Introduction to Processor Architecture

Assignment - 1

Ananya Sane (2020102007)

L Lakshmanan (2020112024)

Team wireWire

February 1, 2022

Contents

64-bit ALU in Verilog HDL

Inputs

Operands, Opcode.

Outputs

Result, Overflow flag, Zero flag.

Functions

The ALU has 4 functions that are binary or unary, and that can be used on 64-bit binary numbers. These 4 functions are

- Add $\rightarrow$ 00
- Subtract $\rightarrow$ 01
- AND $\rightarrow$ 10
- XOR $\rightarrow$ 11

Each of these functions have their own two bit opcodes, which have been specified beside them. These opcodes are what the ALU uses to determine which operation needs to be computed.

Two flags have been initialised in the ALU, one for the overflow and one for indicating if the output is a zero. The overflow flag is 1 when there is an overflow in the performed operation and is zero in all other cases, while the zero flag is 1 when the output is zero. Results of the operations are stored in the `res` register.

Add function

Takes two 64-bit inputs. Returns the sum as a result, along with the appropriate flags. The 64-bit adder is constructed using loops and full adder modules, which have been instantiated using the `generate` and the for loop structures.

Subtract function

Takes two 64-bit inputs. Returns the difference as a result, along with the appropriate flags. The subtraction module has been constructed using the 64-bit adder as a module, and by complementing the required number using `generate` block, for loops and the `not` primitive.

AND function

Takes two 64-bit inputs. Returns the logical AND value of the two as a result, along with the appropriate flags. Constructed using just the gate primitive, the `generate` block and for loop structure.

XOR function

Takes two 8 bit inputs. Returns the logical XOR value of the two as a result, along with the appropriate flags. Constructed using just the gate primitive and the `generate` block and for loop structure.

Testbenches

Each function's testbench seeds inputs with random numbers and passes them to the ALU, with the final result being stored in the `res` register. For each set of inputs, the value `res - (a\ OP\ b)` is printed. There are two cases: - In most cases, the output is 0; this means the ALU works and both results match. - In the cases where the expression is non zero, the overflow bit is set; this makes sense as `res` stores a value that is not the correct answer, and hence it is not equal to the value of `a\ OP\ b`.

The outputs of the testbenches are written to the .txt files attached.

Results for Add function

```
VCD info: dumpfile ALU_64.vcd opened for output.
a = 0000000000000000000000000000000000000000000000000000000000001011
b = 0000000000000000000000000000000000000000000000000000000000101010
res = 0000000000000000000000000000000000000000000000000000000000110101
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000010010000101010011010100100100
b = 1111111111111111111111111111111111000000100010010101111010000001
res = 1111111111111111111111111111111111010010100111101001001110100101
overflow = 0
zero = 0
```

```
Check:                    0

a = 1111111111111111111111111111111110000100100001001101011000001001
b = 11111111111111111111111111111111110110001111100000101011001100011
res = 1111111111111111111111111111111100110110011101010010110001101100
overflow = 0
zero = 0
Check:                    0

a = 00000000000000000000000000000000000110101110010111101100001101
b = 000000000000000000000000000000001000110110111111001100110001101
res = 000000000000000000000000000000001001101100110010001010010011010
overflow = 0
zero = 0
Check:                    0

a = 11111111111111111111111111111111110110010110000101000010001100101
b = 1111111111111111111111111111111110001001001101110101001000010010
res = 1111111111111111111111111111111100111011111110011101011001110111
overflow = 0
zero = 0
Check:                    0

a = 00000000000000000000000000000000000000111100111110001100000001
b = 000000000000000000000000000000000000110110101111100110100001101
res = 00000000000000000000000000000000000011111001011101100000001110
overflow = 0
zero = 0
Check:                    0

a = 000000000000000000000000000000000111011001000111111000101110110
b = 00000000000000000000000000000000011110100011011100110100111101
res = 0000000000000000000000000000000001011001101100011011111010110011
overflow = 0
zero = 0
Check:                    0

a = 000000000000000000000000000000001110110110101000101011111101101
b = 000000000000000000000000000000001000110001011011111011110001100
res = 0000000000000000000000000000000010111101000000100100111101111001
overflow = 0
zero = 0
Check:                    0

a = 000000000000000000000000000000001111100111111011110100111111001
b = 1111111111111111111111111111111111100011001101110010010011000110
```

```
res = 0000000000000000000000000000000001100000001101010000111010111111
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111111100010111101111000010011000101
b = 1111111111111111111111111111111111010101000100111101001010101010
res = 1111111111111111111111111111111110111000000010110101011101101111
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001110010101011111111011111100101
b = 1111111111111111111111111111111110111011110100100111001001110111
res = 0000000000000000000000000000000000101110100000100110101001011100
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111110001001001100101101011000010010
b = 0000000000000000000000000000000001000111111011001101101110001111
res = 1111111111111111111111111111111111010001000111111011000110100001
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001111001001100000110100111110010
b = 1111111111111111111111111111111111100111011101101001011011001110
res = 0000000000000000000000000000000001100000101001110000000011000000
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111111110100000000000111101011101000
b = 1111111111111111111111111111111111100010110010100100111011000101
res = 1111111111111111111111111111111111010110110010101100100110101101
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000101110010110000100100101011100
b = 1111111111111111111111111111111111011110100011100010100010111101
res = 0000000000000000000000000000000000001100111001100111001000011001
overflow = 0
zero = 0
Check:                    0
```

Results for Sub function

```
VCD info: dumpfile ALU_64.vcd opened for output.
a = 0000000000000000000000000000000000000000000000000000000000001011
b = 0000000000000000000000000000000000000000000000000000000000101010
res = 1111111111111111111111111111111111111111111111111111111111100001
overflow = 0
zero = 0
Check:                   0

a = 0000000000000000000000000000000000010010000101010011010100100100
b = 1111111111111111111111111111111111000000100010010101111010000001
res = 0000000000000000000000000000000001010001100010111101011010100011
overflow = 0
zero = 0
Check:                   0

a = 1111111111111111111111111111111110000100100001001101011000001001
b = 1111111111111111111111111111111110110001111100000101011001100011
res = 1111111111111111111111111111111111010010100101000111111110100110
overflow = 0
zero = 0
Check:                   0

a = 0000000000000000000000000000000000000110101110010111101100001101
b = 0000000000000000000000000000000001000110110111111001100110001101
res = 1111111111111111111111111111111110111111110110011110000110000000
overflow = 0
zero = 0
Check:                   0

a = 1111111111111111111111111111111110110010110000101000010001100101
b = 1111111111111111111111111111111110001001001101110101001000010010
res = 0000000000000000000000000000000000101001100010110011001001010011
overflow = 0
zero = 0
Check:                   0

a = 0000000000000000000000000000000000000000111100111110001100000001
b = 0000000000000000000000000000000000000110110101111100110100001101
res = 1111111111111111111111111111111111111010000111000001010111110100
overflow = 0
zero = 0
Check:                   0

a = 0000000000000000000000000000000000111011001000111111000101110110
```

```
b = 0000000000000000000000000000000000011110100011011100110100111101
res = 0000000000000000000000000000000000011100100101100010010000111001
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001110110110101000101011111101101
b = 0000000000000000000000000000000001000110001011011111011110001100
res = 0000000000000000000000000000000000110000101001100110000001100001
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001111100111111011110100111111001
b = 1111111111111111111111111111111111100011001101110010010011000110
res = 0000000000000000000000000000000010011001110001101100010100110011
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111111100010111101111000010011000101
b = 1111111111111111111111111111111111010101000100111101001010101010
res = 0000000000000000000000000000000000001101111000111011001000011011
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001110010101011111111011111100101
b = 1111111111111111111111111111111110111011110100100111001001110111
res = 0000000000000000000000000000000010110110110111011000010101101110
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111110001001001100101101011000010010
b = 0000000000000000000000000000000001000111111011001101101110001111
res = 1111111111111111111111111111111101000001010001011111101010000011
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001111001001100000110100111110010
b = 1111111111111111111111111111111111100111011101101001011011001110
res = 0000000000000000000000000000000010010001101110011101001100100100
overflow = 0
zero = 0
```

```
Check:                    0

a = 1111111111111111111111111111111111110100000000000111101011101000
b = 1111111111111111111111111111111111100010110010100100111011000101
res = 0000000000000000000000000000000000010001001101100010110000100011
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000101110010110000100100101011100
b = 1111111111111111111111111111111111011110100011100010100010111101
res = 0000000000000000000000000000000001001111110010100010000010011111
overflow = 0
zero = 0
Check:                    0
```

Results for the AND function

```
VCD info: dumpfile ALU_64.vcd opened for output.
a = 0000000000000000000000000000000000000000000000000000000000001011
b = 0000000000000000000000000000000000000000000000000000000000101010
res = 0000000000000000000000000000000000000000000000000000000000001010
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000010010000101010011010100100100
b = 1111111111111111111111111111111111000000100010010101111010000001
res = 0000000000000000000000000000000000000000000000010001010000000000
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111110000100100001001101011000001001
b = 1111111111111111111111111111111110110001111100000101011001100011
res = 1111111111111111111111111111111110000000100000000101011000000001
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000000110101110010111101100001101
b = 0000000000000000000000000000000001000110110111111001100110001101
res = 0000000000000000000000000000000000000110100110010001100100001101
overflow = 0
zero = 0
Check:                    0
```

```
a = 1111111111111111111111111111111110110010110000101000010001100101
b = 1111111111111111111111111111111110001001001101110101001000010010
res = 1111111111111111111111111111111110000000000000100000000000000000
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000000000111100111110001100000001
b = 0000000000000000000000000000000000000110110101111100110100001101
res = 0000000000000000000000000000000000000000110100111100000100000001
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000111011001000111111000101110110
b = 0000000000000000000000000000000000011110100011011100110100111101
res = 0000000000000000000000000000000000011010000000011100000100110100
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000001110110110101000101010111111101101
b = 0000000000000000000000000000001000110001011011111110111110001100
res = 0000000000000000000000000000001000110000001000101010111110001100
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000011110011111101111010011111111001
b = 1111111111111111111111111111111111100011001101110010010011000110
res = 0000000000000000000000000000000001100000001101010010000011000000
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111111100010111101111000010011000101
b = 1111111111111111111111111111111111010101000100111101001010101010
res = 1111111111111111111111111111111111000000000100111000000010000000
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001110010101011111111011111100101
b = 1111111111111111111111111111111110111011110100100111001001110111
res = 0000000000000000000000000000000000110010100000100111001001100101
```

```
overflow = 0
zero = 0
Check:                   0

a = 1111111111111111111111111111111110001001001100101101011000010010
b = 0000000000000000000000000000000001000111111011001101101110001111
res = 0000000000000000000000000000000000000001001000001101001000000010
overflow = 0
zero = 0
Check:                   0

a = 0000000000000000000000000000000001111001001100000110100111110010
b = 1111111111111111111111111111111111100111011101101001011011001110
res = 0000000000000000000000000000000001100001001100000000000011000010
overflow = 0
zero = 0
Check:                   0

a = 1111111111111111111111111111111111110100000000000111101011101000
b = 1111111111111111111111111111111111100010110010100100111011000101
res = 1111111111111111111111111111111111100000000000000100101011000000
overflow = 0
zero = 0
Check:                   0

a = 0000000000000000000000000000000000101110010110000100100101011100
b = 1111111111111111111111111111111111011110100011100010100010111101
res = 0000000000000000000000000000000000001110000010000000100000011100
overflow = 0
zero = 0
Check:                   0
```

Results for the XOR function

```
VCD info: dumpfile ALU_64.vcd opened for output.
a = 0000000000000000000000000000000000000000000000000000000000001011
b = 0000000000000000000000000000000000000000000000000000000000101010
res = 0000000000000000000000000000000000000000000000000000000000100001
overflow = 0
zero = 0
Check:                   0

a = 0000000000000000000000000000000000010010000101010011010100100100
b = 1111111111111111111111111111111111000000100010010101111010000001
res = 1111111111111111111111111111111111010010100111000110101110100101
overflow = 0
```

```
zero = 0
Check:                    0

a = 1111111111111111111111111111111110000100100001001101011000001001
b = 1111111111111111111111111111111110110001111100000101011001100011
res = 0000000000000000000000000000000011010101110100100000000001101010
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000000110101110010111101100001101
b = 0000000000000000000000000000000001000110110111111001100110001101
res = 0000000000000000000000000000000001000000011001101110001010000000
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111110110010110000101000010001100101
b = 1111111111111111111111111111111110001001001101110101001000010010
res = 0000000000000000000000000000000000111011111101011101011001110111
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000000000111100111110001100000001
b = 0000000000000000000000000000000000000110110101111100110100001101
res = 0000000000000000000000000000000000000110001001000010111000001100
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000111011001000111111000101110110
b = 0000000000000000000000000000000000011110100011011100110100111101
res = 0000000000000000000000000000000000100101101011100011110001001011
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001110110110101000101011111101101
b = 0000000000000000000000000000000001000110001011011111011110001100
res = 0000000000000000000000000000000001100001111100110100000001100001
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001111100111111011110100111111001
```

```
b = 1111111111111111111111111111111111100011001101110010010011000110
res = 1111111111111111111111111111111111001111110010101100110100111111
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111111110001011101111000010011000101
b = 1111111111111111111111111111111111110101010001001111010010101010
res = 0000000000000000000000000000000000110111111001000101011001101111
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001110010101011111111011111100101
b = 1111111111111111111111111111111110111011110100100111001001110111
res = 1111111111111111111111111111111111001001011111011000010110010010
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111110001001001100101101011000010010
b = 0000000000000000000000000000000000100011111011001101101110001111
res = 1111111111111111111111111111111111001110110111100000110110011101
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000001111001001100000110100111110010
b = 1111111111111111111111111111111111100111011101101001011011001110
res = 1111111111111111111111111111111110011110010001101111111100111100
overflow = 0
zero = 0
Check:                    0

a = 1111111111111111111111111111111111110100000000000111101011101000
b = 1111111111111111111111111111111111100010110010100100111011000101
res = 0000000000000000000000000000000000010110110010100011010000101101
overflow = 0
zero = 0
Check:                    0

a = 0000000000000000000000000000000000101110010110000100100101011100
b = 1111111111111111111111111111111111011110100011100010100010111101
res = 1111111111111111111111111111111111110000110101100110000111100001
overflow = 0
zero = 0
```

```
Check:                        0
```

The results are as expected. The opcodes are given as input in the respective testbenches and is compiled along with the ALU code and hence, the working of the wrapper module as well as the individual modules is verified.