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

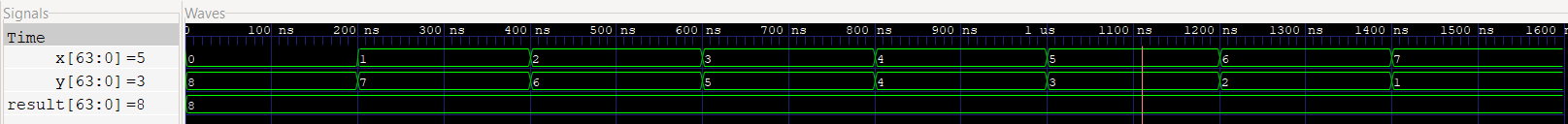
* We iterated through each bit and called the inbuilt and function to calculate the and of the input bits.

**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**

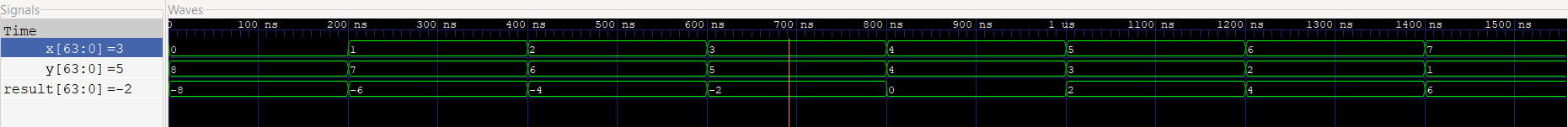
1. ADD



Text

Description automatically generated

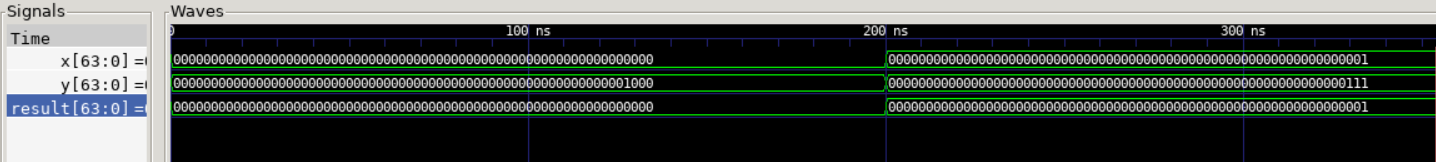
2. SUBTRACT

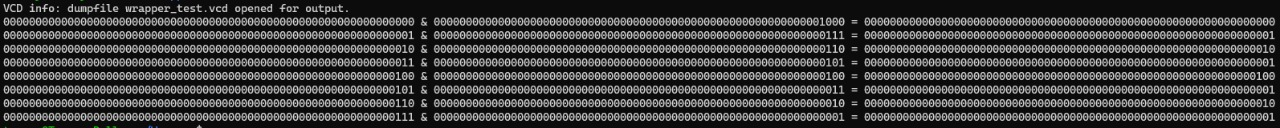


Text

Description automatically generated

3. AND





4. XOR

