CSE 331 - Spring 2021

Homework 3 Report

Okan Torun

1801042662

Explaining All Verilog Files:

_32_bit_and: This module takes two 32-bit inputs into the and operation and saves the result to the 32-bit register.

_32_bit_or: This module takes two 32-bit inputs into the or operation and saves the result to the 32-bit register.

_32_bit_nor: This module takes two 32-bit inputs into the nor operation and saves the result to the 32-bit register.

_32_bit_xor: This module takes two 32-bit inputs into the xor operation and saves the result to the 32-bit register.

full_adder: This module gets two 1-bit input and gives the answer of add operation of two input

_4bit_adder: This module takes two 4-bit inputs and saves their add operation to the 4-bit register. This module calls the full_adder module 4 times.

_32bit_adder: This module takes two 32-bit inputs and saves their add operation to the 32-bit register. This module calls the 4_bit_adder module 8 times.

_32bit_sub: This module takes two 32-bit inputs and saves their sub operation to the 32-bit register. For the sub operation, I put my B input into xor operation with a 32-bit 1-value register. Then I added the result with 1 and added the final result with the value A. This gave us the sub result.

_32bit_slt: This module takes two 32-bit inputs and saves their sub operation to the 32-bit register. In this module I did the same operations as I did in the sub module. And then I put the 32nd bit in the or operation with 0. Thus, I saved the value 1 if the result is negative, 0 if it is positive. I set the other bits to 0.

_mux8x1: This module takes 8 32-bit inputs and 3-bit opcodes. I continued by sending these values to a 4x1 mux with opcode values and the results to a 2x1 mux.

_mux4x1: This module takes 4 32-bit inputs and 2-bit opcodes. I process these inputs by calling 2x1 mux module operations. And I save the output values to 32-bit wires. Finally, I insert 2 32-bit wires into 2x1 mux and save the result to 32-bit output.

_mux2x1: This module takes 2 32-bit inputs and 1-bit opcodes. It processes the incoming inputs according to the 2x1 mux formula and saves the result to the F output.

alu: This module performs all the desired operations and saves them to different wires. Then it sends all wires and opcodes to an 8x1 mux and saves the result to the R output.

Test Cases:

```
VSIM 6> step -out -current
# time =  0, a =00000000000000000000000000001101, b=00000000000000000000000000001001 , S=000, result=00000000000000000000000000010110
# time = 20, a =00000000000000000000000000001001, b=00000000000000000000000000001000 , S=001, result=00000000000000000000000000000001
# time = 40, a =00000000000000000000000000001101, b=00000000000000000000000000001001 , S=010, result=00000000000000000000000000000100
# time = 60, a =00000000000000000000000000001111, b=00000000000000000000000000000101 , S=011, result=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# time = 80, a =00000000000000000000000000001101, b=00000000000000000000000000000101 , S=100, result=00000000000000000000000000000000
# time = 100, a =00000000000000000000000000000011, b=00000000000000000000000000000011 , S=101, result=11111111111111111111111111111100
# time = 120, a =00000000000000000000000000001000, b=00000000000000000000000000001101 , S=110, result=00000000000000000000000000001000
# time = 140, a =00000000000000000000000000001001, b=00000000000000000000000000000110 , S=111, result=00000000000000000000000000001111
```

```
VSIM 5> step -out -current
# time =  0, a =00000000000000000000000000001101, b=00000000000000000000000000001001 , S=000, result=00000000000000000000000000010110
# time = 20, a =00000000000000000000000000001011, b=00000000000000000000000000001110 , S=001, result=00000000000000000000000000000101
# time = 40, a =00000000000000000000000000001111, b=00000000000000000000000000001010 , S=010, result=00000000000000000000000000000101
# time = 60, a =00000000000000000000000000000001, b=00000000000000000000000000001101 , S=011, result=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
# time = 80, a =00000000000000000000000000001011, b=00000000000000000000000000000110 , S=100, result=00000000000000000000000000000000
# time = 100, a =00000000000000000000000000000011, b=00000000000000000000000000000011 , S=101, result=11111111111111111111111111111100
# time = 120, a =00000000000000000000000000000100, b=00000000000000000000000000001111 , S=110, result=00000000000000000000000000000100
# time = 140, a =00000000000000000000000000001001, b=00000000000000000000000000000100 , S=111, result=00000000000000000000000000001101
```