Project 1: Environment Setup and Simple ALU

Objectives:

1. Build your Verilog environment
2. Explore a simple FSM
3. Build a simple ALU

Part A): Sample FSM

Using the SampleFSM Verilog files, build and successfully run the FSM sample using the Getting Started and Lab 1 documents as a guide. Make sure you understand what’s going on in the Verilog by reviewing the syntax AND reviewing waveform outputs. Learn how to scale and show results on Waveform viewers that are meaningful.

Repeat the exercise using the FIFO Verilog files.

Results:

1. Show results of SampleFSM and Testbench on Waveform Viewer.
   1. Show all inputs and outputs
   2. Show State
   3. Place one blank row between each waveform for easier viewing
   4. Show all results until steady state
2. Build a FSM to count from 0 to 7 three times.
   1. Have a reset signal and use three bits for the count. If reset is ever active (high), counter goes back to zero on next clock. If reset is low, the FSM counts based on the rules.
   2. The counter will count from 0 to 7 only three times. It will hold the value 7 on the third count’s completion. In other words, if reset stays low long enough, the counter will count 0 to 7, 0 to 7, and 0 to 7 and hold 7 until reset goes high.
   3. Show your Verilog working on a waveform viewer (gtkwave).

PART B): Simple ALU

Write Verilog and a testbench to perform ALU operations on two 16 bit numbers.

Inputs:

Numbers on which to operate: Data\_A (16 bits) and Data\_B (16 bits)

Opcode for operation: Opcode

Clock: CLK

Reset: RST

Enable: If high ALU operates, if low ALU will not change values.

Outputs:

Result: Results (16 bit value)

Overflow: CF (1 bit)

Opcode table: Note, no change means that the results output does not change value. A = Results means that the register A will store the results and overwrite old A.

|  |  |  |
| --- | --- | --- |
| Opcode | Operation | Result based on Enable |
| 000 | NOP | No Change |
| 001 | A+B | Results |
| 010 | A&B | Results |
| 011 | A|B | Results |
| 100 | A<B | A<B ? 1:0 |
| 101 | A+B | A = Results (A’s value changes) |
| 110 | A&B | A = Results |
| 111 | A|B | A = Results |

Operation:

The ALU is a static design (clocked).

When Resetting the ALU all internal registers should go to known state and zeros. Output should be zeros.

The Enable input will act as an enable to tell the ALU to provide the results based on the operation in the Opcode.

The overflow (CF) will equal a “1” if the result of your Result is greater than 16 bits. We will not worry about underflow.

Provide a test bench with your Verilog to test your ALU. Use the names given for input and outputs.

An example test flow is shown below (numbers in Hex). This is a clocked ALU so all changes are on clock edges.

All inputs zero

1. Reset (1 clk)
2. Enable=0

A=6464, B=4646

Opcode=001

1. Enable=1
2. Enable=0

Opcode=002

1. Enable=1
2. Opcode=011
3. Opcode=000
4. Opcode=100
5. Opcode=101
6. Opcode=101
7. A=6464, Opcode=110
8. Opcode=111
9. Done

PART C) Replace your adder function (+) with a Carry lookahead algorithm. Refer to Example A.6.3 in text. It uses 4-bit adders to create a 16-bit carry lookahead. You may also reference the pdf or other online examples.

Implement a 4-bit subblock. Use the 4-bit block to build your 16-bit carry lookahead. Your Verilog must call (use) the 4-bit subblock to build you carry lookahead adder.

Test your design with your testbench from Part B to show it still works.

Test Checks

Part A): Sample FSM

Using the SampleFSM Verilog files, build and successfully run the FSM sample using the Getting Started and Lab 1 documents as a guide. Make sure you understand what’s going on in the Verilog by reviewing the syntax AND reviewing waveform outputs. Learn how to scale and show results on Waveform viewers that are meaningful.

Repeat the exercise using the FIFO Verilog files.

Results:

1. Show results of SampleFSM and Testbench on Waveform Viewer.
   1. Show all inputs and outputs
   2. Show State
   3. Place one blank row between each waveform for easier viewing
   4. Show all results until steady state
2. Build a FSM to count from 0 to 7 three times.
   1. Have a reset signal and use three bits for the count t.
   2. The counter will count from 0 to 7 only three times. It will hold the value 7 on the third count’s completion
   3. Show your Verilog working on a waveform viewer (gtkwave).

Part B and Part C

All inputs zero

1. Reset (1 clk)

All Registers and outputs go to a known state (no “x”).

1. Enable=0

A=6464, B=4646

Opcode=001

Enable = 0, so no change to results, CF. Reg A = 6464

1. Enable=1

Results = AAAA, CF = 0, Reg A = 6464.

1. Enable=0

Opcode=002 (Should be 010)

Enable =0. So no changes to output, CF. Reg A = 6464

1. Enable=1

A&B. Results = 4444, CF = 0, Reg A = 6464

1. Opcode=011

A|B. Results = 6666, CF = 0. Reg A = 6464

1. Opcode=000

NOP. Results = 6666, CF = 0, Reg A = 6464

1. Opcode=100

A<B. A is greater than B, so Results = 0, CF = 0, Reg A = 6464

1. Opcode=101

A+B results in A. Results = AAAA, CF = 0, Reg A = AAAA

1. Opcode=101

A+B results in A. Results = F0F0, CF = 0, Reg A = F0F0

1. Opcode=101

A+B results in A. Results = 3736. CF = 1. Reg A = 3736

1. A=6464, Opcode=110

A&B results in A. Results = 4444. CF = 0. Reg A = 4444

1. Opcode=111

A|B results in A. Results = 4646. CF = 0. Reg A = 4646

1. Done