Ande Chen U93195418

Modules

NBitNOT_str

This module is a parameterized bitwise NOT for a N-bit input. It produces a N-bit output by performing the logical NOT by flipping each bit of the input.

FA_str

This module is a 1-bit adder that takes two 1-bit inputs and a 1-bit carry in as inputs and produces a 1-bit output and a 1-bit carry out.

NBitAdder_str

This module is a parameterized adder that takes two N-bit inputs and a 1-bit carry in and produces a N-bit sum and a1-bit carry out by generating N 1-bit full adders connected as a ripple carry adder.

NBitSubtractor_str

This module is a parameterized subtractor that takes two N-bit inputs and a 1-bit carry in and produces a N-bit difference and a 1-bit carry out. It calculates the difference by taking the 2's complement of the second N-bit inputs by flipping the bits and adding 1 and summing that with the first N-bit input.

NBitOR_str

This module is a parameterized bitwise OR for two N-bit inputs. It produces a N-bit output by performing the logical OR on each ith pair of bits of the inputs.

NBitAND_str

This module is a parameterized bitwise AND for two N-bit inputs. It produces a N-bit output by performing the logical AND on each ith pair of bits of the inputs.

mux_str

This module is a parameterized 3-8 multiplexer that takes the N-bit outputs of the 7 operation modules (MOV, NOT, ADD, SUB, OR, AND, and STL) and the 3-bit ALUOps code as inputs, selects the output of the operation that corresponds to the ALUOps code, and returns that output.

NBitRegister_str

This module is a parameterized N-bit register that takes a N-bit input D and a clock signal as its inputs and produces a N-bit output Q. It synchronizes the updating of the register with the positive edge of the clock signal.

dff

Ande Chen U93195418

This module is a 1-bit register that takes a 1-bit input D and a clock signal as its inputs and produces a 1-bit output Q. It synchronizes the updating of the register with the positive edge of the clock signal.

ALU_str

This module is a parameterized Arithmetic Logic Unit that takes 2 N-bit inputs, a 1-bit carry in, a 3-bit operation code, and a clock signal as inputs and produces a N-bit output and a 1-but carry out. It can manipulate the inputs by

- passing through (MOV) the first N-bit input
    - This functionality is implemented directly in the ALU module by simply passing the first N-bit input to the mux as the N-bit wire MOV
- performing the bitwise NOT of the first N-bit input
- adding the 2 N-bit inputs
- subtracting the 2 N-bit inputs
- performing the bitwise OR of the 2 N-bit inputs
- performing the bitwise AND of the 2 N-bit inputs
- STL

It instantiates each of the operation modules and computes their N-bit outputs in parallel. Then, it passes those outputs to 2 instances of the mux module to select the correct output given the ALUOps code and the correct carry out, if applicable. The outputs of the mux are then passed to a N-bit register for the overall output of the ALU and a 1-bit register for the final carry out.

ALU_behavioral

This module is the behavioral Verilog representation of the ALU's functionalities.

ALU_str_tb

This module is a testbench that verifies the functionality of the structural Verilog ALU by instantiating it along with the behavioral ALU. Then, it runs through a representative set of test cases and raises an error flag whenever the final output of the structural ALU does not match the final output of the behavioral ALU.

Ande Chen U93195418

Design Hierarchy

- ALU_str_tb
    - ALU_str
        - NBitNOT_str
        - NBitAdder_str
            - FA_str (generated N times)
        - NBitSubtractor_str
            - NBitNOT_str
            - NBitAdder_str
                - FA_str (generated N times)
        - NBitOR_str
        - NBitAND_str
        - mux_str
        - NBitRegister_str
            - dff (generated N times)
    - ALU_behavioral

Ande Chen U93195418

Simulation Results & Waveforms

ALUOp	Function
000 = 0	MOV
001 = 1	NOT
010 = 2	ADD
011 = 3	SUB
100 = 4	OR
101 = 5	AND
110 = 6	SLT

Initial Error is from not declaring an initial ALUOp code