Simple Computation

Ben Heard

Due: February 22, 2023

Abstract

In this assignment you will be designing and implementing a system that accepts four 4-bit numbers, performs the computation $(A + B) - (C + D)$, and provides the result as output.

1 Introduction

In this assignment you will be designing and implementing (in Verilog) a system that accepts four 4-bit numbers on its input, computes the value for $(A + B) - (C + D)$, and provides the result as output.

1.1 Top Level Interfaces

At the top level the signals from Table 1 shall be connected.

Signal	Direction	Description
rst_n	INPUT	An active low synchronous reset
clock	INPUT	A freerunning clock
d_in	INPUT	A 4-bit unsigned input
start	INPUT	A single bit start indication
result	OUTPUT	A 5-bit unsigned result as output
valid	OUTPUT	A single valid indication

Table 1: Top Level Ports

1.2 Output Interface

Once your system has calculated the result it will assert the **valid** signal along with the **result** indicating that the result may be captured by another system.

1.3 Input Interface

The input interface is a little more complex. The four inputs for **A**, **B**, **C**, and **D** will appear on the input, **d_in**, in four consecutive cycles beginning with the clock edge in which **start** is seen.

To note, this means that **A**, **B**, **C**, and **D** will all come in on the same **d_in** input, just at different times. The times are 4 consecutive clock edges.

1.4 Requirements

- Naming Requirements
    1. The Verilog module shall be called **proj001**. This is just the name of the module, the file(s) may have any name you choose.
    2. The implementation shall have an active low synchronous reset input called **rst_n** that, when asserted, will reset the entire system to its initial state.
    3. The implementation shall have a free-running clock input called **clock**.
    4. The implementation shall accept a 4-bit input called **d_in** that will be an unsigned quantity.
    5. The implementation shall accept a single bit input called **start** that provides indication that the value on **d_in** is the value for the variable **A** and that the following 3 values or **d_in** represent the variables **B**, **C**, and **D**
    6. The implementation shall provide a 5-bit output called **result** that represents the result of the computed function.
    7. The implementation shall provide a single bit output called **valid** that provides indication as to when the **result** may be captured.
- Data Requirements
    1. The output, **result** shall be an unsigned quantity.
- Timing Requirements
    1. An active low synchronous reset will be provided to your system at the beginning of operation to initialize the system to a known state. This will be asserted to the **rst_n** port of your system.

2. To facilitate grading, the **result** and **valid** must be asserted within 10 clock cycles of the last operand being provided to your system. Any clock period within the 10 clock cycles will be sufficient.
3. The **valid** signal shall assert for a single clock cycle.

- Implementation Requirements

1. The implementation shall be any combination of structural or dataflow, with the exception of the adders and subtractors. The single bit component of the adder/subtractor may be dataflow but the wider adder/subtractors shall be structural combinations of the base components. (Think carry-ripple).
2. The top level module shall instance a datapath and a controller, structurally, connecting them to the inputs, outputs and each other as necessary.
3. The core D Flop Flop shall be the one provided by the instructor.

- Git Submission Requirements

1. The completed assignment shall be pushed to your class Git repository at the NCSU GitHub. The SHA for the submission shall be provided in the Moodle assignment.
2. The directory projects/project_001, created for you in your repository, shall contain your submission.
3. There are no naming requirements for the files that are submitted. Only that the files be text files with a .v extension and placed in the top level project_001 directory.

2 Notes

2.1 Input Details

- No new values for **A**, **B**, **C**, or **D** will be provided until **valid** is seen asserted on your output.

- The testbenches will ensure that the result of $(A + B) - (C + D)$ will always be positive or zero; there is no need to check for whether a result goes negative.