NC State University
Department of Electrical and Computer Engineering

ECE 310 – Fall 2025

Project 2

Summary

In this project, you will be designing and implementing a system that accepts 8-bit unsigned numbers as inputs, performs the operation $(A + B) - (C + D)$, and provides the result as an output.

1. Introduction

The computation will be implemented using any mix of gate-level and dataflow modeling, **no behavioral modeling is allowed (aside from the DFF model)**.

1.1 Top-Level Interfaces

At the top level (module **Project2**), signals given in Table 1 will serve as port connections.

Signal	Direction	Description
reset_n	input	Active-low synchronous reset
clock	input	Free-running clock
d_in	input	8-bit unsigned input
op	input	2-bit operand selector
capture	input	1-bit capture indicator
result	output	9-bit unsigned result as output
valid	output	1-bit valid indicator

Table 1: Top-Level Ports

1.2 Output Interface

Once the model calculates the result, it will assert the **valid** signal along with the result indicating the result may be captured by another module.

1.3 Input Interface

The input interface is more complex. The input will be accepted on the positive/rising edge of the clock only when **capture** signal is High. There is also an additional selection input (**op**) that provides an indication as to which operand is being provided, specifically whether **A**, **B**, **C**, or **D** is being provided to the input port **d_in** during **capture**. The mapping between **op** and the operands is shown in Table 2.

op	operand
2'b00	A
2'b01	B
2'b10	C
2'b11	D

Table 2: Mapping between **op** and operands

The above implies that all inputs will be coming in on the same line, just at different times and the model must differentiate between the inputs. This also allows the 4 input values to arrive in any order, not necessarily in alphabetical order. Your controller and datapath need to ensure that the correct values are used as the correct operand. Once all 4 values are captured, the system is then to compute the required function above and provide the result (project needs logic to account for this).

1.4 Requirements

- Naming Requirements:
  - The Verilog module shall be named **Project2**. This is the name of the module, your file(s) may have any name you choose.
  - The project shall have an active-low synchronous reset input named **reset_n** that will reset the entire system to its initial state (load zeros) when asserted. This input will be asserted once one computation is finished (after output is read once **valid** is asserted).
  - The project shall have a free-running clock input named **clock**.
  - The project shall accept a 8-bit input named **d_in** that will be an unsigned number.
  - The project shall accept a 2-bit input named **op** that indicates the operand being provided on input **d_in**.
  - The project shall accept a 1-bit input named **capture** that provides an indication that the value on **d_in** is ready to be captured, where there will be at least 0 clock cycle delay between inputs.
  - The project shall provide a 9-bit output named **result** that will represent the result of the computed function. There can be a 1 cycle delay between the last input being captured and the result being available.
  - The project shall provide a 1-bit output named **valid** that provides an indication as to when the **result** is available/may be captured. The **valid** signal shall be asserted for 1 clock cycle only after output is available. Output **valid** may be high for more than 1 cycle as long as the **result** is stored.
- Data Requirements:
  - The input (**d_in**) and output (**result**) shall be unsigned quantities. The input and output values will always be non-negative (0 and above).
- Timing Requirements:
  - An active-low synchronous reset (**reset_n**) will be provided to your system at the beginning of an operation to initialize the system to a known state.

- To help with grading, outputs **result** and **valid** must be asserted within 10 cycles of the last operand being provided to your system. Any clock period within 10 clock cycles will be sufficient.

- Implementation Requirements:
  - Implementation shall use any combination of structural/gate-level or dataflow, except for the adders and subtractors. The single bit component of the adder/subtractor may be dataflow, but wider adder/subtractors shall be structural combinations of the base component. Lab 2 will be helpful here.
  - The top-level module shall instantiate a datapath and a controller modules, structurally, connecting them to the inputs, outputs, and each other as necessary.
  - The core DFF shall be the one designed in class and modified as needed to accommodate any changes (such as reset, multi-bit, etc) as other module(s). This is the only behavioral model allowed to be used in the project.

2. Additional Information

- Although the inputs can be provided in any order, an input will only be provided once.
- No new values for **A**, **B**, **C**, or **D** will be provided until **valid** is seen asserted on the output. Once **valid** is asserted, the result will be read by the testbench, the system will be reset (using **reset_n**), another set of inputs will be provided, and the operation will repeat.
- The testbenches will ensure that the result of the operation will always be positive or zero (non-negative), there is no need to check whether the result of the entire operation goes negative.
- **You are not allowed to use ChatGPT (and similar sources) to produce your code.** While these tools may help optimize and check your code for correctness, this shall be the only acceptable use for this project. Any code that has been processed using any AI tools shall be clearly delimited in your .v file(s) and a summary of pre- and post-tool versions shall be present in your code as well as your report. Additionally, a description of each non-obvious functional block of code processed through these tools shall be present.

Submission Details:

1. Similar to Project 1, there is a Project 2 Design submission (see Moodle for due date). Identify all units present as part of your solution to this project in as much detail as possible. Blackbox designs are acceptable for some units as long as a detailed view of each blackbox is separately shown in the submission. If the hardware for a given part cannot be readily identified, provide as much information on the part as to allow the reader to understand what hardware would eventually substitute the text. Break down the design in terms of datapath (generally combinational) and controller (generally sequential) parts. The more details provided in the design, the more feedback can be received and more points can be awarded.
2. Submit 1 .zip file of your project folder (the entire folder, not just the .xpr or .v files). On Windows, go to the folder containing all projects, right-click on your Project 2 folder, select "Compress to ZIP file". This archive will contain your source code for the model file (under **\... .srcs\sources_1\new**) and testbench (under **\... .srcs\sim_1\new**), as well as a variety of

other folders and files. **Do not include the tesbench in the source code file, it must be a separate file.**

3. Submit 1 .pdf file with a clear waveform running through a few examples of inputs. **Ensure the waveform has enough resolution to be visible for your test inputs and outputs** (waveforms can be zoomed in and shifted to clearly show the values used). Justify the inputs used for testing your code and provide an analysis of the results. You can also include a short description of code, approach, code snippets, etc, in case there are issues with compiling your code and to better understand your implementation of the project. Use an approach for the report similar to Project 1. Up to 10% extra credit will be awarded for using LaTeX to typeset your report, depending on the amount of LaTeX elements used in the report.

Points Breakdown:

15 pts: Diagram of architecture drawing with datapath components and control structures identified (color-coded or outlined). Ensure individual units are shown with enough detail to be described at the dataflow (or structural) level, depending on requirements. See Moodle for due date.

10 pts: Design of adder(s) and subtractor(s)

7 pts: High-level module instantiation (module **datapath** and module **controller**). Top module **Project2** body can only have 2 lines of code for instantiating the 2 modules above in addition to additional signal declarations inter-connecting these 2 modules.

20 pts: Datapath design (design of muxes, DFF, and other units as needed, including logic). Computation operation, data and timing requirements met. Module name must be **datapath**, other module names are up to you.

20 pts: Controller design (design of muxes, DFF, and other units as needed, including logic). Computation operation, data and timing requirements met. Module name must be **controller**, other module names are up to you.

15 pts: A .pdf report detailing the design structure and approach, test vectors (minimum 4) and reasoning for choices made, any specific description of code as necessary, expected results vs timing diagram, discussion, and anything else relevant to this project. The more formal the report, the more points will be awarded. If the report is written using LaTeX, up to 10 additional points will be awarded, depending on the use of LaTeX elements.

10 pts: Up to 10 individual tests passed

3 pts: Correct upload to Moodle (1 .zip with proper folder structure) and Gradescope (1 .pdf file report, as described above)