3C7 Digital Systems Design

Lab Session C

Aim:

The purpose of this session is to simulate and explore an arithmetic circuit using combinational logic in Vivado. It will also revise setting up projects in Vivado, simulating, and using the waveform viewer.

Learning Outcomes:

On completing this lab session, you will be able to:

- Predict carryout and overflow in 2's complement addition and subtraction
- Implement a carry ripple adder

Carry Ripple Adder/Subtractor

A single full adder stage is shown in the diagram below. You are given code for this basic module. You will cover this is Lecture 4. Refer to your notes to see how to construct a 6-bit carry ripple adder/subtractor from this building block. The block should take in two 6-bit numbers in 2's complement form and output a 6-bit sum, the MSB carry-out and a signal detecting overflow. The **`sel`** input will control the add/subtract function. Refer to the details below for complete instructions for this exercise.

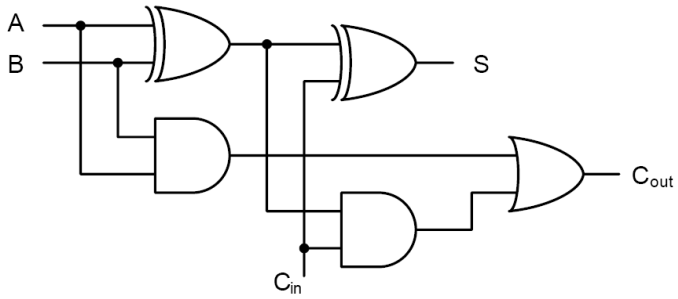

Diagram: Full adder stage

Instructions:

1. Start a new project (at a local location – say C:/temp/student_id/LABC)

2. Add the file ***full_adder.v*** to the project and create a new file ***6bit_ripple_adder.v***

3. This new module will be a 6-bit ripple carry adder built from the full adder as a structural building block. The module must have ONLY the following ports:

Port name	Type	Description
`x`	input	6-bit 2's complement number to add
`y`	input	6-bit 2's complement number to add
`sel`	input	Select functionality for add/subtract
`overflow`	output	Output flagging overflow in the sum output
`c_out`	output	MSB Carry out from the sum
`sum`	output	6-bit 2's complement sum of x and y

4. In your submission, draw a functional block diagram for ***6bit_ripple_adder.v***. (**Note:** Draw the diagram based on your design, not export it from the Vivado hierarchy). Clearly label the inputs and outputs of the module and wiring connections. The names you use in this diagram should match those in the description above and in your Verilog. You can use an online tool like draw.io to draw the diagram.
5. Using this functional diagram as the basis of your design, write the Verilog description for the adder, instantiating full adders as required, with appropriate wiring to connect modules, plus any necessary logic for overflow detection.

6. Using previous testbenches as a template, exercise the block with a suitable range of both positive and negative inputs. You should consider have you covered all the following combinations or cases:

Inputs x and y	Sel	Function	Overflow	Carryout
Both positive	High/low	Sub/Add	Present/absent	High/low
Both negative	High/low	Sub/Add	Present/absent	High/low
One positive/one negative	High/low	Sub/Add	Present/absent???	High/low
All zero	High/low	Sub/Add	??	??

7. Think whether all of the above actually make sense. Add more test cases to increase your coverage. Using the table above, include in your write up a summary of how many of each combination you have managed to include. Comment on cases that are not possible etc.

8. To help you, write a list of the inputs you are sending to the DUT and the ***expected*** outputs in a table with a format similar to below.

Test Vector ID	sel	x	y	c_out expected	Overflow expected	Sum expected
1	0	6'b00_0001	6'b00_0100	1'b0	1'b0	6'b00_0101
2	0	6'b00_0000	6'b00_0000	1'b0	1'b0	6'b00_0000

Example table for your own work

9. Run your simulation and then extend this table to include the ***observed*** outputs. Indicate whether each test has passed or failed. In your report, include a discussion where you address whether:
    - Your circuit has passed all your tests
    - Why your circuit has failed any particular test
    - How extensive your range of test vectors is?

Test ID	x	y	sel	Cout exp	Overflow expected	Sum expected	Cout ob	OFlow obs	Sum observed	Pass or Fail
1	6'b00_0001	6'b00_0100	0	1'b0	1'b0	6'b00_0101	1'b0	1'b0	6'b00_0101	Pass
2	6'b00_0000	6'b00_0000	0	1'b0	1'b0	6'b00_0000	1'b0	1'b0	6'b00_0100	Fail

Example table to include to help you track your expected outputs

10. In your report, include a waveform clearly showing all your test cases. Make sure any waveforms included are legible. It will probably make sense to use multiple waveforms, each showing a small number of test cases.

11. Ensure that ALL code is COMMENTED!!

12. You MUST use the signal names in the specification

Overflow:

13. Choose one of your test inputs that will cause overflow.

14. Zoom in on this particular test in your waveform.

15. Delete all unnecessary signals in the waveform window leaving only the internal signals in the ripple adder module, i.e.:
    - Inputs x, y and sel
    - Outputs sum, overflow and c_out
    - Internal carry values between the full adder sections.

16. Analyse the waveform and by working through the addition on paper, show that all the internal signals are correct.

Submission:

You must submit the following parts, contained in a single PDF, via Blackboard:

- Labelled waveforms demonstrating the design is working where appropriate
- Table of testvectors, including expected and observed outcomes (as in the example in step 9)
- Code for the adder and testbench you have written

All code MUST be suitably commented

Name the file LabC_surnameinitial.pdf, e.g. LabC_shankers.pdf.

Deadline:

Group A (Tuesday labs): Tuesday 21st February 2023.
Group B (Friday labs): Friday 24th February 2023.
All submissions are via blackboard.

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.