# 3C7 Digital Systems Design

# Assignment 1

## Aim:

The purpose of this assignment is to bring together elements from Lab A-E inclusive, and give you experience with developing and testing a larger combinational design and implementing it on the Basys-3 Board. This assignment is worth 25% of the overall 3C7 mark.

## MINI ARITHMETIC LOGIC UNIT (MINI-ALU)

You need to **design, write/modify** the Verilog modules for the following functions, and test a "mini" arithmetic logic unit. An **arithmetic logic unit (ALU)** uses combinatorial logic to implement common arithmetic and logical functions. A typical ALU has a wide range of functionality from **addition to bit shifting**. Your ALU will provide a **narrow range of functions** performed on **two 6-bit inputs A and B**. A and B are **in 2's complement format**. The **output of the ALU is a 6-bit number** X, also **in 2's complement** form as appropriate, and the input **fxn** controls the output as follows:

| fxn | X[5:0] |
|---|---|
| 000 | A |
| 001 | B |
| 010 | -A |
| 011 | -B |
| 100 | A<B (is A less than B) |
| 101 | (A *nxor* B) (Bitwise XNOR) |
| 110 | A+B |
| 111 | A-B |

## Instructions:

1. You must re-use code from blocks previously implemented in 3C7, i.e. for the A+B output you shouldn't simply have a block that implements "output = A+B". You MUST use the carry-ripple adder you designed and tested, modified only if required.
2. Decide an approach to partitioning the design. Identify modules that you can use "off-the-shelf" and what new modules you must write. Modules should be easily tested units with a clear function (or small, clearly defined, set of functions).
3. Some of your modules are not exactly what you need (think of your 8bit-greater than or equals circuit). Think about how you can use it though **WITHOUT** modifying it (i.e., by not reducing the input width to 6-bits)
4. **Your top level ALU module should only contain instantiations with NO logic in that block.**
5. Include a functional diagram in your write-up, clearly showing the hierarchy of your design and how modules are all connected. Wire names should reflect what is implemented in your design. (You can use Vivado to do this)

6. Note any modifications required for existing blocks and modifications made to existing testbenches.
7. Design a top level testbench that instantiates your ALU module for testing. Design an appropriate set of test-vectors to test your ALU module and its full range of functionality.
8. For each function, you need to show a legible, well-structured waveform (i.e. relevant signals visible) that clearly demonstrates the correct operation of the ALU.
9. For any function that is not working, provide ideas on why.
10. Comment on how well tested each function is and any shortcomings in your test strategy.
11. How robust is your design to unexpected inputs?
12. Once you are satisfied the module is working, target your final ALU design onto the FPGA board. Use the peripherals of the board to control the inputs and display the output.
13. **Include the final .bit file in your submission.**

## Submission:

You must submit the following items via Blackboard:

1. **Full write-up** named Assign1_surnameinitial.pdf, e.g. called Assign1_shankers.pdf. (**Must be in pdf format**)
    a. **Include waveforms** in the write-up to demonstrate the design is working where appropriate
    b. **Include a screen grab** of your Vivado environment that clearly shows the hierarchy of the design. Include also a screen grab of your file directory structure that clearly shows those same files on your computer, e.g. in windows explorer.
    c. **Include a section called "Demo"** which clearly outlines how to demo your design on the board (which switches etc to use for inputs etc). This will be verified against your submitted bitfile.
    d. **Include previous submissions** for the adder and greater-that circuit as appendices. (if you did not submit anything, you won't be able to do this)

    **NOTE:** Use the **first link** (marked Assignment-1 writeup) to submit the pdf file.
2. **A zip file** named Assign1_surnameinitial_codes.zip, e.g., called Assign1_shankers_codes.zip containing the following
    a. **code for the modules and testbench** you have written such that a 3rd person can recreate your *bitfile*.
        i. **Note**: all code MUST be suitably commented, i.e. to a very high level.
    b. **Your final bit file** used for the Demo section of the report.

    **NOTE:** Submit the zip file using the **second link** (marked Assignment-1 code).

Deadline: **Sunday, 31th October 11:00 pm** via Blackboard.

Late Submissions – half the marks will be lost for up to 1 week late. No marks after this.

## Plagiarism

Plagiarism is interpreted by the University as the act of presenting the work of others as one's own work, without acknowledgement. Plagiarism is considered as academically fraudulent, and an offence against University discipline. The University considers plagiarism to be a major offence, and subject to the disciplinary procedures of the University.

Visit http://tcd-ie.libguides.com/plagiarism

All students must complete the TCD Ready Steady Write plagiarism tutorial and sign a declaration when submitting course work, whether in hard or soft copy or via Blackboard, confirming that you understand what plagiarism is and have completed the tutorial. If you read the information on plagiarism, complete the tutorial and still have difficulty understanding what plagiarism is and how to avoid it, please seek advice from your College tutor, your Course Director, your supervisor, or from Student Learning Development.

**e.g.** having exactly the **same testvectors or module** as another person **is plagiarism**.

Plagiarism will result in loss of **ALL** marks for this assignment.