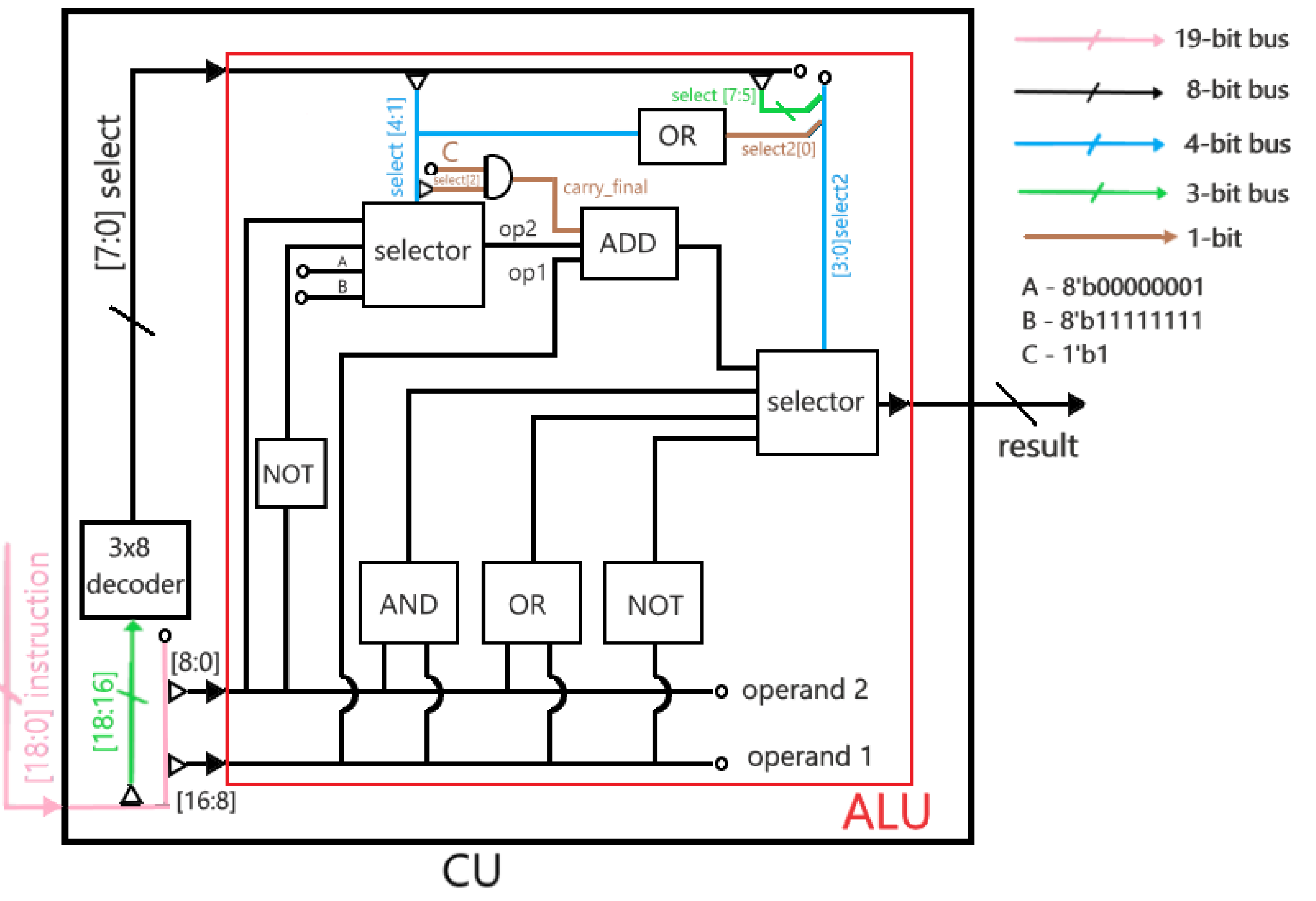
CPU Design Lab Report

CPU Design Schematic Circuit Diagram



- The schematic shows the implementation of the simple CPU design using a control unit (CU) and an arithmetic and logic unit (ALU).

- The CU takes the 19-bit instruction code as input, separates the 3-bit operation code, decodes it using a 3x8 decoder, and sends it as input to the ALU along with the two 8-bit operands derived from the instruction code.

- Now, we observe the functionalities of 8-bit addition, subtraction, increment, and decrement are similar and can be achieved using a single 8-bit adder module by changing the operand_2 and carry_bit accordingly.

- The modifications are as follows:
    - For ADD functionality, we needn't modify operand_2. The carry bit is 0.
    - For subtraction, we take the 2's complement for operand_2. This is done by giving the second operand as its 8-bit complement and the carry bit as 1.
    - For increment, we take operand_2 as the literal "8'b00000001" and carry to be 0.
    - For decrement, we take operand_2 as the literal "8'b11111111" and carry to be 0.

- To select the appropriate second operand, we use a 4x1 8-bit selector, which takes four 8-bit inputs and a 4-bit "one-hot encoded" select bus and selects the corresponding input based on the 4-bit select line, and gives an 8-bit output.

- Since we need carry to be 1 only for subtraction, we use an and-gate with its inputs as its corresponding select bit (select[2]) and the literal 1. The output is fed to the ADDer module carry input.

- We simultaneously compute the result of other computations, namely AND, OR, and NOT. These are simple bitwise operations and are realized using the basic 1-bit input gates for each bit and taking the combined output as an 8-bit bus.

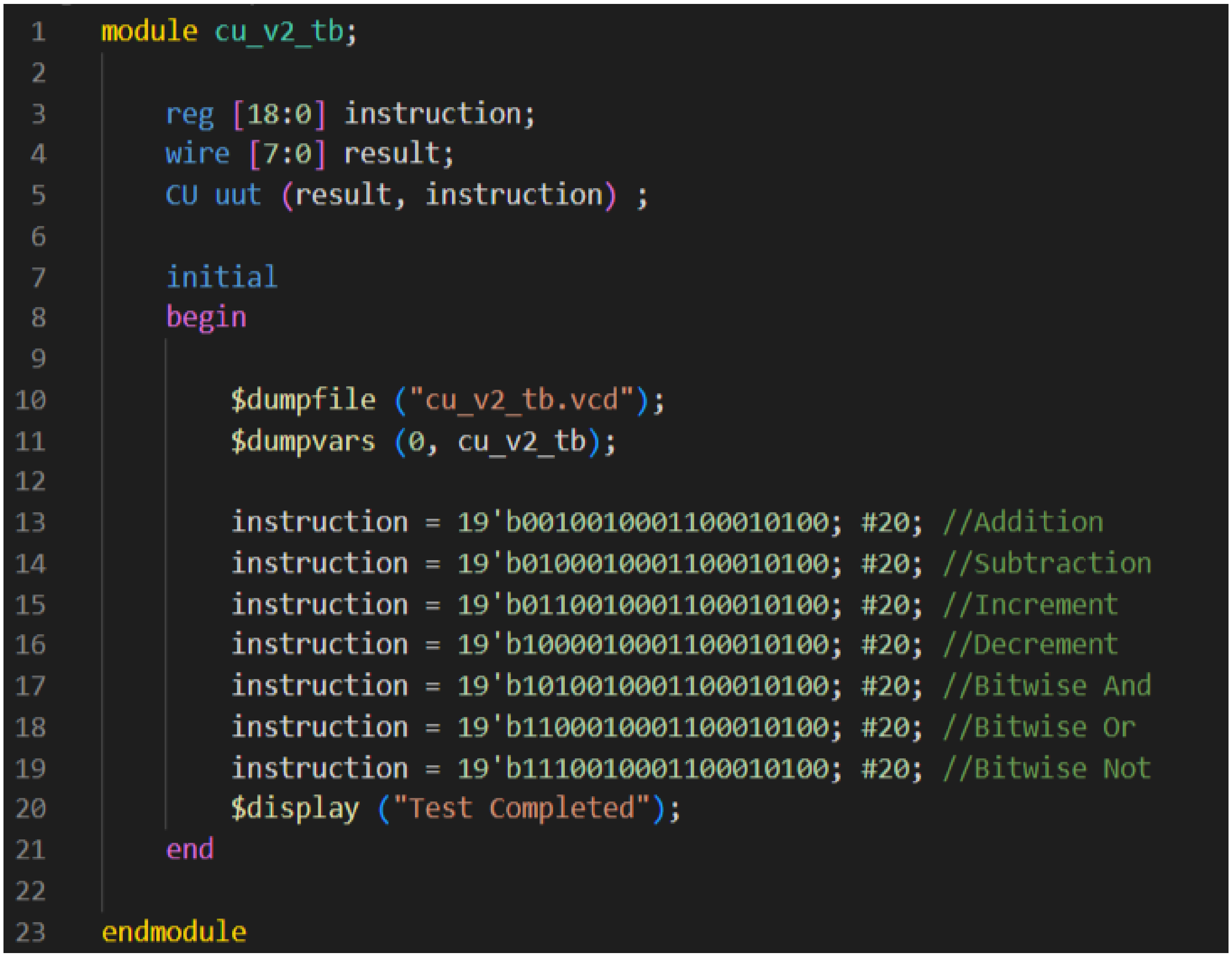
- Now, we use a second 4-bit select line to select between these outputs. The select bit corresponding to the output of the combined adder is the OR output of the select bits corresponding to these functions, i.e., select [4:1].

- Rest three select inputs are taken from select [7:5].

- Using a similar selector as above, we get the final 8-bit result.

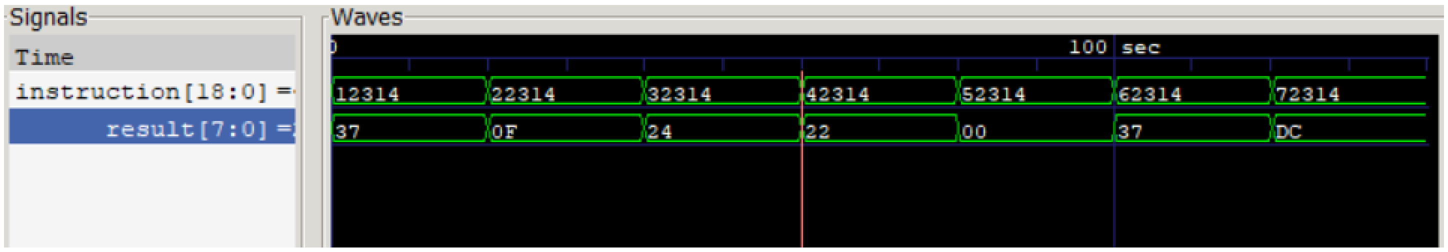
Working and Testing

- A snapshot of the testbench for the entire cpu :



```verilog
module cu_v2_tb;

    reg [18:0] instruction;
    wire [7:0] result;
    CU uut (result, instruction) ;

    initial
    begin

        $dumpfile ("cu_v2_tb.vcd");
        $dumpvars (0, cu_v2_tb);

        instruction = 19'b0010010001100010100; #20; //Addition
        instruction = 19'b0100010001100010100; #20; //Subtraction
        instruction = 19'b0110010001100010100; #20; //Increment
        instruction = 19'b1000010001100010100; #20; //Decrement
        instruction = 19'b1010010001100010100; #20; //Bitwise And
        instruction = 19'b1100010001100010100; #20; //Bitwise Or
        instruction = 19'b1110010001100010100; #20; //Bitwise Not
        $display ("Test Completed");
    end

endmodule
```

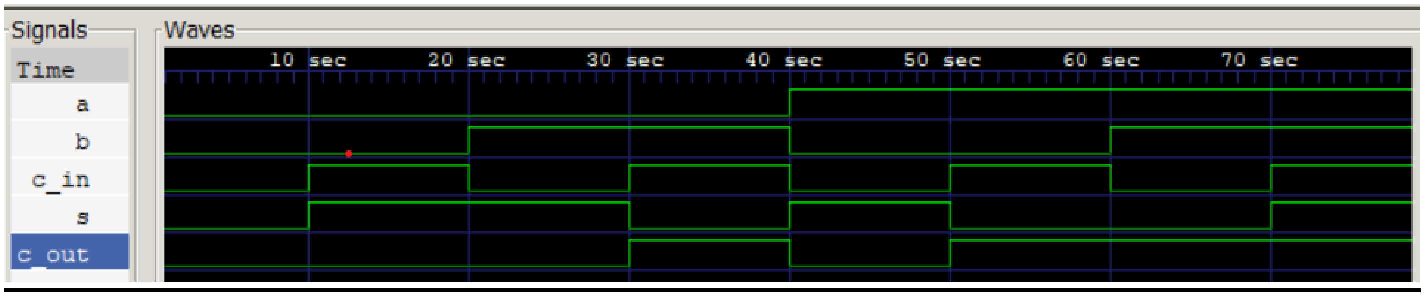
- Gtkwave output for the test bench: (results are in hexadecimal format)



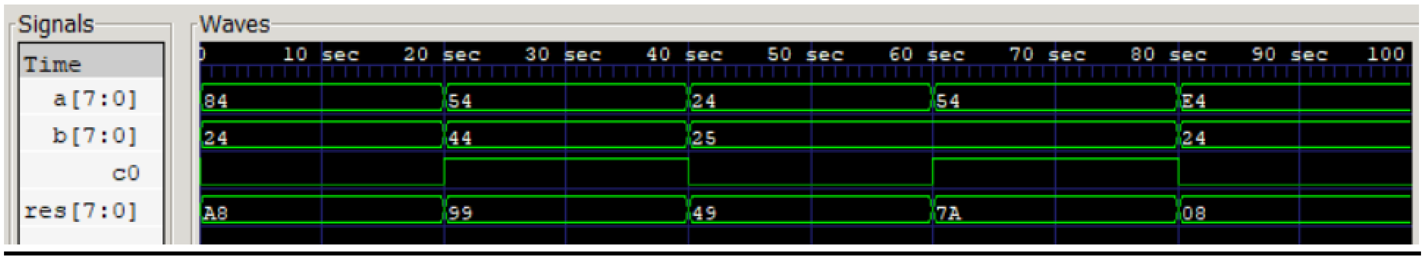
```
instruction[18:0] =  12314  22314  32314  42314  52314  62314  72314
result[7:0] =        37     0F     24     22     00     37     DC
```

List of Modules and their Functionalities

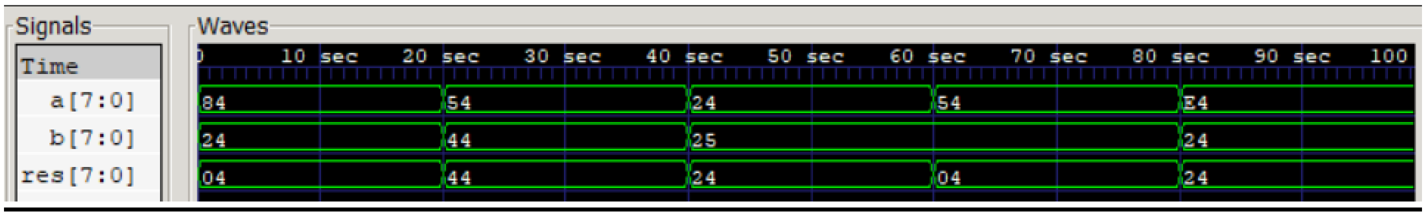
- **fa:**
- **Desc:** 1-bit full adder
- **Input:** op1, op2, carry_in
- **Output:** sum, carry_out
- **Module Pre-requisites:** None.
- **Output waveform:**



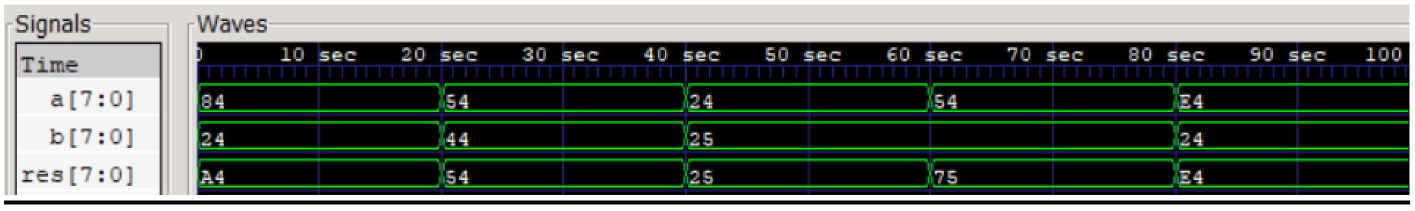
- **ADD:**
- **Desc:** 8-bit full adder
- **Input:** [7:0] op1, [7:0] op2, carry_in
- **Output:** [7:0] sum.
- **Module Pre-requisites:** fa.
- **Output waveform:**



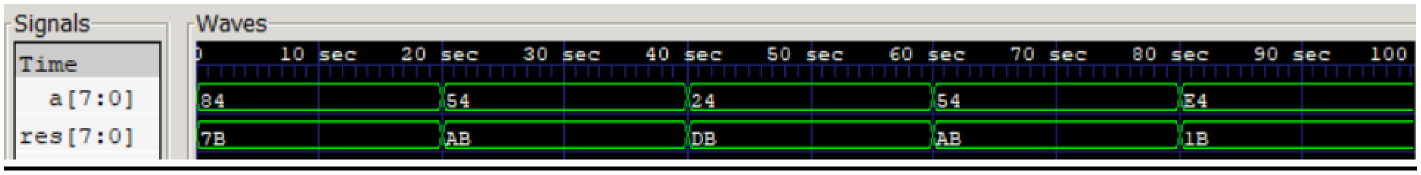
- **AND:**
- **Desc:** 8-bit Bitwise AND gate.
- **Input:** [7:0] op1, [7:0] op2
- **Output:** [7:0] res.
- **Module Pre-requisites:** None.
- **Output waveform:**



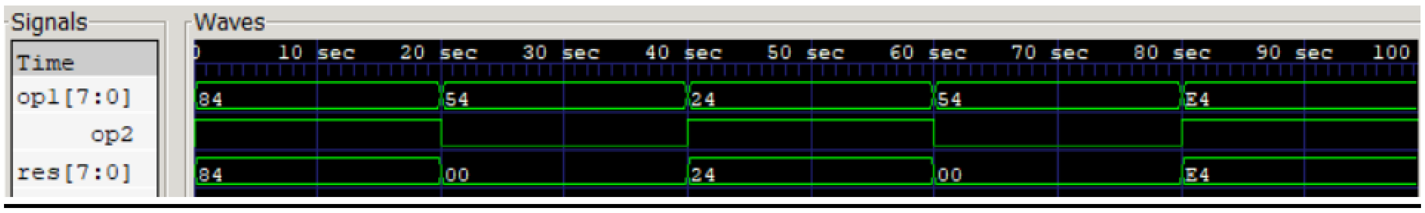
- **OR:**
- **Desc:** 8-bit Bitwise OR gate.
- **Input:** [7:0] op1, [7:0] op2
- **Output:** [7:0] res.
- **Module Pre-requisites:** None.
- **Output waveform:**



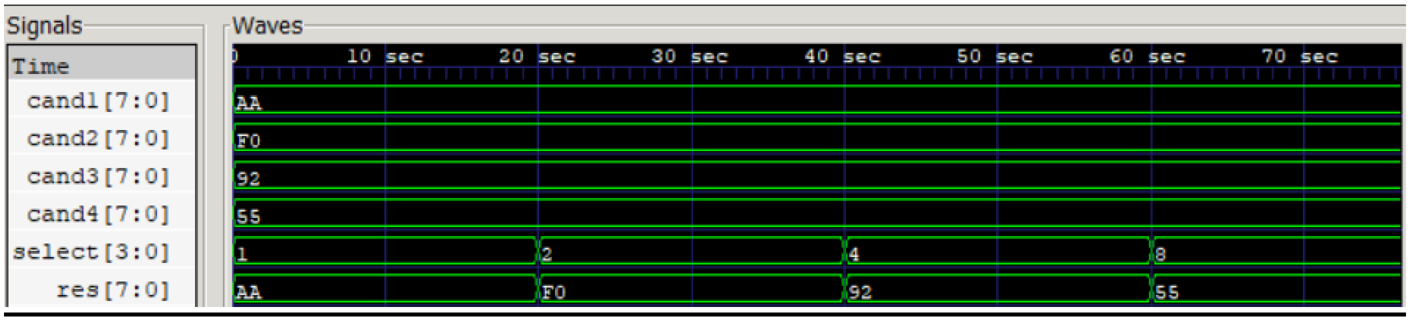
- **NOT:**
- **Desc:** 8-bit Bitwise NOT gate.
- **Input:** [7:0] op1
- **Output:** [7:0] res.
- **Module Pre-requisites:** None.
- **Output waveform:**



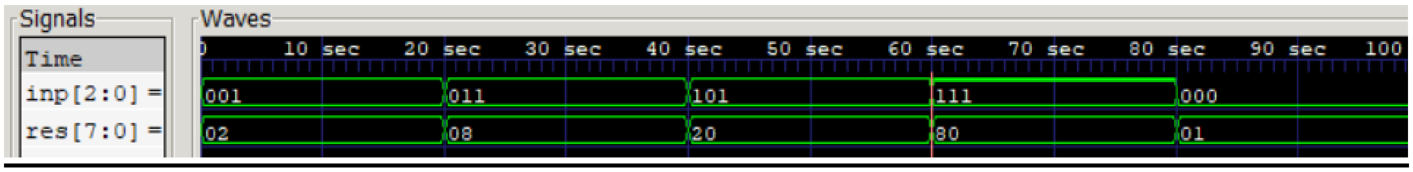
- **AND_2:**
- **Desc:** ANDs an 8-bit bus with a single bit and outputs an 8-bit bus.
- **Input:** [7:0] op1, op2.
- **Output:** [7:0] res.
- **Module Pre-requisites:** None.
- **Output waveform:**



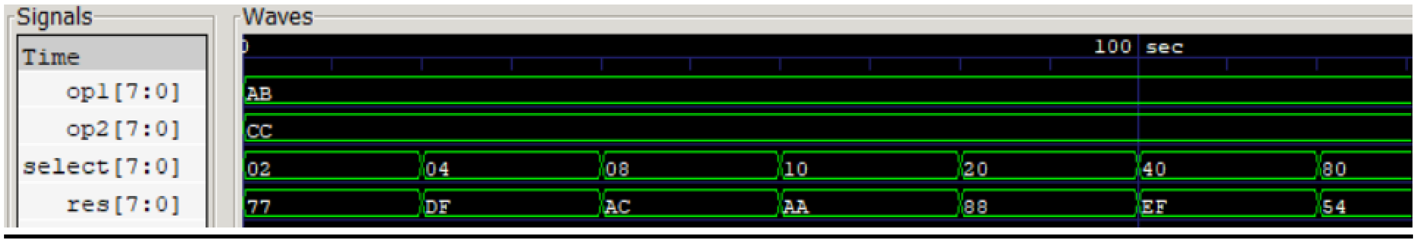
- **selector:**
- **Desc:** selects a 8-bit output from given four 8-bit inputs using an "one-hot encoded" 4-bit select line.
- **Input:** [7:0] cand1, [7:0] cand2, [7:0] cand3, [7:0] cand4, [3:0] select,
- **Output:** [7:0] res.
- **Module Pre-requisites:** AND_2.
- **Output waveform:**



- **decoder_3x8:**
- **Desc:** decodes the given 3-bit input i.e. outputs all the minterms giving a "one-hot encoded" 8-bit output.
- **Input:** [3:0] inp.
- **Output:** [7:0] res.
- **Module Pre-requisites:** None.
- **Output waveform:**



- **ALU:**
- **Desc:** Computes and returns the result of the operation specified by “one-hot encoded” 8-bit select input. Takes two 8-bit input operands.
- **Input:** [7:0] op1, [7:0] op2, [7:0] select.
- **Output:** [7:0] res.
- **Module Pre-requisites:** fa, ADD, NOT, AND, OR, AND_2, selector.
- **Output waveform:**



- **CU:**
- **Desc:** Takes a 19-bit instruction code and produces an 8-bit final result
- **Input:** [18:0] instruction.
- **Output:** [7:0] res.
- **Module Pre-requisites:** fa, ADD, NOT, AND, OR, AND_2, selector, ALU, decoder_3x8.
- **Output waveform:** Test_bench snapshot and output waveform has already been included above under “Working and Testing” Heading.