CSCI 330 Final Project

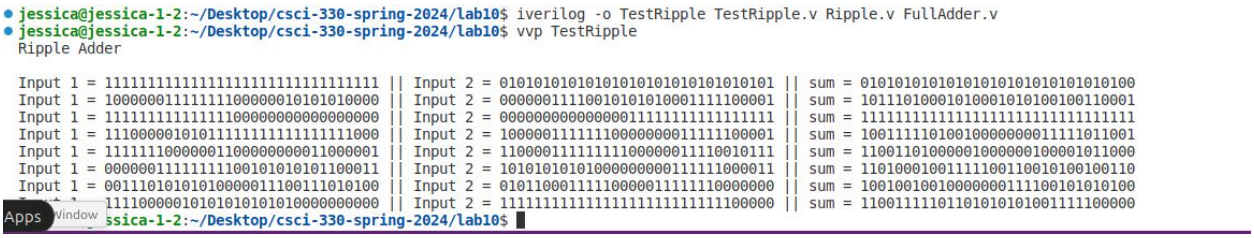
Component Write-Up Document

Jessica Taylor

**Note: I was unable, unfortunately, to record a video demonstrating the following components due to my VM continually crashing while running my screen recording software **

Ripple Carry Adder

1. The full adder takes two inputs (A, B) and produces one output (Sum) which is the 32-bit sum of A and B. The ripple carry adder is comprised of full adders.
2. The Ripple.v is the main file. This file contains two 32-bit inputs, one 32-bit output which is the sum of the two 32-bit inputs, 31 wires, and one Cout wire.
3. The main file contains numerous lines of full adder instances. These instances handle one bit of the input operands, along with the carry from the previous bit's addition.
4. The TestRipple.v is the test file. This file contains two 32-bit inputs, one 32-bit sum, one Cout wire, and one integer.
5. The test file contains eight test cases with various combinations to assess code accuracy. Each test case contains a for loop, which iterates over each bit of the input and output registers from the most significant bit to the least significant bit.
6. Command to compile and run:
    a. iverilog -o TestRipple TestRipple.v Ripple.v FullAdder.v
    b. VVP TestRipple
7. Below is a screenshot of the output.



32-Bit Register File

1. The 32-bit register file is named Register.v
2. The 32-bit register file is used in the single cycle processor to temporarily store data during execution
3. This file contains a clk, which is a clock signal used to update the register file
4. This file contains a reset which is a signal to reset the content of the register file and output registers
5. ReadReg1, ReadReg2 are 5-bit inputs specifying which registers' contents are to be read.
6. WriteReg is a 5-bit input specifying which register is to be written to
7. WriteData is the 32-bit data to be written to the specified register
8. RegWrite is a control signal that enables writing to the register when high
9. ReadData1, ReadData2 are outputs where the contents of the registers specified by ReadReg1 and ReadReg2 are placed
10. The 'always' block handle reading data from the register file. If the reset signal is asserted, the output registers 'ReadData1' and 'ReadData2' are updated. If 'reset' is high, the outputs are cleared. Otherwise, the contents of the registers are copied to 'ReadData1' and 'ReadData2'

Instruction Memory Module

1. The instruction memory module file is named Memory.v
2. The file contains: clk(clock), rst(reset), addr(address), instr(instructions)
3. In the file, clk is a clock signal that determines the timing of reading data from memory
4. In the file, rst is a reset signal that is used to reset the contents of the output register (instr)
5. In the file, addr is a 32-bit input specifying the memory address from which to fetch the instruction
6. In the file, instr is a 32-bit output register that holds the instruction fetched from memory
7. The 'initial' block initializes the first two locations of the memory with specific values that represent example machine code instructions in hexadecimal format
8. The 'always' block triggers on the 'rst' signal. If it is high, the 'instr' will reset to zero. If 'rst' is not asserted the 'addr' input is loaded into the 'instr' register

Data Memory Module

1. The data memory module file is named DataMemory.v
2. This component is used in the single cycle processor to store and retrieve data
3. In this file, input wires Ina and Inb are 32-bit input ports representing data inputs for read and write operations
4. In this file, input wire enable is an input signal indicating whether the memory should perform any operations
5. In this file, input wire readwrite is an input signal indicating whether to perform a read operation or a write operation
6. In this file, input wire clk is the clock input used for triggering operations
7. In this file, input wire rst is the reset input to reset the 'dataOut' register
8. In this file, dataOut is the 32-bit output register representing the data output from the memory
9. The 'always' block is triggered on the 'rst'
10. If 'rst' is asserted, the 'dataOut' register is cleared
11. If 'enable' is high, the memory performs an operation based on 'readwrite' signal
12. If 'readwrite' is high, the memory performs a read operation and the data at address 'Ina' is output to 'dataOut'
13. If 'readwrite' is low, the memory performs a write operation and the data at address 'Ina' is written to the memory at address 'Inb'

Control ROM

1. The Control ROM file is named Decoder.v and the test file is DecoderTest.v
2. This component is used in the single cycle processor to generate control signals for various operations based on the current instruction being executed.
3. The following are the commands to compile and run the files:
    a. Iverilog -o DecoderTest DecoderTest.v Decoder.v
    b. vvp Decoder Test
4. Below is the screenshots of the test output (May need to enlarge):