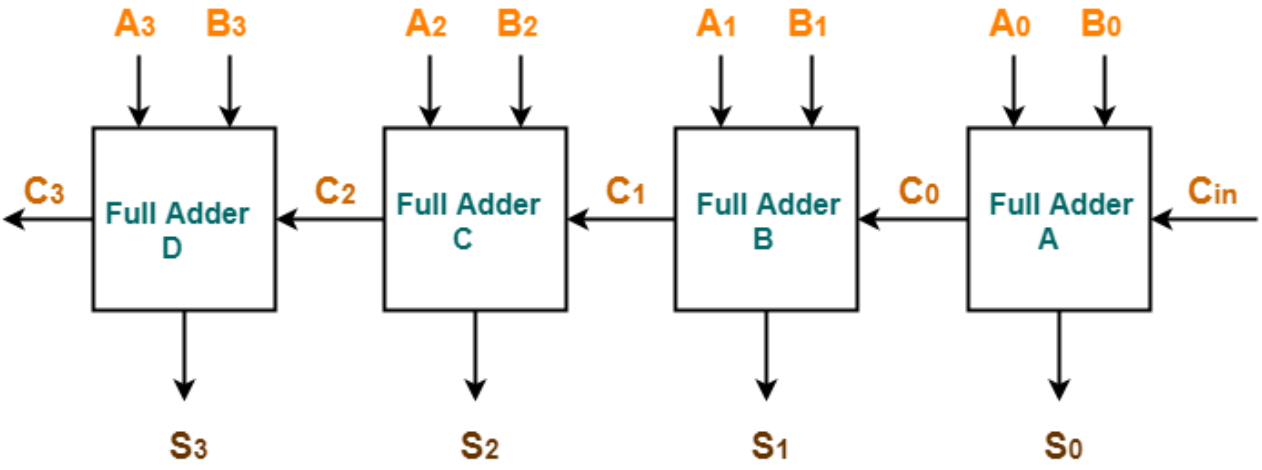
IPA Assignment 1

Prasoon Garg - 2020102049
Prashant Gupta - 2020102030

➢ *Approach*

1. ADD Module : In this module we use the Ripple carry adder fashion of adding numbers, where there are full adders present which require the carry out of the previous stage to compute the sum and carry out of the current stage.We have written a loop to iterate over all the 64 bits and store the out and carry results in a 64 bit register. Also we have written modules for half adder and full adder. The miniature version of the 64 bit adder is as shown below.



We have also implemented an overflow warning in the addition module which gives a overflow flag/warning when the xor of the last and the second last carry in the ripple carry adder is equal to 1.

2. SUB Module : We have implemented the subtraction operation using the 2's complement method. The inputs to this module are the 2 64 bits numbers. Then the 2nd number's 1's complement is taken (whose module is seperately written) and then the 1st number and 1's complement of the 2nd number are passed to the ADD module along with c_in = 1 which will make the 1's complement as 2's complement. Finally the result is stored in an output register.

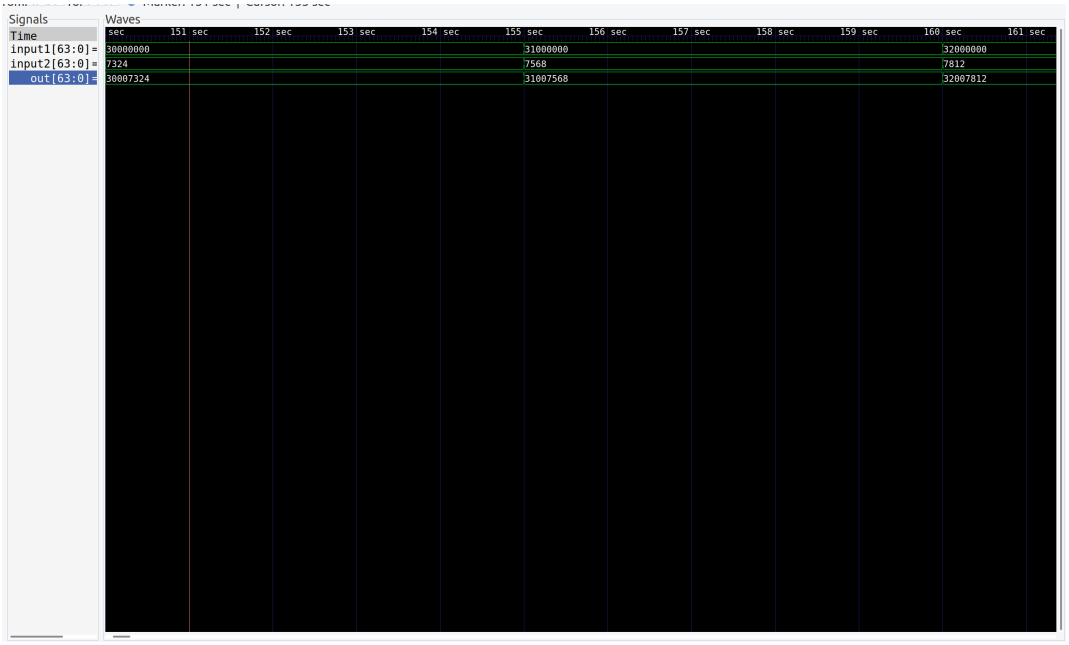
3. AND Module : In this Module the 2 64 bit numbers are taken and the AND is taken individually of each bit by iterating through the loop 64 times and output is stored in a 64 bit output register.

4. XOR Module : This module approach is similar to the AND module. We perform the same iterations as in the AND module but instead of AND we perform the XOR operation and store the result in a 64 bit output register.

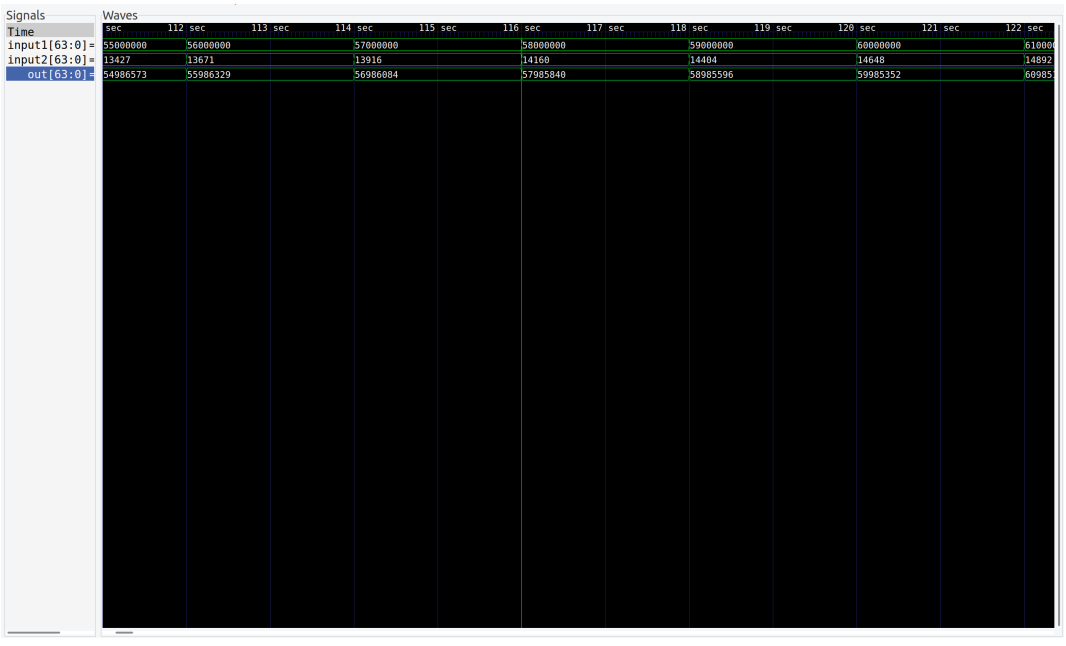
5. ALU Module : This is the wrapper module which combines all the above 4 operations into a single module and hence we can say that this module is a kind of multiplexer. Here we take a control input along with the 2 64 bit numbers which decide which operation to be performed among addition,subtraction,AND , XOR. The approach here is slightly lengthy. We initially perform all the 4 operations on the 2 inputs and store them in 4 different registers. Then we run case conditional statements and inside these we assign the alu output to the output of one of the operations according to the value of control and the alu output is then returned as the result.

➢ *Results*

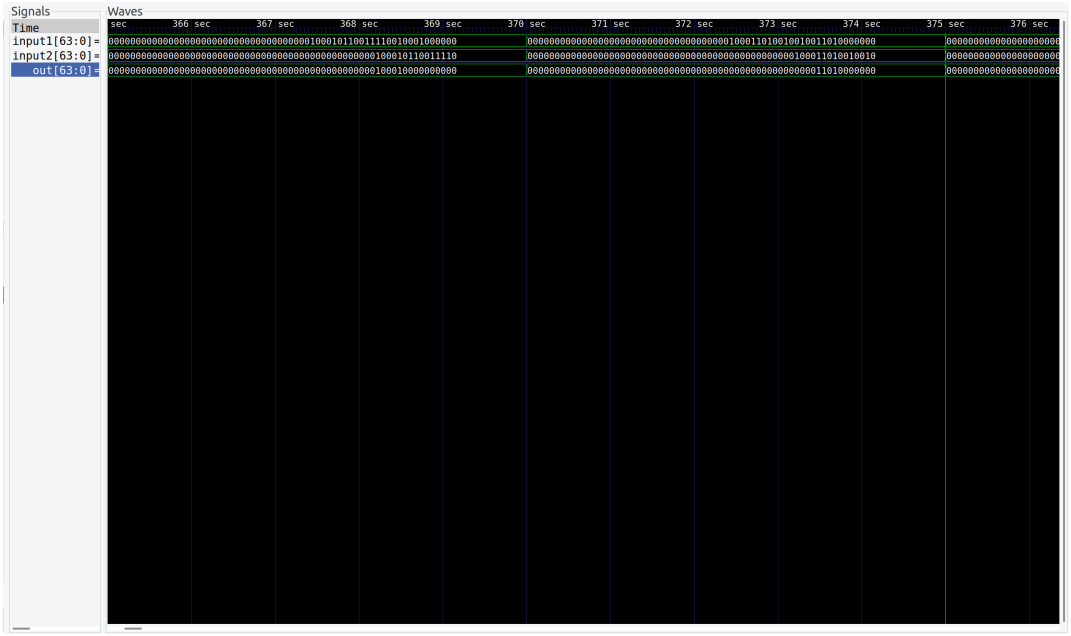
1. ADD Module



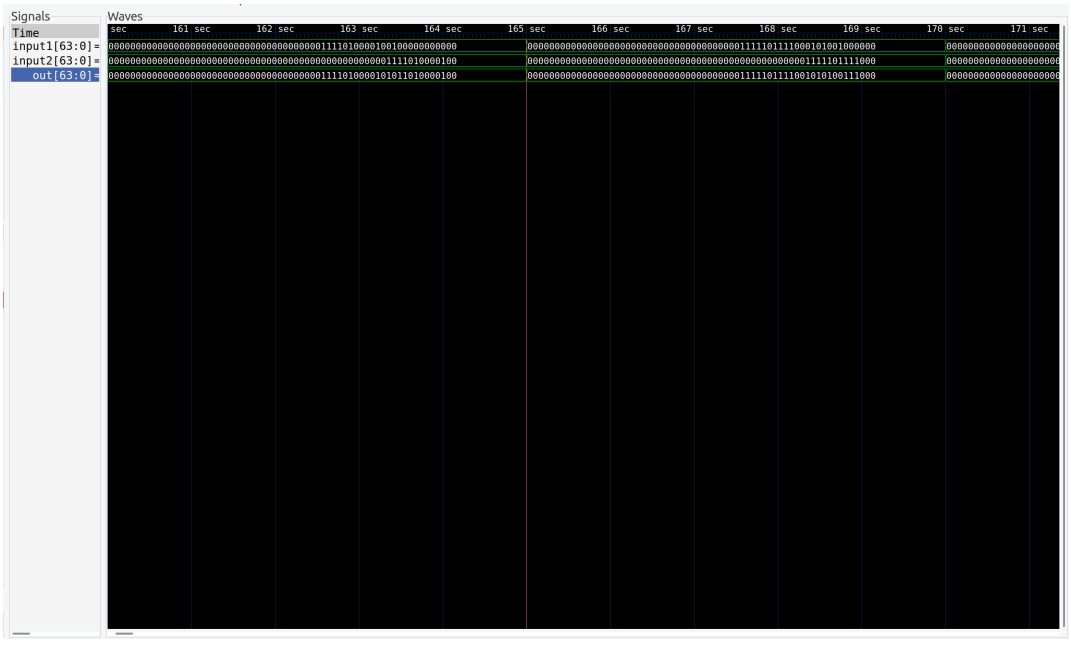
2. SUB Module



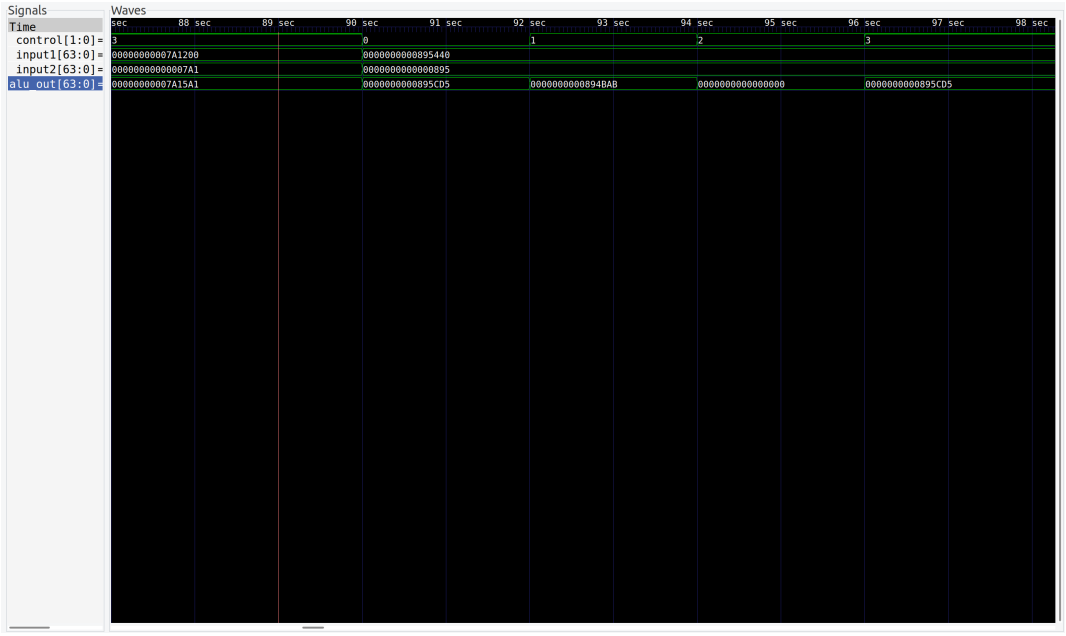
3. AND Module



4. XOR Module



5. ALU Module



Note : All test cases couldn’t fit in the screenshot so only few have been shown above. For more testcases we have added .vcd files in the github repository.