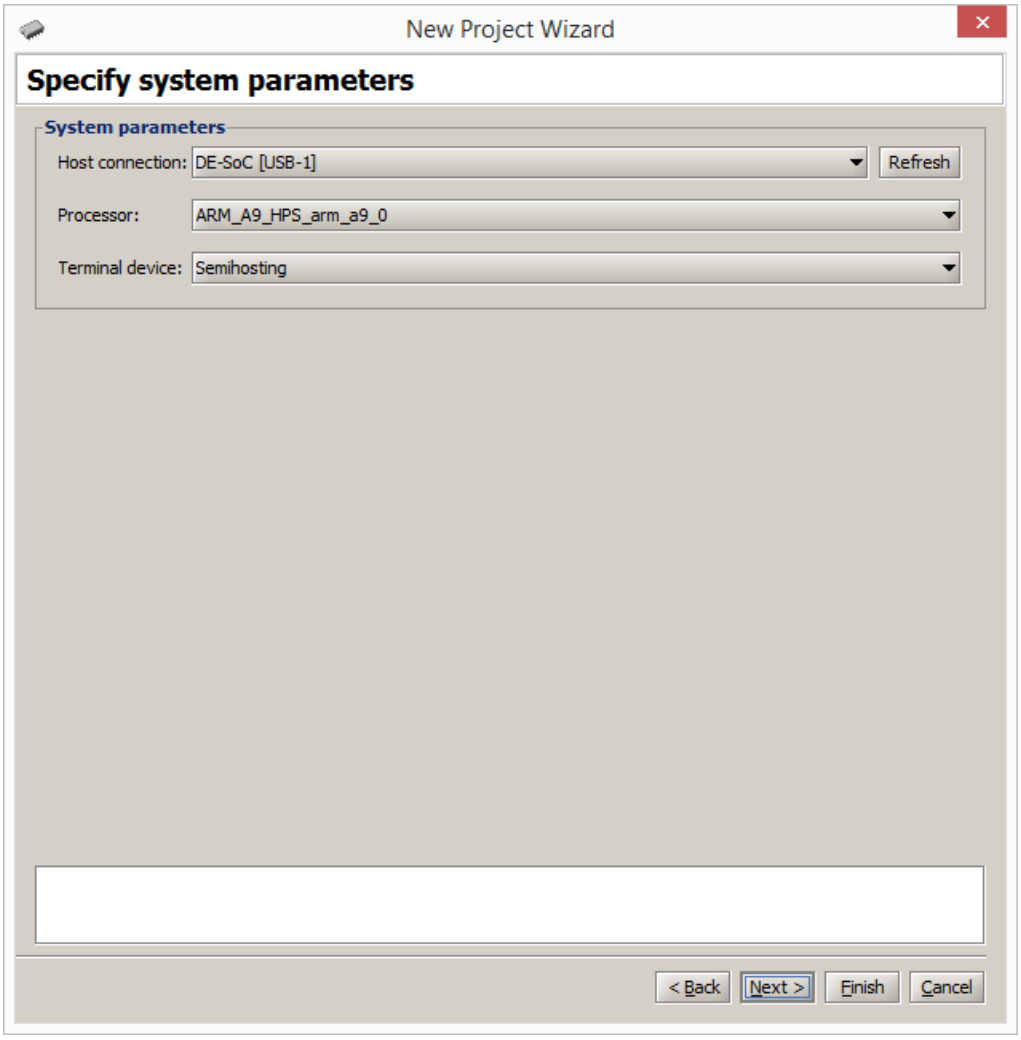

Figure 3: Configuring the Terminal window.

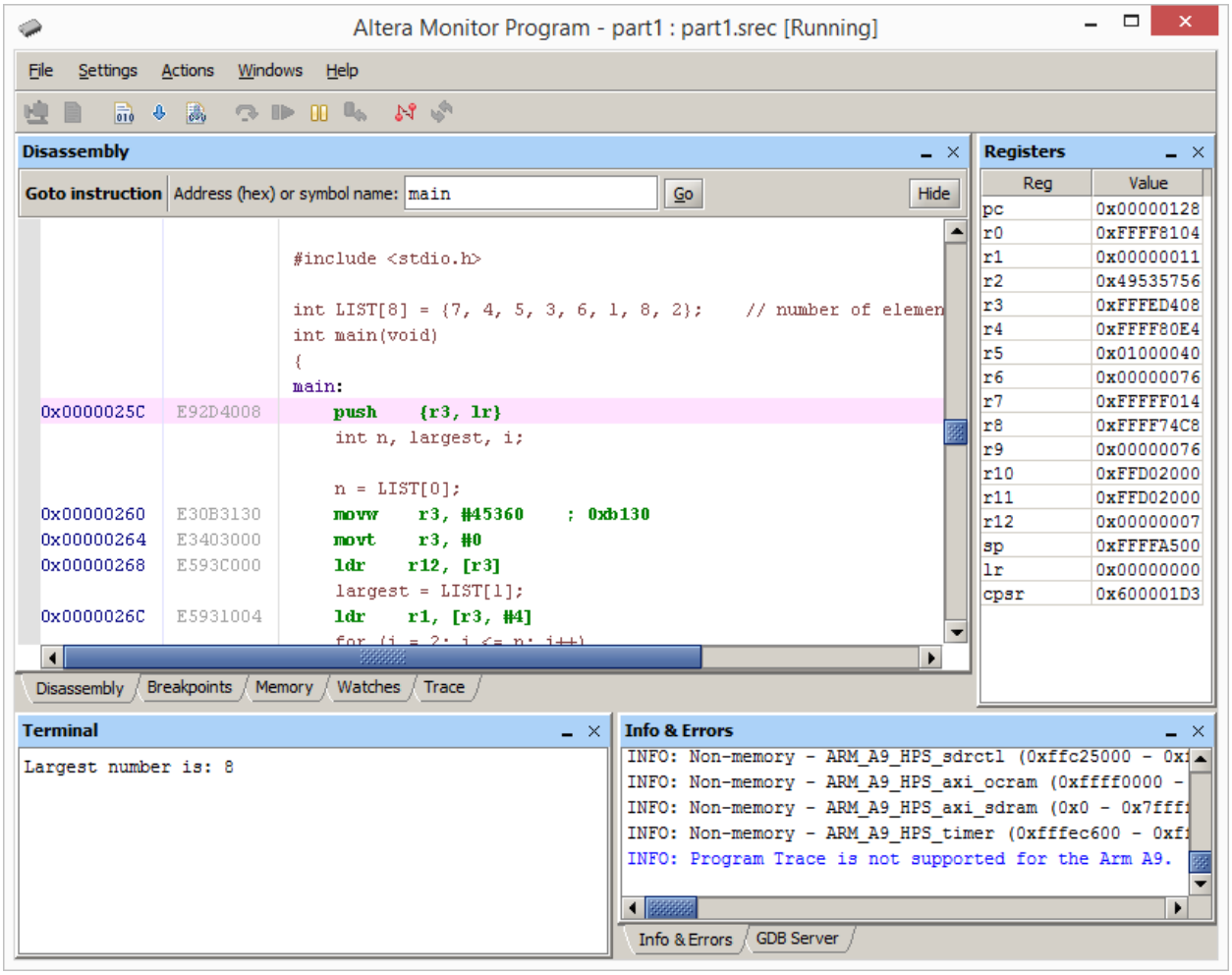

Figure 4: Displaying the code for a C program.

4. In previous section you were asked to implement a real-time clock in the DE1-SoC Computer. The clock-time was shown on the HEX3 - 0 seven-segment displays in the format SS:DD, with SS representing seconds and DD representing hundredths of a second. Time was measured in intervals of 0.01 seconds by using polled I/O with the A9 Private Timer, and the clock could be stopped/run by pressing one of the pushbutton KEYs.

   In this part of the exercise you are to write a C program that implements a real-time clock. Display the clock-time on the 7-segment displays HEX5 - 0 in the format MM:SS:DD, where where MM are minutes, SS are seconds, and DD are hundredths of a second. Measure time intervals of 0.01 seconds in your program by using polled I/O with the A9 Private Timer. You should be able to stop/run the clock by pressing any pushbutton KEY. When the clock reaches 59:59:99, it should wrap around to 00:00:00.

   Make a new folder to hold your Monitor Program project for this part. Create a file called part2.c and type your C code into this file. Make a new Monitor Program project for this part of the exercise, and then compile, download, and test your program.

5. Write a C program that scrolls the message dE1-SoC in the right-to-left direction across the 7-segment displays HEX5 – 0. The content of the displays in each step should appear as illustrated in Table 1. You should scroll the display at a rate of 0.2 seconds per character. You should be able to stop/run the scrolling message by pressing any pushbutton KEY.

Time slot	Display					
0	d	E	1	-	S	o
1	E	1	-	S	o	C
2	1	-	S	o	C	
3	-	S	o	C		
4	S	o	C			
5	o	C				
6	C					
7						
8						d
9					d	E
...			...			

Table 1. Scrolling the message dE1-SoC on *HEX*5 – 0.

Note that scrolling a message across the 7-segment displays is similar in nature to the task of implementing a real-time clock, from Part 2. You should be able to reuse most of your code from Part 2. But instead of updating the clock each time the A9 Private Timer expires, you need to update the scrolling message. Make a new folder to hold your Monitor Program project for this part. Create a file called part3.c and type your C code into this file. Make a new Monitor Program project, compile, download, and test your program.

MODULE III: CONNECTING THE HPS TO THE FPGA

1. Download and follow the directions in the My first HPS-Fpga pdf on Canvas.
2. Record the fMAX for each lab. What is the highest speed clock used in the design?
3. Estimate the % utilization of the FPGA logic for each lab at completion.
4. Record your observations of the board behavior once the FPGA is programmed. Does it behave as you expected?

MODULE IV: DO IT YOURSELF

1. Present your idea to the professor or TA and use the concepts you learned in the class and build a system of your choice. Use either the DE10-Lite, SmartFusion2, Pynq or DE-1 SoC board.

II. DELIVERABLES

Modules 1 deliverables are due at 11:59 pm of 19 November 2023. Module 2 deliverable are due on 29 November 2023. Modules 3 deliverables are due on 9 December 2023. Module 4 deliverables, including class presentations are due on 14 December 2023. The final report is also due on December 14th.

TECHNICAL REPORT

Each partner pair must provide a technical report that explains the project objectives and test results. At a minimum, it should include:

1. Executive Summary
2. Objectives
3. Procedure
4. Module Test Results
5. Lessons Learned
6. Conclusions
7. Appendix: References
8. ReadMe.txt

The report should be 7 pages or less in length in a Word document. Be sure the report addresses the questions posed in the module procedures.

RECORDED OBSERVATIONS, TEST DATA, AND IMAGES

Include observations recorded in a lab notebook, test data taken, and any digital pictures of the proceedings.

FPGA PROJECT DIRECTORY ZIPPED FOR EACH OF 4 MODULES

Starting with the top level of each module, zip up the entire directory including subfolders. This will allow us to replicate your work and help provide feedback on any errors you encounter. Label them appropriately so we can find the information for each module. With submission include a ReadMe.txt to describe and list the locations of deliverables.

SOFTWARE FILES FOR ALL THE MODULES IN A ZIP FILE

Zip up all source files used to run the ARM Cortex-A9 software.

III. EVALUATION

Any award to be made pursuant to this project will be based upon deliverables. The following elements will be the primary considerations in evaluating all submitted projects and in the selection of a partner team for Top Prizes:

25% Technical Report

20% TA Demo of Modules 1-3

10% Deliverables for Project Module 1

10% Deliverables for Project Module 2

10% Deliverables for Project Module 3

25% Deliverables for Project Module 4