VERILOG CPU ASSIGNMENT

Here in this assignment, we need to design a CPU - ALU unit that takes in an instruction and then processes it such that we get the operation that is to be performed on the two operands.
We get a 19-bit instruction that has its first 3 bits as the operation and then the remaining 16 bits as two 8-bit operands.

We decode the 3 bits using a 3x8 Decoder and then make the output line i one if the input line sequence makes i using its binary representation.

After decoding we assign each line to a distinct operation result.

- 1st line is the Addition result of both the operands
- 2nd line is the Subtraction result of both the operands
- 3rd line is the Increment result of the first operand
- 4th line is the Decrement result of the first operand
- 5th line is the bitwise - AND result of both the operands
- 6th line is the bitwise - OR result of both the operands
- 7th line is the bitwise - NOT result of the first operand

The increment, decrement, addition and subtraction are performed using the Ripple Carry 8-bit adder. These operations are done without any carry generation.

The logical flow of the code is well explained in the code using the comments and apart from that some logic that we used were of two's complement for the subtraction. Subtraction logic - Take the two's complement of the second number and then add it to the first number.

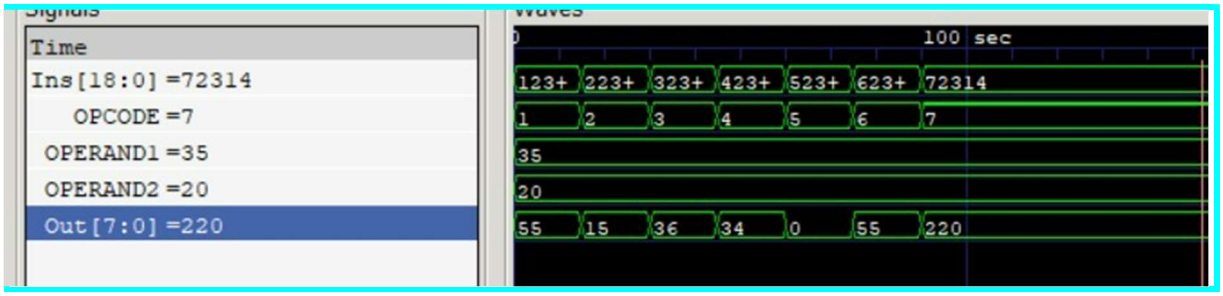
Also, the RCA is implemented using the Full - Adders that we have implemented using the Half - Adder.

The overall errors might occur when we add the numbers such that the final carry will be one. In that case as we have not taken the carry into consideration, the outputs might not match the exact answer. The solution to this is that we mark the flags as 1 whenever there is an overflow. Such operations are there in the actual CPU and hence we can find the bugs.
In order to get a better idea of the operations the output, opcode, operands, and the instruction are displayed on the terminal.

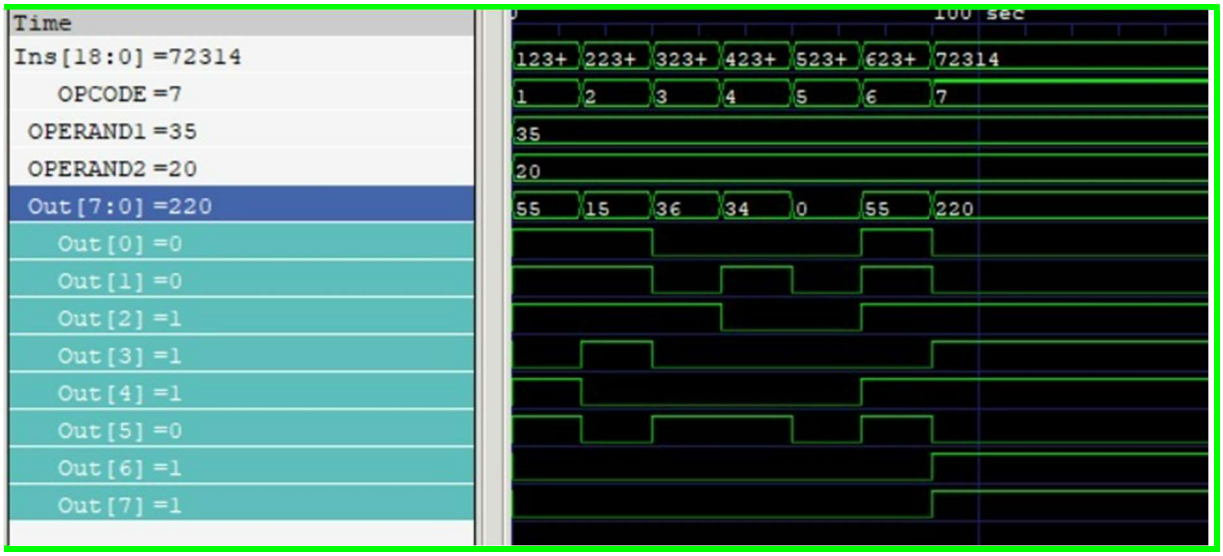
- The outputs of the Program on the terminal given the following instructions are as follows:

```
PS C:\Users\shaun\OneDrive\Desktop\Verilog Lab> vvp a.out
VCD info: dumpfile CPU.vcd opened for output.
Instruction = 0010010001100010100, OpCode = 001, Operand 1 = 00100011, Operand 2 = 00010100, Out = 00110111
Instruction = 0100010001100010100, OpCode = 010, Operand 1 = 00100011, Operand 2 = 00010100, Out = 00001111
Instruction = 0110010001100010100, OpCode = 011, Operand 1 = 00100011, Operand 2 = 00010100, Out = 00100100
Instruction = 1000010001100010100, OpCode = 100, Operand 1 = 00100011, Operand 2 = 00010100, Out = 00100010
Instruction = 1010010001100010100, OpCode = 101, Operand 1 = 00100011, Operand 2 = 00010100, Out = 00000000
Instruction = 1100010001100010100, OpCode = 110, Operand 1 = 00100011, Operand 2 = 00010100, Out = 00110111
Instruction = 1110010001100010100, OpCode = 111, Operand 1 = 00100011, Operand 2 = 00010100, Out = 11011100
TestBench.v:23: $finish called at 270 (1s)
```

- The GTK Wave output of the program is as follows :



- The Alternate GTK Wave Output that demonstrates the binary form of the output is as follows:



These outputs in the GTK Wave are represented in the decimal form for better understanding of the instructions.
The overall individual operations can be understood by checking out the waveforms of the GTK Wave.

According to the sample test bed given in the manual the operand 1 was 35 and operand 2 was 20. We have applied all the possible operations on them and demonstrated them in the above report.

—---------- **THANK YOU** —----------