Summing Up Subtraction: A Dual-Purpose Design Challenge in Verilog.

Course Information:

Fall 2023

Main Instructor: Zahra Sadat Shariatmadar Mortazavi

Contact Via: zshariatmadar@aut.ac.ir

Main Teaching Assistant: Danial Fathpour

Contact Via: dfathpour@aut.ac.ir

Project Assistant: Seyyed Mohammad Hamidi

Contact Via: 1380hamidi@gmail.com

Introduction

In the realm of digital design, the ability to describe and simulate hardware using high-level languages has revolutionized the way we approach the creation of digital systems. Among these languages, Verilog stands out as a pivotal tool for both academia and industry. Verilog, a hardware description language (HDL), allows designers to describe the behavior and structure of digital systems in a format that is both human-readable and machine-simulatable. This dual nature provides a bridge between abstract design concepts and their tangible realizations in hardware.

The importance of Verilog cannot be overstated. As digital systems grow in complexity, the manual design and verification of circuits become impractical. Verilog offers a systematic approach, enabling designers to model intricate systems, simulate their behavior, and subsequently synthesize them into hardware. This iterative process of design, simulation, and synthesis ensures that the final hardware meets the desired specifications and functions correctly. For students at the Amirkabir University of Technology, mastering Verilog is not just an academic exercise but a crucial skill that will empower them to contribute effectively to the ever-evolving field of digital design.

Moving from the broader landscape of digital design to the foundational building blocks, the full adder emerges as a quintessential component. At its core, the full adder is a digital circuit that performs the arithmetic sum of three input bits. While this might seem rudimentary, the full adder is a cornerstone in the construction of more complex arithmetic units, such as multi-bit adders, subtractors, and even multipliers. Understanding the intricacies of a full adder, therefore, provides insight into the inner workings of these larger systems.

The full adder's significance is further magnified when we consider its role in various applications. From basic calculators to advanced microprocessors, the full adder's ability to handle binary arithmetic is indispensable. Its design principles, involving the management of carry propagation and the handling of multiple input combinations, introduce students to challenges they will frequently encounter in more advanced digital circuits.

In this exercise, we will meld the power of Verilog with the foundational importance of the full adder. By designing a parameterized full adder/subtractor module, students will not only deepen their understanding of arithmetic circuits but also hone their skills in Verilog, preparing them for more advanced projects and real-world applications.

Table of Content

Exercise Objective

Design a parameterized full adder/subtractor module in Verilog. The module should be able to handle operands of varying bit-widths based on a parameter and perform either addition or subtraction based on a control signal.

Specifications

Inputs

- **A**: First operand.
  - Width: Determined by the parameter `WIDTH`.
- **B**: Second operand.
  - Width: Determined by the parameter `WIDTH`.
- **C_in**: Carry-in.
  - Width: Single bit.
- **Sub**: Control signal.
  - Width: Single bit.
  - Functionality:
    - When `Sub` is 1, the module should subtract B from A.
    - When `Sub` is 0, the module should add A and B.

Outputs

- **Sum**: Represents the sum (or difference) of A and B.
  - Width: Determined by the parameter `WIDTH`.
- **C_out**: Carry-out.
  - Width: Single bit.
- **V**: Represents the OverFlown state in the module
  - Width: Single bit
- **Z**: This bit shows if the sum is zero or not
  - Width: Single bit

Parameters

WIDTH: Determines the bit-width of the operands (A and B) and the sum.

Requirements

- Design: Implement the full adder/subtractor using basic logic gates or using arithmetic operations in Verilog.
- Modularity: Your design should be modular. Consider using a basic full adder module and instantiating it multiple times.
- Testbench: Write a testbench to verify the functionality of your design. The testbench should test various bit-widths and both addition and subtraction operations.
- Simulation: Simulate your design and testbench. Ensure that the design works correctly for all possible input combinations.
- Documentation: Comment your code thoroughly. Explain the functionality of each module and how they interact.

Hints

- Remember that subtraction can be performed using addition. Consider how the two's complement can be used for this purpose.
- For subtraction, invert the bits of operand B and add 1 (two's complement) when the `Sub` signal is active.
- The carry-out after the most significant bit can be used as an overflow indicator for addition and underflow indicator for subtraction.

Submission

- Verilog Code: Submit your parameterized full adder/subtractor module and any other supporting modules.
- Testbench Code: Submit your testbench Verilog code.
- Simulation Results: Provide waveforms or screenshots showing the simulation results for various test cases.
- Report: Write a brief report explaining your design approach, the challenges you faced, and how you overcame them. Include observations from your simulation results.

Evaluation Criteria

- Functionality: Does the module correctly perform addition and subtraction based on the control signal?
- Parameterization: Does the module correctly handle different operand bit-widths based on the `WIDTH` parameter?
- Code Quality: Is the code well-organized, modular, and thoroughly commented?
- Testbench Quality: Does the testbench thoroughly test all functionalities and edge cases?
- Documentation: Is the report well-written, and does it provide a clear understanding of the design and results?

Best Regards, Hamidi