UMass Amherst

College of Engineering

Lab 1: Exploring Buffer Overflow Attacks [2]

ECE 371: Intro to Security Engineering

Objectives

The objective of this lab is to help you experiment with buffer overflow attacks in a simple program. This project will teach you how use the Quartus Prime Lite software that we are using to design hardware for the Nios II FPGA. You will be programming the buffer overflow script in C.

Tools

- Altera Quartus Prime
  - Program that allows for circuits to be designed on Altera FPGA boards
  - Helps with design, simulation, synthesis, and downloading bitstream
  - Contains another tool called QSYS that is used to add other peripherals to the board, such as ethernet, serial ports, etc.
- Nios II Software Build Tools for Eclipse
  - Software platform that streamlines developing programs to use with Nios II board, program in C language

Lab Overview

- Follow tutorial on how to make a very simple "Hello World Small" program that prints text to the console
- Written instructions with screenshots included here
- Add additional peripherals to design to allow for both reading and writing to console
- Write a program that has the potential to have a buffer overflow (more details ahead)
- We assume that you are proficient in C at this point based on the prerequisites for this course. However, if you feel that you need a refresher, we recommend the following resources:
  TutorialsPoint: https://www.tutorialspoint.com/cprogramming/
  W3Schools: https://www.w3schools.in/c-tutorial/

Continued from Lab 0...

- You are now familiar with adding and removing components on QSYS and Quartus and compiling the machine code
- We will make some changes to them to start using the board with this software
- Connect your board to Power and the USB port on your computer

What is an FPGA?

For more information on how the Nios II board works, or on FPGAs in general, please see the following:

Basics of Programmable Logic, FPGA Architecture:
https://www.youtube.com/watch?v=jbOjWp4C3V4
DE1 SoC Manual:
https://www.intel.com/content/dam/www/programmable/us/en/portal/dsn/42/doc-us-dsnbk-42-1004282204-de1-soc-user-manual.pdf

Altera DE1 SoC Short Video: https://www.youtube.com/watch?v=aPXMkTJxD_s

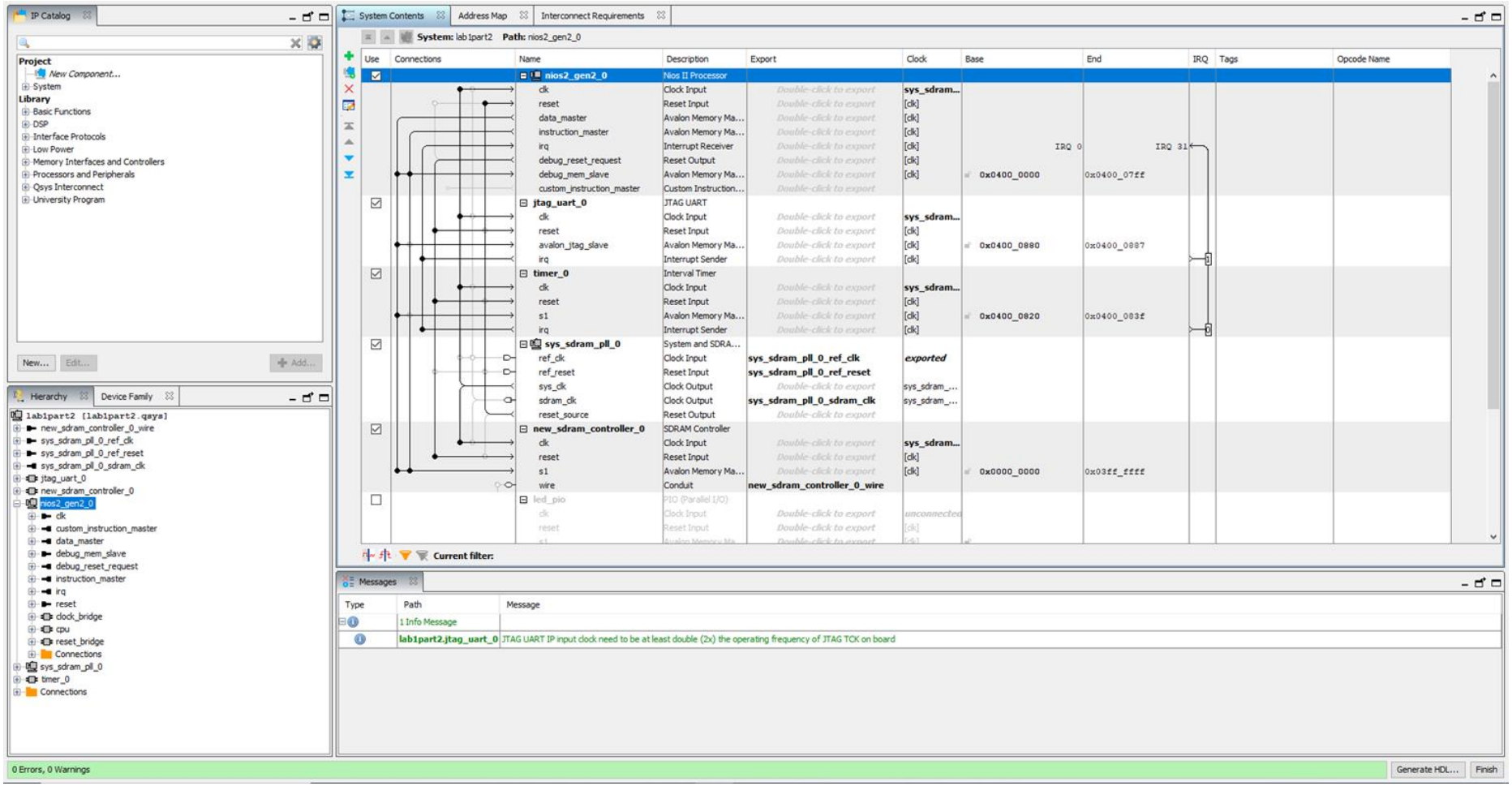
In this Lab:

- you will need to create a new project and follow most of the same steps from Lab 0 apart from two main differences:
  - You will need to use slightly different components in QSYS
  - You will be using the Hello World template in Eclipse
  - You will be using the C code on Moodle (buffer_overflow.c) and will make a few modifications to the code

QSYS

- You will need to use the following hardware in your QSYS design and connect as shown on the next slide
  - Nios II Processor (altera_nios2_gen2)
    - Specify Nios II/e
    - Change reset memory vector to **new_sdram_controller_0.s1**
  - JTAG UART (jtag_uart)
  - SDRAM Controller (altera_avalon_new_sdram_controller) with memory width of 16 bits, 13 rows, and 10 columns
  - PLL (altera_up_avalon_sys_sdram_pll)
  - Timer (altera_avalon_timer) with period 125ms, counter size 32 bits

QSYS


IP Catalog
Project
New Component...
System
Library
Basic Functions
DSP
Interface Protocols
Low Power
Memory Interfaces and Controllers
Processors and Peripherals
Qsys Interconnect
University Program
New...
Edit...
Add...
Hierarchy
Device Family
lab1part2 [lab1part2.qsys]
new_sdram_controller_0_wire
sys_sdram_pll_0_ref_clk
sys_sdram_pll_0_ref_reset
sys_sdram_pll_0_sdram_clk
jtag_uart_0
new_sdram_controller_0
nios2_gen2_0
clk
custom_instruction_master
data_master
debug_mem_slave
debug_reset_request
instruction_master
irq
reset
clock_bridge
cpu
reset_bridge
Connections
sys_sdram_pll_0
timer_0
Connections
System Contents
Address Map
Interconnect Requirements
System: lab1part2 Path: nios2_gen2_0
Use
Connections
Name
Description
Export
Clock
Base
End
IRQ
Tags
Opcode Name
nios2_gen2_0
Nios II Processor
clk
Clock Input
sys_sdram...
reset
Reset Input
[clk]
data_master
Avalon Memory Ma...
instruction_master
irq
Interrupt Receiver
IRQ 0
IRQ 31
debug_reset_request
Reset Output
debug_mem_slave
Avalon Memory Ma...
0x0400_0000
0x0400_07ff
custom_instruction_master
Custom Instruction...
jtag_uart_0
JTAG UART
avalon_jtag_slave
0x0400_0880
0x0400_0887
timer_0
Interval Timer
s1
0x0400_0820
0x0400_083f
sys_sdram_pll_0
System and SDRA...
ref_clk
sys_sdram_pll_0_ref_clk
exported
ref_reset
sys_sdram_pll_0_ref_reset
sys_clk
sdram_clk
sys_sdram_pll_0_sdram_clk
reset_source
new_sdram_controller_0
SDRAM Controller
0x0000_0000
0x03ff_ffff
wire
new_sdram_controller_0_wire
led_pio
PIO (Parallel I/O)
Double-click to export
unconnected
Current filter:
Messages
Type
Path
Message
1 Info Message
lab1part2.jtag_uart_0
JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board
0 Errors, 0 Warnings
Generate HDL...
Finish

QSYS

- You will also need to export several items, as shown on the previous slide. To export, simply double click on the cell of the item to export in the Export column. You do not need to change any of the names
- The items you are exporting are:
  - sys_sdram_pll_0_ref_clk
  - sys_sdram_pll_0_ref_reset
  - sys_sdram_pll_0_sdram_clk
  - new_sdram_controller_0_wire
- As in Part 1, Generate HDL from your QSYS project. Do not forget to add the qip file to your project.

Pin Assignment

- Add the count_bin_top.v file to your project and set as the top-level entity
- Edit the count_bin_top.v file such that it matches the name of your qip file (typically the same name as your project) as shown on the next slide

- The DE1_SoC.qsf file is used to input all of the pin assignments for you, since there are many pins in this design
- To import the DE1_SoC.qsf file and assign the pins, go to Assignments > Import Assignments and add this file
- Please read through both of these files

count_bin_top.v*

```
61
62
63      wire [15:0] hex5_hex4;
64
65      assign HEX4 = ~hex5_hex4[ 6: 0];
66      assign HEX5 = ~hex5_hex4[ 14: 8];
67
68
69    lab1part2 bin_count (
70
71
72          .sys_sdram_pll_0_ref_clk_clk  (CLOCK_50),
73          .sys_sdram_pll_0_ref_reset_reset (1'b0),
74
75           // SDRAM
76          .sys_sdram_pll_0_sdram_clk_clk                    (DRAM_CLK),
77          .new_sdram_controller_0_wire_addr                      (DRAM_ADDR),
78          .new_sdram_controller_0_wire_ba                   (DRAM_BA),
79          .new_sdram_controller_0_wire_cas_n                (DRAM_CAS_N),
80          .new_sdram_controller_0_wire_cke                  (DRAM_CKE),
81          .new_sdram_controller_0_wire_cs_n                      (DRAM_CS_N),
82          .new_sdram_controller_0_wire_dq                   (DRAM_DQ),
83          .new_sdram_controller_0_wire_dqm                  ({DRAM_UDQM,DRAM_LDQM}),
84          .new_sdram_controller_0_wire_ras_n                (DRAM_RAS_N),
85          .new_sdram_controller_0_wire_we_n                      (DRAM_WE_N),
86
87          );
88
89
90          endmodule
91
```

Change text in red to match the name of your qip file!

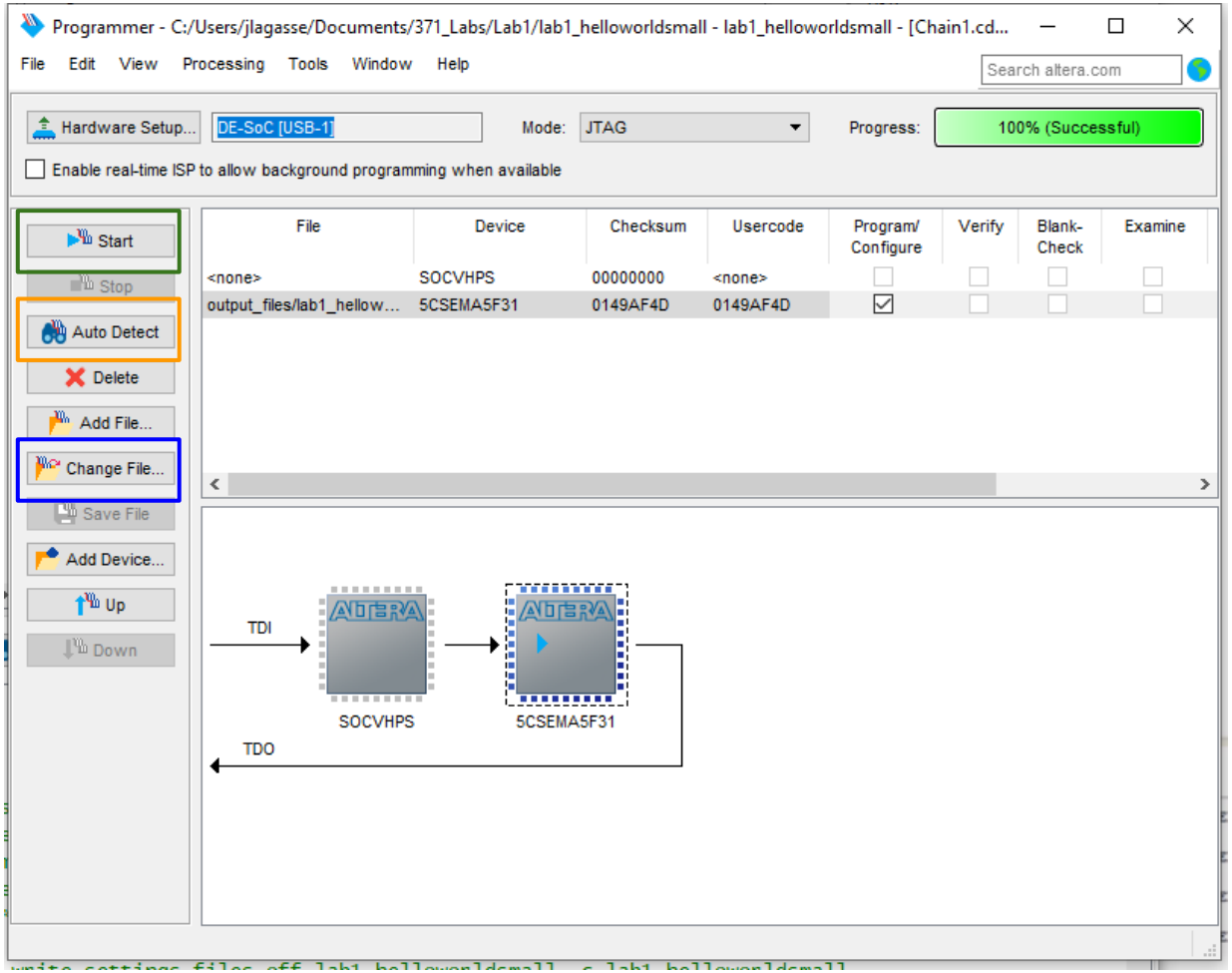
Pin Assignment

- Run Analysis and Synthesis
- Open the pin assignments (Assignments > Pin Planner), look to see if all pins were assigned properly
- Run Start Compilation

Program

- Plug in the DE1-SoC Board to power using the black power cable, and to your computer using the grey cable
- Select Tools -> Programmer to open the Programmer tool
- Select Auto Detect, a new dialogue box should open, choose **5CSEMA5** from the list
- Right click on the 5CSEMA5 entry, change the file to the target SOF file
- **5CSEMA5** will turn into **5CSEMA5F31**
- Be sure that DE1-SoC is connected to the computer via the provided USB cable and turned on
- Press Start to program the device!

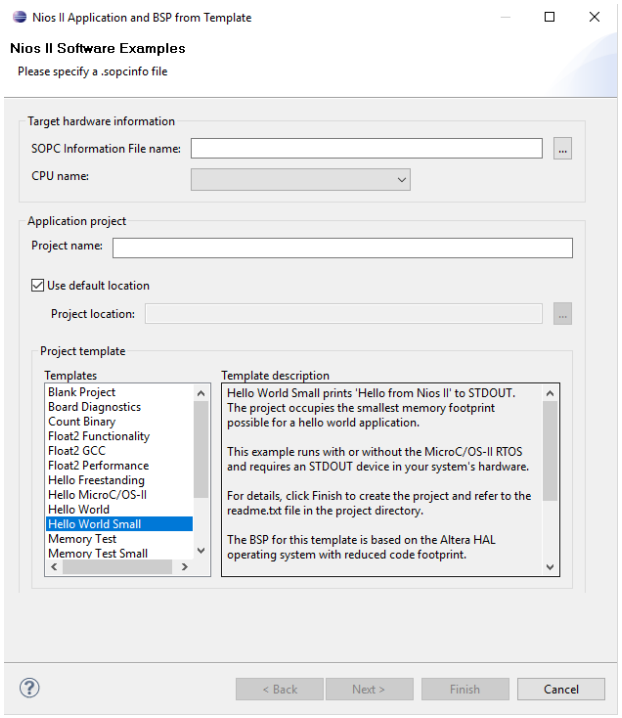
Program



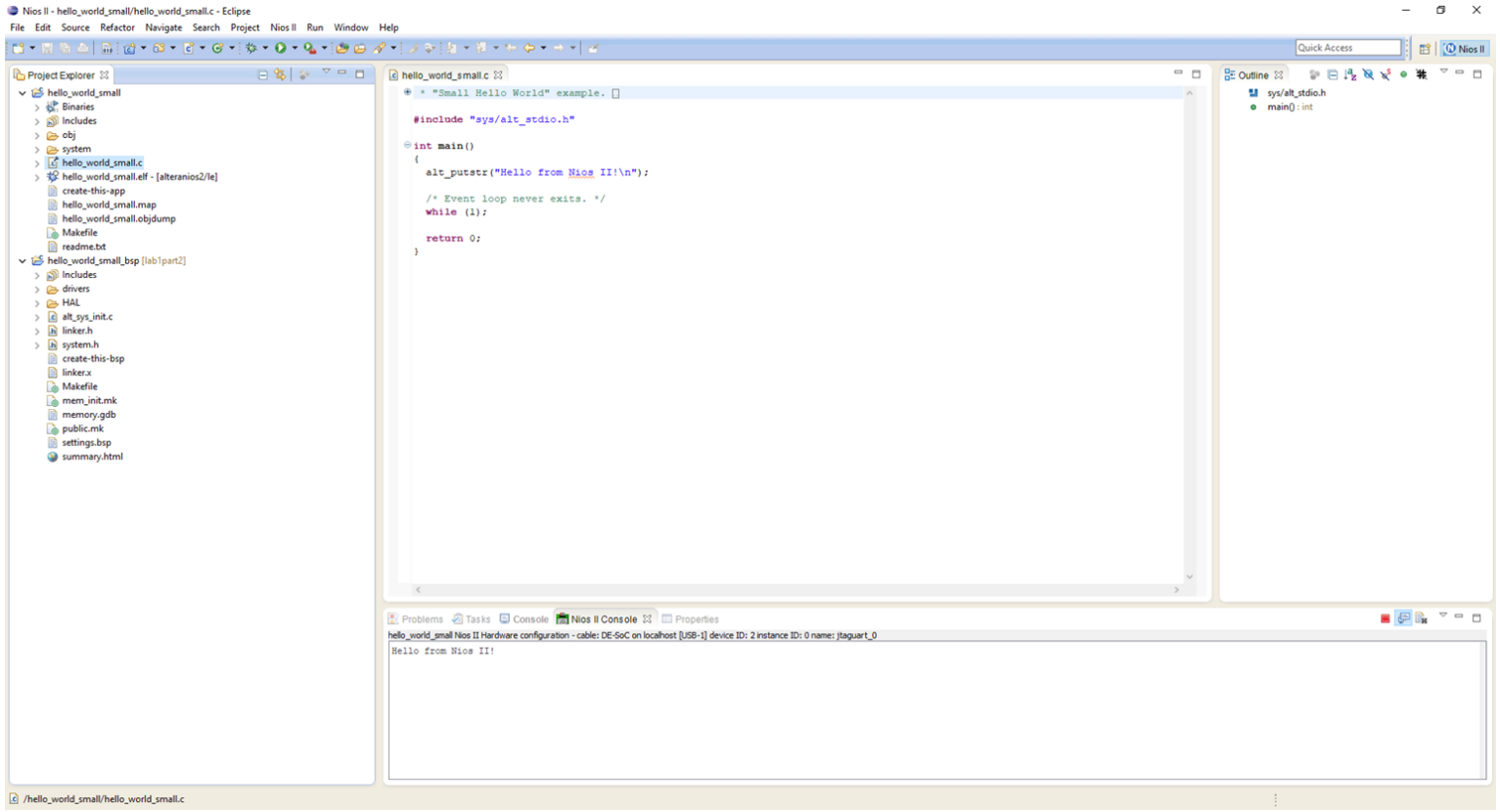
Should look like this when successfully programmed

Eclipse

- Tools -> NIOS II Software Build Tools for Eclipse
- Once Eclipse opens, go to File -> New -> NIOS II Application and BSP
- Name the project, load the sopcinfo file, and select the "Hello World Small" template
- **Please check to make sure you are using the correct sopcinfo file (check path and timestamp)**
- Right click on the project, click Run As -> Nios II Hardware
- It should build without errors and print a sentence to the console



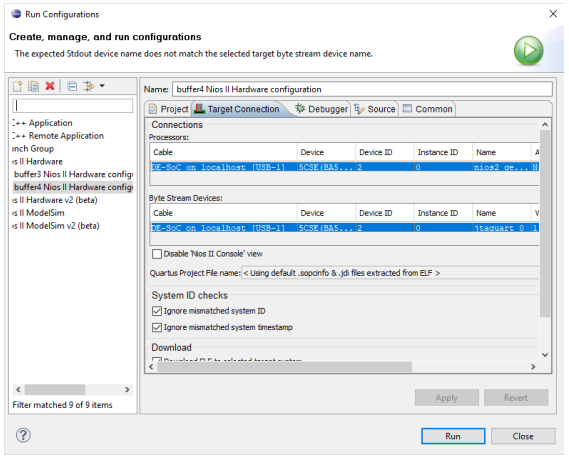
Eclipse

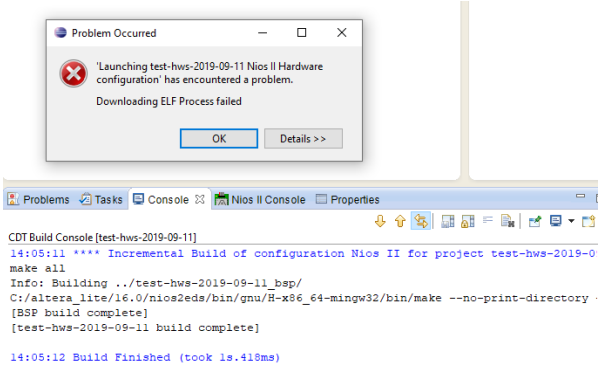


This is what Hello World Small should look like when completed!

Eclipse

- Note: To prevent a timestamp warning when running the code at a later point in time, do the following:
  - Click on the drop down by the green play button
  - Run Configurations -> Nios 2 Hardware Configurations -> System ID Checks
  - Select Target Connections tab and check both boxes:
    - Ignore mismatched system ID
    - Ignore mismatched system timestamp
- Note: If you get an error about Downloading ELF Process failed, check the position of your switches!





Programming

- Your task is to modify the provided code such that:
  - No buffer overflow should occur if both Student IDs are 8 digits or less, otherwise a buffer overflow should occur
  - Of course, in industry, a good programmer would "user proof" the code such that this sort of buffer overflow would not occur. But in this project, we are deliberately trying to make the buffer overflow occur under certain conditions to demonstrate how they occur.
  - The sum_id method should take in a Student ID, add up each individual number in the string, and return the sum
    - Example: sum_id(12345678) = 1+2+3+4+5+6+7+8 = 36
  - We would expect sum_id to compute the correct sum in case of no buffer overflow, but may return an incorrect sum in the case of a buffer overflow
  - During the demo, you will be asked to run your code twice, once correct result with no buffer overflow, once incorrect result with buffer overflow

Questions

- What is meant by Buffer Overflow?
- What would be a good example of a buffer overflow attack?
- How can buffer overflow be prevented in a program?
- Explain how buffer overflow occurs in your code using screenshots of the code as well as the results shown on the terminal.

Submission

You will need to submit answers to all questions in the form of a PDF document by the deadline.

Please include the following:

- Name of all team members, name of lab.
- Answers to questions in proper order and in paragraph form.
- Screenshots of your code to support your answers.

In addition, you will need to demonstrate that your code works properly to the instructors.

- The Demo timeslots will be announced later in the Moodle.
- Please sign up for a demo time in advance once it is available. Arrive at least 15 minutes before your scheduled time to set up your demo. All team members need to be present.

Best of luck!