CSC B58 – Computer Organization
Lab #2 – Binary arithmetic and binary adders

Before we start working, here's a quick reminder of the rules in effect while in the Embedded Systems Lab.

In-lab rules:

We have limited space, and you will be working with sensitive electronic components. Therefore:

- ***No food or drinks are allowed inside the lab.*** Be sure to eat properly before your session, you can step out for water if you need to. Notify your TA since he/she is the only person that can grant you access to the lab.
- ***No laptops, notebook computers, tablets, or other electronic gizmos*** allowed during the session.
- ***Bags and backpacks must be stored at the front or back of the room***.
- ***While in the lab, you're responsible for taking good care of all equipment***.
- ***Treat everyone else in the lab with respect and consideration***.
- ***You must listen and follow all instructions provided by your TA***.

How your work will be marked:

Your TA will observe your work during the three hour period.

- 20% of the grade for this lab is given by attending the session and working hard .
- 25% is given for completing the work due ***prior to the start of the lab***.
- 55% is given for completing the work requested during the lab session and having your TA verify it performs as expected.

While each member of the team must individually show the TA they have completed the work due at the start of the lab. Each team will hand-in a ***single*** *completed handout.*

If you encounter any problems with the software or hardware, bring this to the attention of your TA immediately. If no solution can be found quickly, your TA may have to have you join another team for the duration of the session.

CSC B58 – Computer Organization
Lab #2 – Binary arithmetic and binary adders

This lab is designed to allow you to experiment with binary arithmetic, and with implementing Simple binary operations using Boolean logic. You will build a very simple adder circuit, and study how different circuits can be used to carry out the same computations.

By now, you should have some experience using the Quartus II software and programming the FPGA board. But if you run into problems, do not hesitate to ask your TA for help.

Learning Objectives:

You will learn to implement full-adders from half-adders

You will learn to implement circuits with selectable functions

You will learn to create modules you can reuse in the Quartus software

You will understand how binary arithmetic works within a computer

Skills Developed:

Implementing circuits that perform computations

Packaging independent modules to reuse in larger circuits

Programming and testing complex designs on the FPGA boards

Reference material:

This handout. ***Remember to hand one completed handout to your TA at the end of the session***

The Quartus II handbook (*http://www.altera.com/literature/hb/qts/qts_qii51008.pdf*)

The DE2-115 user manual (Google “DE2-115 User Manual”)

Your lecture notes on Boolean Arithmetic and simple circuits

CSC B58 – Computer Organization
Lab #2 – Binary arithmetic and binary adders

Student Name 1 (last, first):

Student Number 1:

Student Name 2 (last, first):

Student Number 2:

Lab session date and time:

We understand that we are responsible for handling the equipment carefully and preserving it in working order.

____
(student's signature 1)

____
(student's signature 2)

You must complete all parts marked as 'prior to the lab session' before your appointed lab section starts.

Part 1 – Building a circuit to add binary numbers. The simplest functional block you could find in an Arithmetic and Logic Unit (ALU) inside a CPU.

First a brief reminder of how to create a new project on Quartus II – For the third lab session I will assume you already know how to do this.

1) Carefully unpack the FPGA board and place is safely on the desk. Connect the power cable to the board and a desk outlet, connect the USB cable to the board's USB 'blaster' port and to a computer USB terminal. ***DO NOT POWER UP THE BOARD yet.***

2) Open the Quartus II software. Close the initial information box. Create a new project from ***File -> New Project Wizard***
Create a directory for the project ***on the Desktop OR a USB drive***, with name ***Lab2_solution***
Set the name of the project to ***Lab2_solution***

Pressing '***next***' at this point would get you advanced configuration options which we will not need for this lab.

Click on '***Finish***' when done.

3) ***To complete prior to the lab session:***

Show below the circuit diagram for a ***Full-Adder*** built from ***2 half-adders***. Make sure your circuit Does the right thing on paper! Otherwise you will not be able to complete your work in the lab.

Circuit diagram

Truth Table for the circuit

Cin	***a***	***b***	***Cout***	***S***

TA check: Students completed this prior to the session

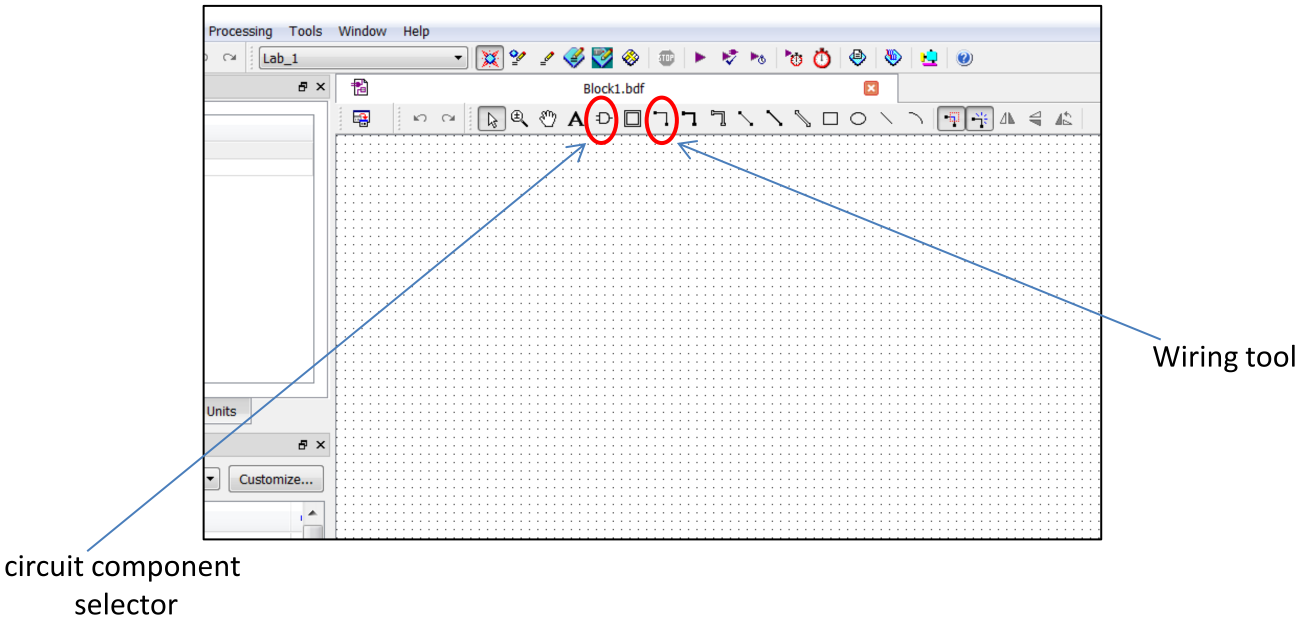
Student 1: ☐

Student 2: ☐

4) Use the Quartus II graphical editor to build a ***half-adder***.

To start the graphical editor, use ***File -> New -> Block Diagram/Schematic File***

The graphical editor allows you to select and connect all sorts of logic functions and circuit components



- In the schematic diagram, implement the circuit for a ***half-adder***. This should be reasonably quick. Be sure to add ***inputs and outputs*** for the input bits, the sum, and the carry out. ***Be sure to name the inputs and outputs appropriately***.

- Once you have completed your circuit. ***Have your TA take a look and check your work.***

Save the file as: ***Half_Adder.bdf*** - Make sure to save this in the project's directory
Select: ***Processing → Analyze current file***

- Once the file compiles successfully, choose:

File → Create/Update → Create Symbol Files for Current File

This should create a ***Half_Adder.bsf*** file within the project directory.
From this point on, you can import the half adder as a circuit block!

4) (continued).

You may receive an error message indicating that the file Half_Adder.bsf is read-only

This is a BUG in Quartus. There is NO file Half_adder.bsf at this point

To fix this problem:

Navigate to the project folder (from the file browser in Windows, not from Quartus). Once inside the ***Lab2_solution*** folder:

Mouse Right-Click → New → Text Document

and create an empty text document called ***Half_Adder.bsf***. Please ***ensure*** that the name is just ***Half_Adder.bsf*** and ***not Half_Adder.bsf.txt***.

Once you have done this. Create the symbol files again from Quartus. This time the software will ask you whether you want to overwrite the existing file. Accept, and your symbol file will be stored.

5) Implement a ***full-adder***.

- Create a new block-design file.

- Use the ***circuit component selector*** to select your newly created ***half-adder*** module, and use two ***half-adders*** plus logic gates to create a ***full-adder***.

Browse to the project directory and select ***Half_Adder.bsf***

or

Select: Tools → Options → Libraries → (select "…" next to Project Library Name)

and add the project directory to the libraries
(this will make any modules you create appear in the circuit selection window, along with Altera's components)

- Save your full adder circuit as ***Full_Adder.bdf***. Once more, ensure this file is saved inside the project directory.

5) (continued).

- Once you have completed your full-adder. ***Have your TA check your work.***

Compile ***only the full-adder*** design file to ensure there are no errors

Then ***create design files*** just as you did for the half-adder.

Once more, you may receive a file permission error from Quartus.
You know how to fix it

6) ***To complete prior to the lab session:***

Using ***full-adder*** blocks, logic gates, and ***2x1 multiplexers***. Design a circuit that

- Has ***8 inputs*** representing ***2, 4-bit numbers***: ***[a3 a2 a1 a0] and [b3 b2 b1 b0]***
- Has an additional input called ***function_select***.
- When ***function_select='0'***. The circuit computes ***a+b***
- When ***function_select='1'***. The circuit computes ***a-b*** using ***2's complement*** arithmetic
- The circuit has ***a 4 bit output for the result: [r3 r2 r1 r0]***
- The circuit has one additional output called ***overflow***

Show the ***block diagram*** for your circuit. ***DO NOT SHOW*** the gates within full-adders. Your circuit will need a way of obtaining the ***2's complement of b*** when the function selector is set to '***1***'. This may involve more ***adders***. As a reminder, a ***2x1 mux*** has ***2 inputs, 1 output, and 1 control line***. The control line selects which of the inputs is connected to the output. ***Use blocks for all components unless they are individual logic gates.***

Use the back of this page for the circuit.
Your TA must check your diagram before you implement anything!

TA check: Students completed this prior to the start of the lab session

Student 1: ☐

Student 2: ☐

7) Implement the circuit you designed. ***If the team members came up with different designs, discuss the designs together and choose the one that both agree is best.***

- *You will need to implement a **2x1 multiplexer***. Create a separate block-design file, implement it (it's only a couple gates!). Save it as ***2x1mux.bdf***. And create the ***design files*** just like you did for the ***half-adder*** and the ***full-adder***

- Create a new block-design file called ***Lab2_solution.bdf***
  Use your components along with logic gates to implement your circuit from 7).
  Be sure to add input and output pins as required.

- Once you have fully implemented your circuit. ***Have your TA check your work.*** Save all your design files. Then ***compile the entire project.***

You can not proceed until all errors have been resolved

- After errors are resolved, you will still have ***critical warnings***. One of which will tell you that pin assignments are missing. Proceed to the next page to set up the device model and assign pins to inputs and outputs.

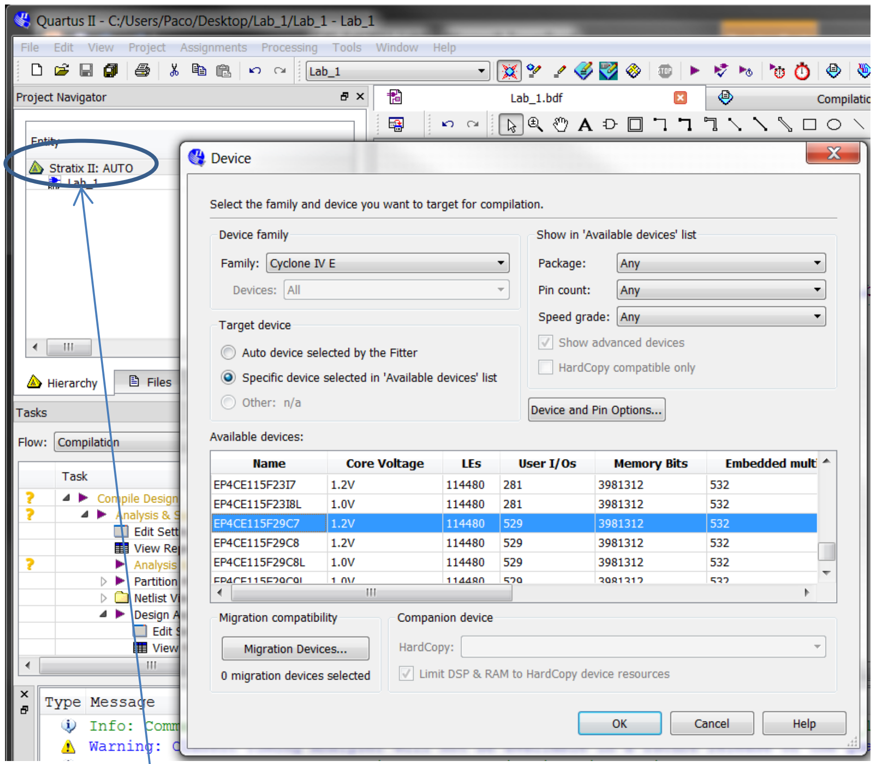
8) ***Do not start this part until you have a circuit that compiles successfully.***

This is exactly the same process you used in the first lab session to assign pins in the FPGA to inputs and outputs.

Be very careful here, choosing the wrong device will void your design and possibly cause problems when attempting to program the device.

First off, ensure you have selected the correct FPGA chip to program.

On the '***Entity***' window, ***right-click*** on 'Stratix II (AUTO)', select '***device***'.



Device

Select the family and device you want to target for compilation.

Device family
Family: Cyclone IV E
Devices: All

Target device
( ) Auto device selected by the Fitter
(•) Specific device selected in 'Available devices' list
( ) Other: n/a

Show in 'Available devices' list
Package: Any
Pin count: Any
Speed grade: Any
[✓] Show advanced devices
[ ] HardCopy compatible only

Device and Pin Options...

Available devices:

Name	Core Voltage	LEs	User I/Os	Memory Bits	Embedded mult
EP4CE115F23I7	1.2V	114480	281	3981312	532
EP4CE115F23I8L	1.0V	114480	281	3981312	532
EP4CE115F29C7	1.2V	114480	529	3981312	532
EP4CE115F29C8	1.2V	114480	529	3981312	532
EP4CE115F29C8L	1.0V	114480	529	3981312	532
EP4CE115F29C9L	1.0V	114480	529	3981312	532

Migration compatibility
Migration Devices...
0 migration devices selected

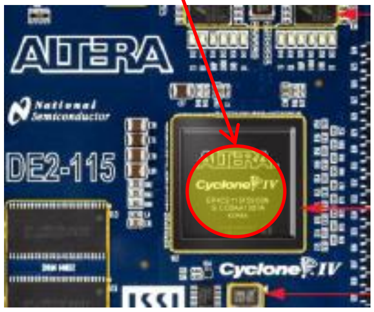
Companion device
HardCopy:
[✓] Limit DSP & RAM to HardCopy device resources

OK Cancel Help

Right-click here

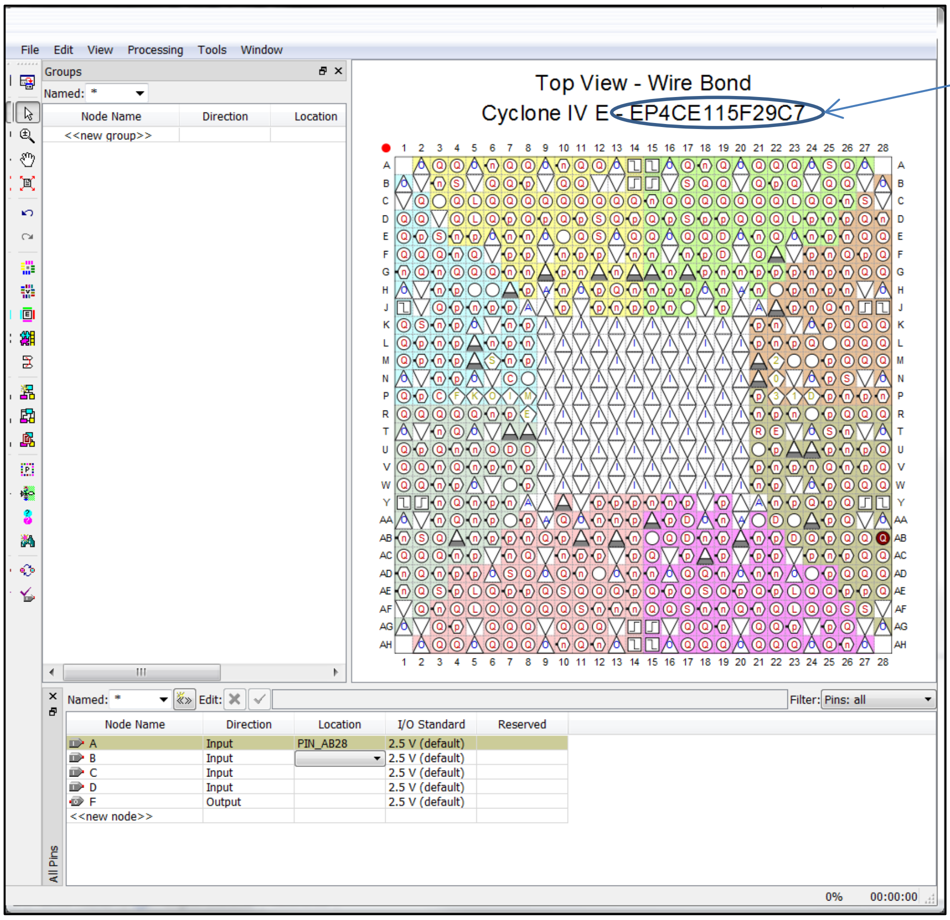
Select ***Cyclone IV E*** for 'family', click on '***Specific device selected…***' and choose **EP4CE115F29C7** as the device.

Double-check that the above matches the device type printed on the FPGA on the DE2-115 board



9) In Quartus II, go to ***Assignments -> Pin Planner***

If you selected the device correctly, you will see a pin diagram for the FPGA, along with a list of the input and output pins in your circuit.



Node Name	Direction	Location
<<new group>>		

Node Name	Direction	Location	I/O Standard	Reserved
A	Input	PIN_AB28	2.5 V (default)	
B	Input		2.5 V (default)	
C	Input		2.5 V (default)	
D	Input		2.5 V (default)	
F	Output		2.5 V (default)	
<<new node>>				

Check you have the correct device

Check the DE2-115 user user manual for the mapping of switches and LEDs to pins, this can be found starting on P. 35.

Complete and ***double check*** your PIN mapping, one helpful way to reduce errors at this stage is to have each member of the team complete/check the PIN mappings independently, and then look for disagreements.

PIN mapping table – record the pin names here

a0 : sw0	*b0 : sw4*	*r0 : ledr0*
a1 : sw1	*b1 : sw5*	*r1 : ledr1*
a2 : sw2	*b2 : sw6*	*r2 : ledr2*
a3 : sw3	*b3 : sw7*	*r3 : ledr3*
f_select: sw8	*overflow: ledg0*	

10) Programming the FPGA

Save your work, ***compile the project and fix any errors found at this stage***.

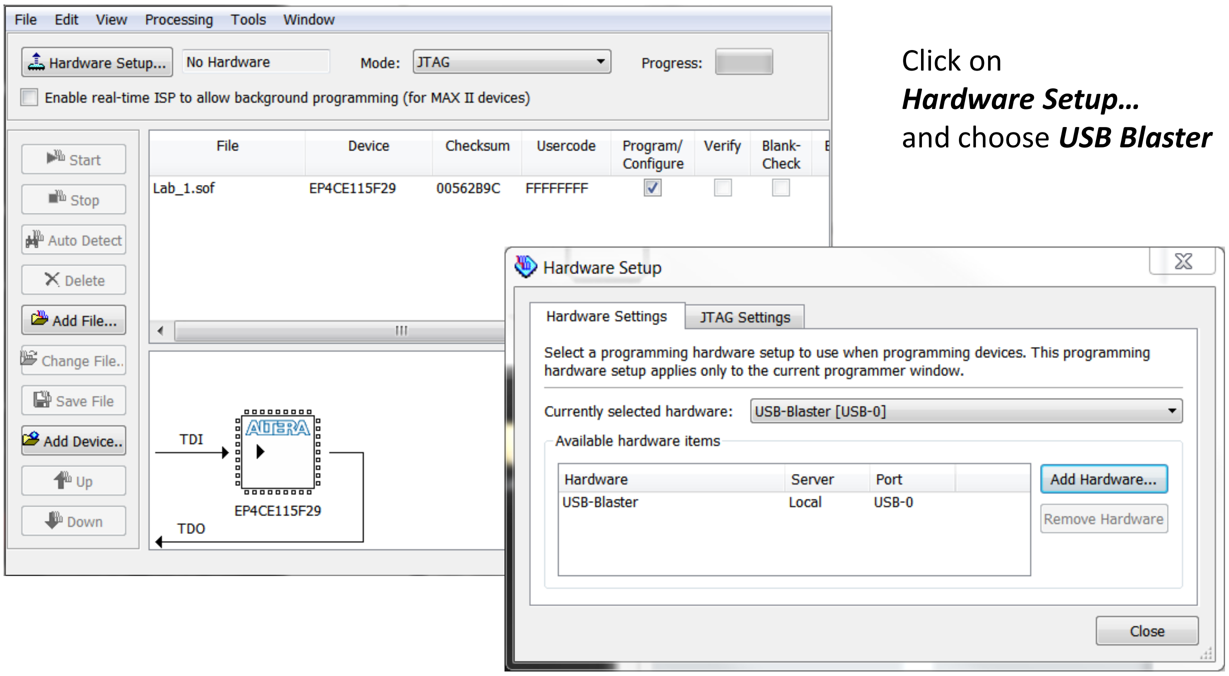
Turn your attention now to the DE2-115 board

- Make sure power and USB connections are set up properly
- Make sure the programming mode switch is set to '***RUN***' (see page 2)

Turn on the FPGA board by pressing the red power button.

You should see the welcome message on the LCD display, the numeric displays will be cycling digits, and the LEDs will be flashing. Wait a few seconds for the computer to recognize the board and load the appropriate drivers.

Now in Quartus II go to ***Tools -> Programmer*** which brings up the window shown below



No Hardware Mode: JTAG

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

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check
Lab_1.sof	EP4CE115F29	00562B9C	FFFFFFFF	✓		

Hardware Setup

Hardware Settings | JTAG Settings

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

Currently selected hardware: USB-Blaster [USB-0]

Available hardware items

Hardware	Server	Port
USB-Blaster	Local	USB-0

Click on ***Hardware Setup…*** and choose ***USB Blaster***

To program your design onto the FPGA board, press ***Start***.

After the operation completes, the DE2-115 should be working as ***your own little calculator!***

11) Testing

Now proceed to test your mini electronic-calculator.

Complete and check the results for the additions shown below. ***You must call your TA to check your circuit's operation and show it works in all cases.***

With ***function_select*** set to ***0*** (add)

a3 a2 a1 a0	b3 b2 b1 b0	r3 r2 r1 r0	Ov
0 0 1 1	0 1 0 1		
1 0 1 0	1 0 1 0		
1 1 1 1	0 0 0 1		
0 0 0 0	1 1 1 1		
1 0 1 0	0 1 0 1		

With ***function_select*** set to ***1*** (subtract)

a3 a2 a1 a0	b3 b2 b1 b0	r3 r2 r1 r0	Ov
0 0 1 1	0 1 0 1		
0 0 1 1	1 0 1 1		
1 1 1 1	0 0 0 1		
0 0 0 0	1 1 1 1		
1 0 1 0	0 1 0 1		

(To be checked by TA) ☐
Function verified in-lab

12) Wrapping up (10 minutes before the end of the session)

Power-down and carefully store the DE2-115 board, bring the boards and cables to the storage locker your TA will indicate.

Locate the ***Lab2_solution*** folder and create a .zip file called "***Lab2_solution.zip***" by selecting within that folder ***ALL THE DESIGN FILES*** (i.e. all files with extension ***.bdf, .bsf, and .qpf***) but nothing else. ***DO NOT*** include anything else, and in particular, ***DO NOT*** add the ***db*** or ***incremental_db*** folders.

e-mail this .zip file to the ***course instructor*** with the following subject:

'***CSCB58, Lab 2'*** (without the quotes!)

Inside the email, note the name and student numbers of each team member.

Don't forget to attach the .zip file!

13) Post-session feedback. Feel free to use simple yes/no answers, or to add comments where you feel they are relevant. Use the back of the page if needed.

- Did you complete the lab exercise in the 3-hour allotted time span? (if no, how much more time do you think you would have needed?)

- Did this exercise improve your understanding of how computation is carried out within digital computers? (yes/no and a brief comment please)

- Did this exercise improve your understanding of 2's complement arithmetic and how it can be used to handle signed quantities? (yes/no and a brief comment please)

- Any general thoughts or comments you want to add about this lab session:

TA check: FPGA returned with all cables and properly packed: ☐