Report

Team Name: atmicro

Approach

ADD_64

- ADD_1
  We designed a 1 bit full adder, ADD_1, that takes 1 bit input of three registers and returns the sum and carry
- We then instantiated the ADD_1 module inside the ADD_64 module for each bit (64 times).
- We store the resulting sum and carry in a 64 bit signed register

SUB_64

- NOT_64
  We designed 64 bit NOT gate, that takes 64 bit input and returns the complement of the number by taking NOT of each bit
- We instantiated the NOT_64 inside SUB_64 to get the 1s complement of the second operand.
- We used ADD_64 inside SUB_64, to add 1 to the complement to get the 2s complement.
- The twos complement is added to the first operand using ADD_64.

XOR_64

- We iterated through each bit and called the inbuilt xor function to calculate the xor of the input bits.

AND_64

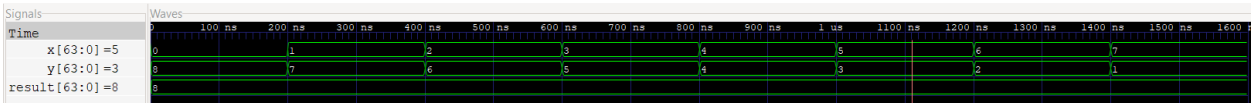
- We iterated through each bit and called the inbuilt and function to calculate the and of the input bits.

Control Input to ALU

- We defined a parameter “crtl” in wrapper_test.v, which determines the kind of operation to be carried out.
    - crtl = 0 → Addition
    - crtl = 1 → Subtraction
    - crtl = 2 → AND Operation
    - crtl = 3 → XOR Operation
- We plan to modify the code such that crtl can be controlled using other modules, but as of now crtl needs to be manually changed in the wrapper_test.v file in order to perform different operations.

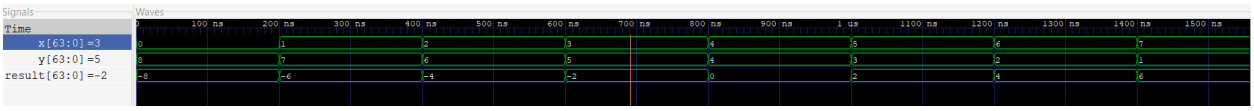
Results

1. ADD



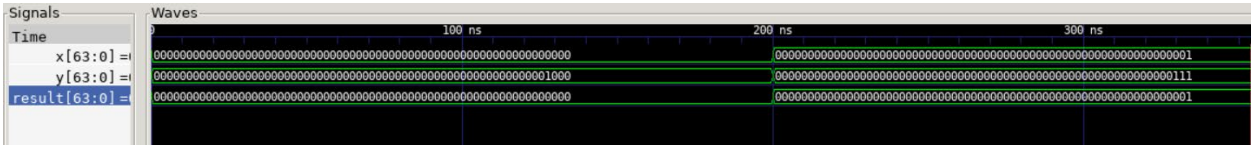

```
VCD info: dumpfile wrapper_test.vcd opened for output.
0 + 8 = 0 8
1 + 7 = 0 8
2 + 6 = 0 8
3 + 5 = 0 8
4 + 4 = 0 8
5 + 3 = 0 8
6 + 2 = 0 8
7 + 1 = 0 8
```

2. SUBTRACT



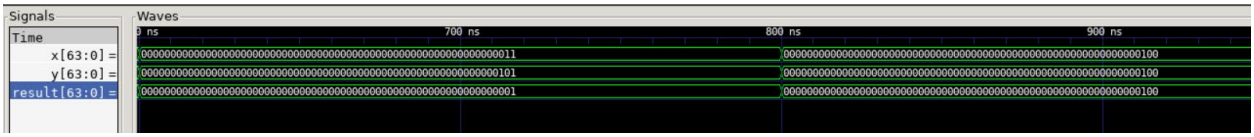
```
VCD info: dumpfile wrapper_test.vcd opened for output.
0 - 8 = -8
1 - 7 = -6
2 - 6 = -4
3 - 5 = -2
4 - 4 = 0
5 - 3 = 2
6 - 2 = 4
7 - 1 = 6
```

3. AND




```
VCD info: dumpfile wrapper_test.vcd opened for output.
0000000000000000000000000000000000000000000000000000000000000000 & 0000000000000000000000000000000000000000000000000000000000001000 = 0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000001 & 0000000000000000000000000000000000000000000000000000000000000111 = 0000000000000000000000000000000000000000000000000000000000000001
0000000000000000000000000000000000000000000000000000000000000010 & 0000000000000000000000000000000000000000000000000000000000000110 = 0000000000000000000000000000000000000000000000000000000000000010
0000000000000000000000000000000000000000000000000000000000000011 & 0000000000000000000000000000000000000000000000000000000000000101 = 0000000000000000000000000000000000000000000000000000000000000001
0000000000000000000000000000000000000000000000000000000000000100 & 0000000000000000000000000000000000000000000000000000000000000100 = 0000000000000000000000000000000000000000000000000000000000000100
0000000000000000000000000000000000000000000000000000000000000101 & 0000000000000000000000000000000000000000000000000000000000000011 = 0000000000000000000000000000000000000000000000000000000000000001
0000000000000000000000000000000000000000000000000000000000000110 & 0000000000000000000000000000000000000000000000000000000000000010 = 0000000000000000000000000000000000000000000000000000000000000010
0000000000000000000000000000000000000000000000000000000000000111 & 0000000000000000000000000000000000000000000000000000000000000001 = 0000000000000000000000000000000000000000000000000000000000000001
```

4. XOR




```
VCD info: dumpfile wrapper_test.vcd opened for output.
0000000000000000000000000000000000000000000000000000000000000000 ^ 0000000000000000000000000000000000000000000000000000000000001000 = 0000000000000000000000000000000000000000000000000000000000001000
0000000000000000000000000000000000000000000000000000000000000001 ^ 0000000000000000000000000000000000000000000000000000000000000111 = 0000000000000000000000000000000000000000000000000000000000000110
0000000000000000000000000000000000000000000000000000000000000010 ^ 0000000000000000000000000000000000000000000000000000000000000110 = 0000000000000000000000000000000000000000000000000000000000000100
0000000000000000000000000000000000000000000000000000000000000011 ^ 0000000000000000000000000000000000000000000000000000000000000101 = 0000000000000000000000000000000000000000000000000000000000000110
0000000000000000000000000000000000000000000000000000000000000100 ^ 0000000000000000000000000000000000000000000000000000000000000100 = 0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000101 ^ 0000000000000000000000000000000000000000000000000000000000000011 = 0000000000000000000000000000000000000000000000000000000000000110
0000000000000000000000000000000000000000000000000000000000000110 ^ 0000000000000000000000000000000000000000000000000000000000000010 = 0000000000000000000000000000000000000000000000000000000000000100
0000000000000000000000000000000000000000000000000000000000000111 ^ 0000000000000000000000000000000000000000000000000000000000000001 = 0000000000000000000000000000000000000000000000000000000000000110
```