Digital Logic Final Project

Cohort: Lone Wolf

(Sudarshana Jagadeeshi)

Project: ALU

List of Parts

Inputs

- Clock: 1 bit. Feeds into the A and B registers, as well as the storedvalue register and the error register.
- Reset: 1 bit. Feeds into storedvalue and errorbit registers.
- A_userinput: 32 bits, unsigned integer.
- B_userinput: 32 bits, unsigned integer.
- Opcode: 4 bits. Used to control the SelectorBigMux.

Outputs

- Error: 1 bit, high when overflow or a negative number has occurred.
- C: the current value of the register storedvalue.

Interfaces

- Splitter: Concats the 4 bits of the opcode to feed into the SelectorBigMux.
- Adder.A: 32 bit, the value of register A
- Adder.B: 32 bit, the value of register B
- Adder.cin: Grounded to 0
- Adder.cout: 1 bit, the leading digit of the 33 digit sum, used in error check.
- Subtractor.A: 32 bit, the value of register A
- Subtractor.B: 32 bit, the value of register B
- Subtractor.bin: Grounded to 0
- Subtractor.bout: 1 bit, high if the result of A-B is negative. Used in the error check.
- A.D: 32 bits, the current user_input value for A
- A.Q: 32 bits, the next value of the register.
- A.0: 1 bit, resets the contents of the register when high.
- A.en: 1 bit, left unused. Would theoretically enable the register.
- B.D: 32 bits, the current user_input value for B
- B.Q: 32 bits, the next value of B.
- B.0: 1 bit, resets the contents of the register when high.
- B.en: 1 bit, left unused. Would theoretically enable the register.

- storedvalue.D: 32 bits, the selected operations value and the value to be stored.
- storedvalue.Q: 32 bit, the next value of the register.
- storedvalue.0: 1 bit, resets the contents of the register when high.
- storedvalue.en: 1 bit, left unused. Would theoretically enable the register.
- errorbit.D: 1 bit, the current error in
- errorbit.Q: 1 bit, the updated value of the error and the next to be stored
- errorbit.0: 1 bit, resets the contents of the register when high.
- errorbit.en: 1 bit, left unused. Would theoretically enable the register.
- SelectorBigMux.Ch0: 32 bits, all zeros.
- SelectorBigMux.Ch10: 32 bits, the result of the addition A+B
- SelectorBigMux.Ch2: 32 bits, the result of the subtraction A-B
- SelectorBigMux.Ch3: 32 bits, the result of and'ing A and B
- SelectorBigMux.Ch4: 32 bits, the result of or'ing A and B
- SelectorBigMux.Ch5: 32 bits, the result of xor'ing A and B
- SelectorBigMux.Ch6: 32 bits, the result of not'ing A
- SelectorBigMux.Ch7: 32 bits, the result of nand'ing A and B
- SelectorBigMux.Ch8: 32 bits, the result of nor'ing A and B
- SelectorBigMux.Ch9: 32 bits, the result of xnor'ing A and B
- SelectorBigMux.Ch10: 32 bits, the current value of C
- SelectorBigMux.Ch11:16 are 32 bit channels each. They are left unused.
- SelectorBigMux.S: 4 bits, the opcode to select an operation.
- SelectorTinyMux1.Ch0: 32 bits, the result of the addition A+B
- SelectorTinyMux2.Ch0: 32 bits, the result of the subtraction A-B
- SelectorTinyMux3.Ch0: 32 bits, the result of and'ing A and B
- SelectorTinyMux4.Ch0: 32 bits, the result of or'ing A and B

- SelectorTinyMux5.Ch0: 32 bits, the result of xor’ing A and B
- SelectorTinyMux6.Ch0: 32 bits, the result of not’ing A
- SelectorTinyMux7.Ch0: 32 bits, the result of nand’ing A and B
- SelectorTinyMux8.Ch0: 32 bits, the result of nor’ing A and B
- SelectorTinyMux9.Ch0: 32 bits, the result of xnor’ing A and B
- SelectorTinyMux1-9.Ch1: 32 bits, the current value of C in the register.
- SelectorTinyMux1-9.S, 1 bit, the error bit.

Gates

- AND Gate: 2 channels of 32 bits each. The top channel feeds from register A, the other from register B.
- OR Gate: 2 channels of 32 bits each. The top channel feeds from register A, the other from register B.
- XOR Gate: 2 channels of 32 bits each. The top channel feeds from register A, the other from register B.
- AND Gate: 2 channels of 32 bits each. The top channel feeds from register A, the other from register B.
- NOT Gate #1: 1 channels of 32 bits. The top channel feeds from register A.
- NAND Gate: 2 channels of 32 bits each. The top channel feeds from register A, the other from register B.
- NOR Gate: 2 channels of 32 bits each. The top channel feeds from register A, the other from register B.
- XNOR Gate: 2 channels of 32 bits each. The top channel feeds from register A, the other from register B.
- AND Gate #2: 5 channels of 1 bit each. Used to test if addition overflow has occurred.
- AND Gate #3: 5 channels of 1 bit each. Used to test if subtraction negative number has occurred.
- OR Gate #2: 2 channels of 1 bit each. Used to test if either of the errors have returned true.
- NOT Gate #2: Inverts highest bit of opcode (opcode[3]), feeds into AND gate #2
- NOT Gate #3: Inverts opcode[2], feeds into AND gate #2
- NOT Gate #4: Inverts opcode[1], feeds into AND gate #2

- NOT Gate #5: Inverts highest bit of opcode (opcode[3]), feeds into AND gate #3
- NOT Gate #6: Inverts opcode[2], feeds into AND gate #3
- NOT Gate #7: Inverts lowest bit of opcode (opcode[0]), feeds into AND gate #3

Combinational Logic Components

- Adder: 32-bit, with c_in always 0
- Subtractor: 32-bit, with c_in always 0
- SelectorBigMux: 16 channels of 32 bits each. Used to select which value to store in C.
- SelectorTinyMux1: 2 channels of 32 bits each. Used to select whether to send the addition result to the SelectorBigMux (as opposed to the current value)
- SelectorTinyMux2: Used to select whether to send the Subtraction result to the SelectorBigMux
- SelectorTinyMux3: Used to select whether to send the AND result to the SelectorBigMux.
- SelectorTinyMux4: Used to select whether to send the OR result to the SelectorBigMux
- SelectorTinyMux5: Used to select whether to send the XOR result to the SelectorBigMux.
- SelectorTinyMux6: Used to select whether to send the NOT result to the SelectorBigMux.
- SelectorTinyMux7: Used to select whether to send the NAND result to the SelectorBigMux.
- SelectorTinyMux8: Used to select whether to send the NOR result to the SelectorBigMux.
- SelectorTinyMux9: Used to select whether to send the XNOR result to the SelectorBigMux.

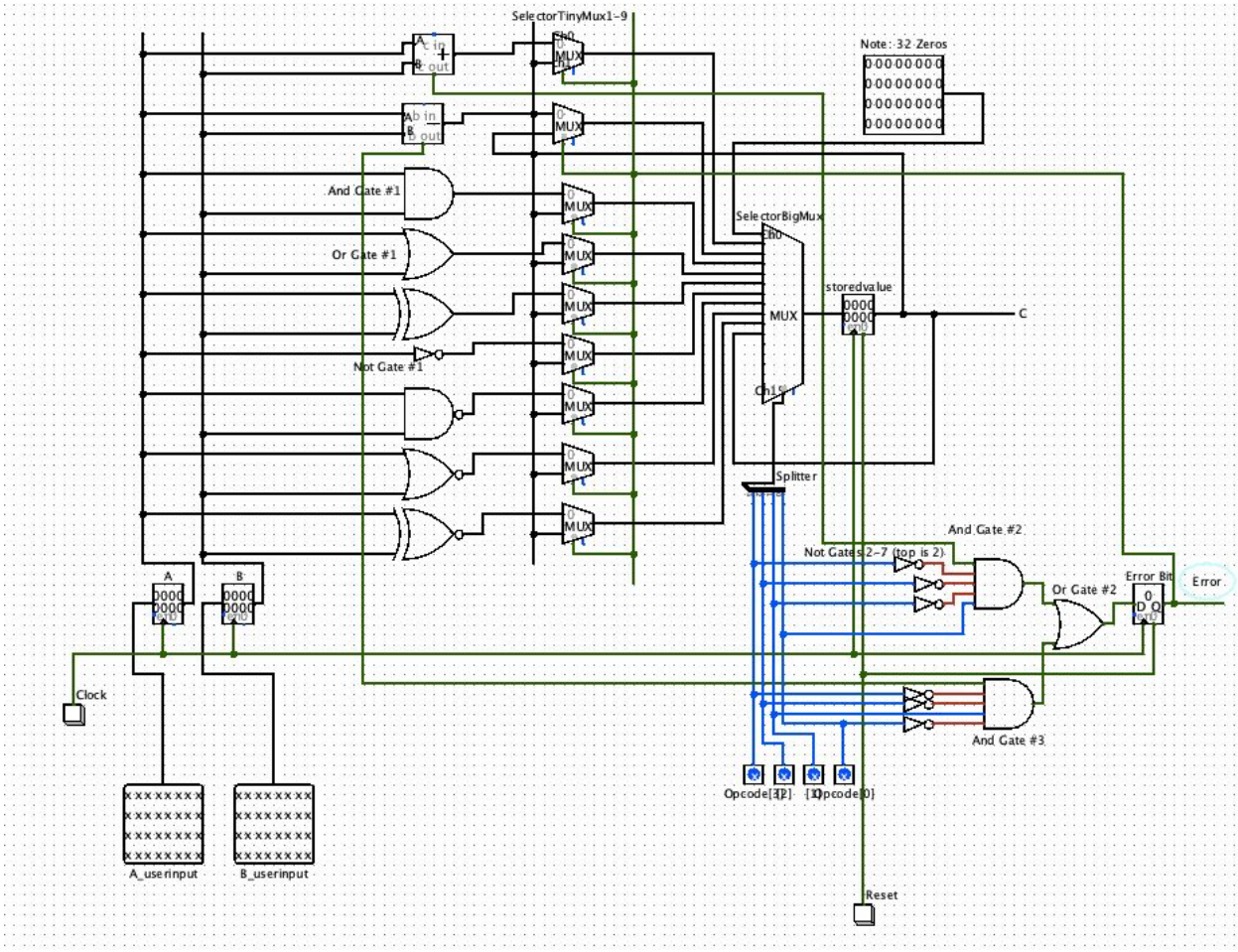
Sequential Logic Components

- storedvalue: 32 bit register that contains the current stored value.
- errorbit: 1 bit register that contains the current error.
- A: 32 bit register that contains the current value of A
- B: 32 bit register that contains the current value of B

Modules

- Test Bench- A module that initializes the square wave for the clock and issues commands via stimulus (not shown).
- Selector- A module that determines how storedvalue is updated based on the results of the Tiny and BigMux.
- ALU- Where all the modules and pieces, including Selector, are assembled. Shown in entirety below.

Circuit Diagram



Notes:

- I didn't add text boxes for every channel of each TinyMux. The channels should be the same as the top one. Similarly, the big mux channels are in ascending order, I have labeled the top and bottom ones.
- My design decision was that you cannot reset the A or B registers. Reset only clears the stored value and error, and you can then assign/overwrite new values to A and B.
- The TinyMux's are not required, but I thought I'd add them in.

List of Opcodes and States

Command	Binary Label	Description
Reset	0000	Clear the accumulator and the error bit
Add	0001	Add the input to the value stored in the accumulator
Sub	0010	Subtract the input from the value stored in the accumulator
And	0011	Perform and with the input value and the stored value
Or	0100	Perform or with the input value and the stored value
Xor	0101	Perform xor with the input value and the stored value
Not	0110	Perform not with the input value and the stored value
Nand	0111	Perform nand with the input value and the stored value
Nor	1000	Perform nor with the input value and the stored value
Xnor	1001	Perform xnor with the input value and the stored value
No-op	1010	Simply pause and do nothing for that clock cycle

Modes of Operation

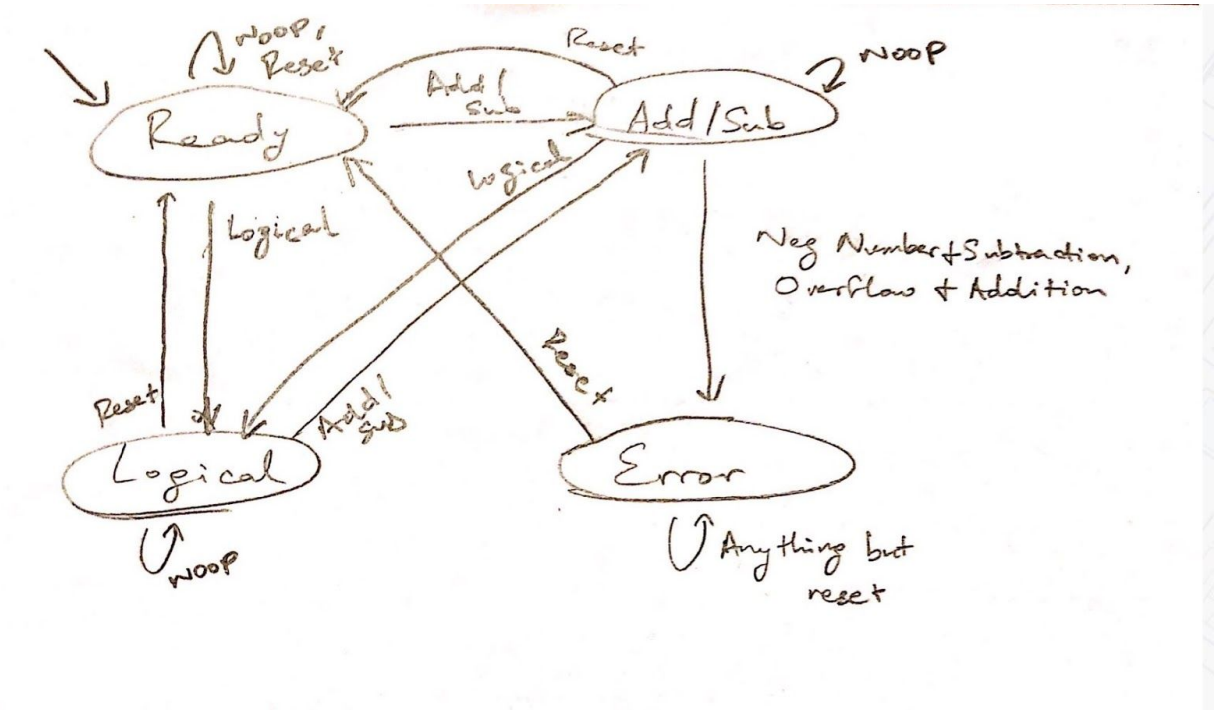
Mode	Code	Operation
Ready	000	System is waiting for both an A and B input, as well as an

		operation.
Add/Sub	001	Performing an add/sub operation
Logical	010	Performing any one of the 6 logical operations described in the above table
Error	011	Continues to NOOP in the error state until a reset is supplied.

State Table

Current State	Command	Next State
Ready	NOOP, RESET	Ready
Ready	ADD, SUB	Add/Sub
Ready	AND, OR, XOR, NOT, NAND, NOR, XNOR	Logical
Logical	NOOP	Logical
Logical	Reset	Ready
Logical	Add/Sub	Add/Sub
Logical	Reset	Ready
Add/Sub	NOOP	Add/Sub
Add/Sub	AND, OR, XOR, NOT, NAND, NOR, XNOR	Logical
Add/Sub	Overflow/Neg Number	Error
Add/Sub	Reset	Ready
Error	Anything but Reset	Error
Error	Reset	Ready

State Machine



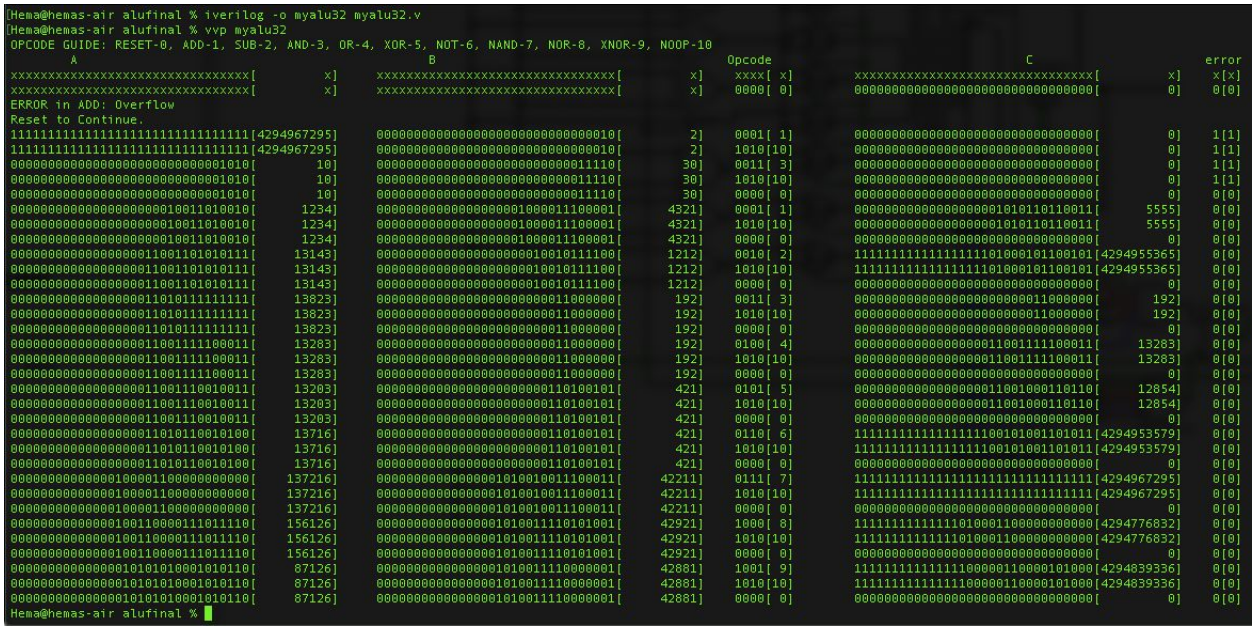
Notes:

- To be explicit, I have denoted the mathematical state as Add/Sub, because those are our only math operations.
- Our ALU actually does allow free travel between the logical and mathematical states.
- The machine will continue to no-op when in the error state, a reset must be supplied to return to the ready state.

Verilog Code

Refer to attached file- myalu32.v.

Verilog Output



Notes:

- The error message appears the line before the offending operation, instead of after.