**ECE 424**

# Team Design #2: 32-bit ALU

## COVER SHEET

Honor Code: ____I have neither given or received, nor have I tolerated others' use _____

_____of unauthorized aid._____

Name:_____Joe Leveille_____ Signature: ____*Joseph Leveille*_____

Honor Code: ___ I have neither given or received, nor have I tolerated others' use _____

_____ of unauthorized aid._____

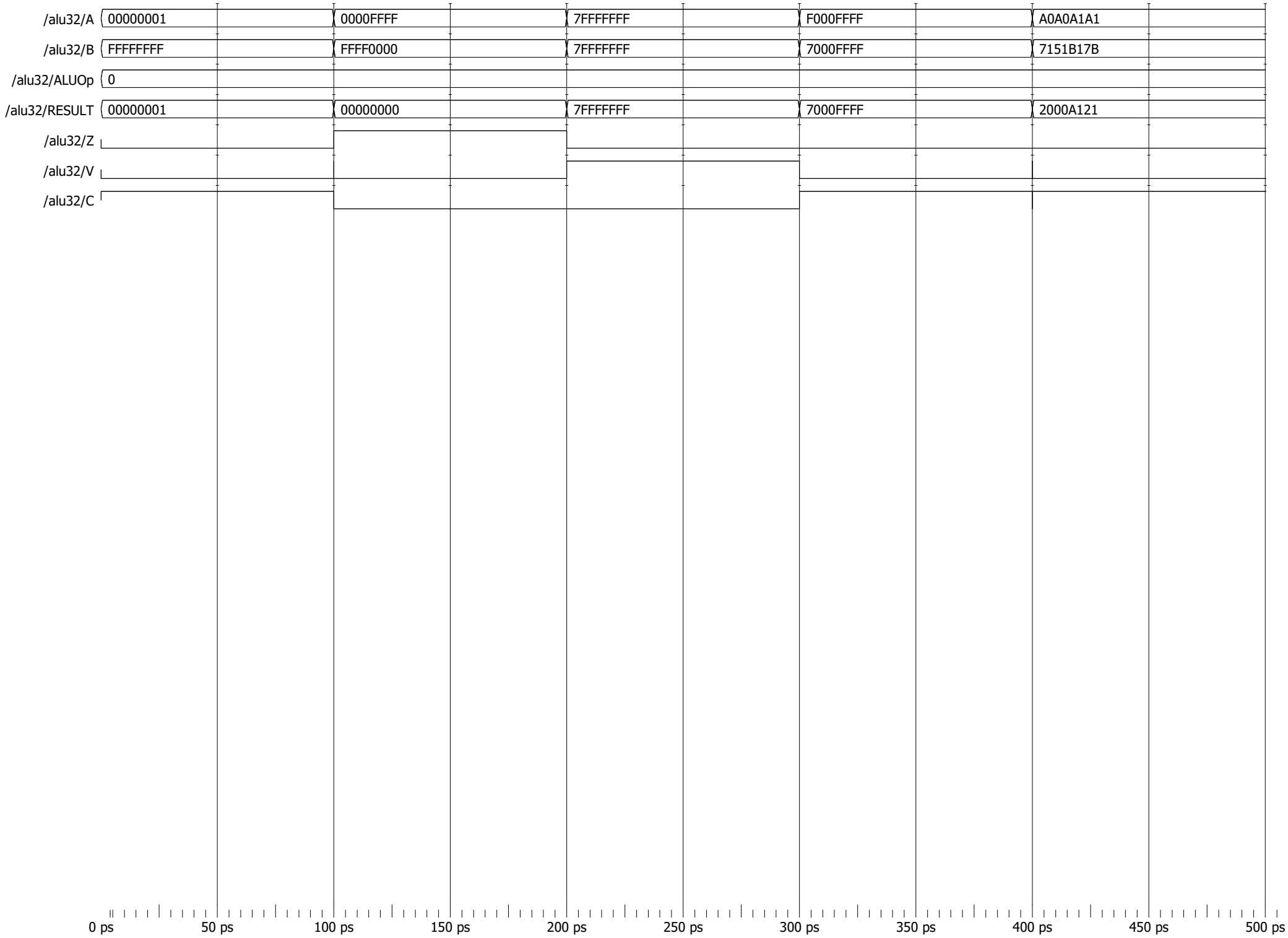
Name:_____Jon Bayert_____ Signature: ____Jonathan Bayert_____

Include:
0) This coverpage stapled on the front
1) Your VHDL code
2) Your waveform results: 6 pages, one for each function

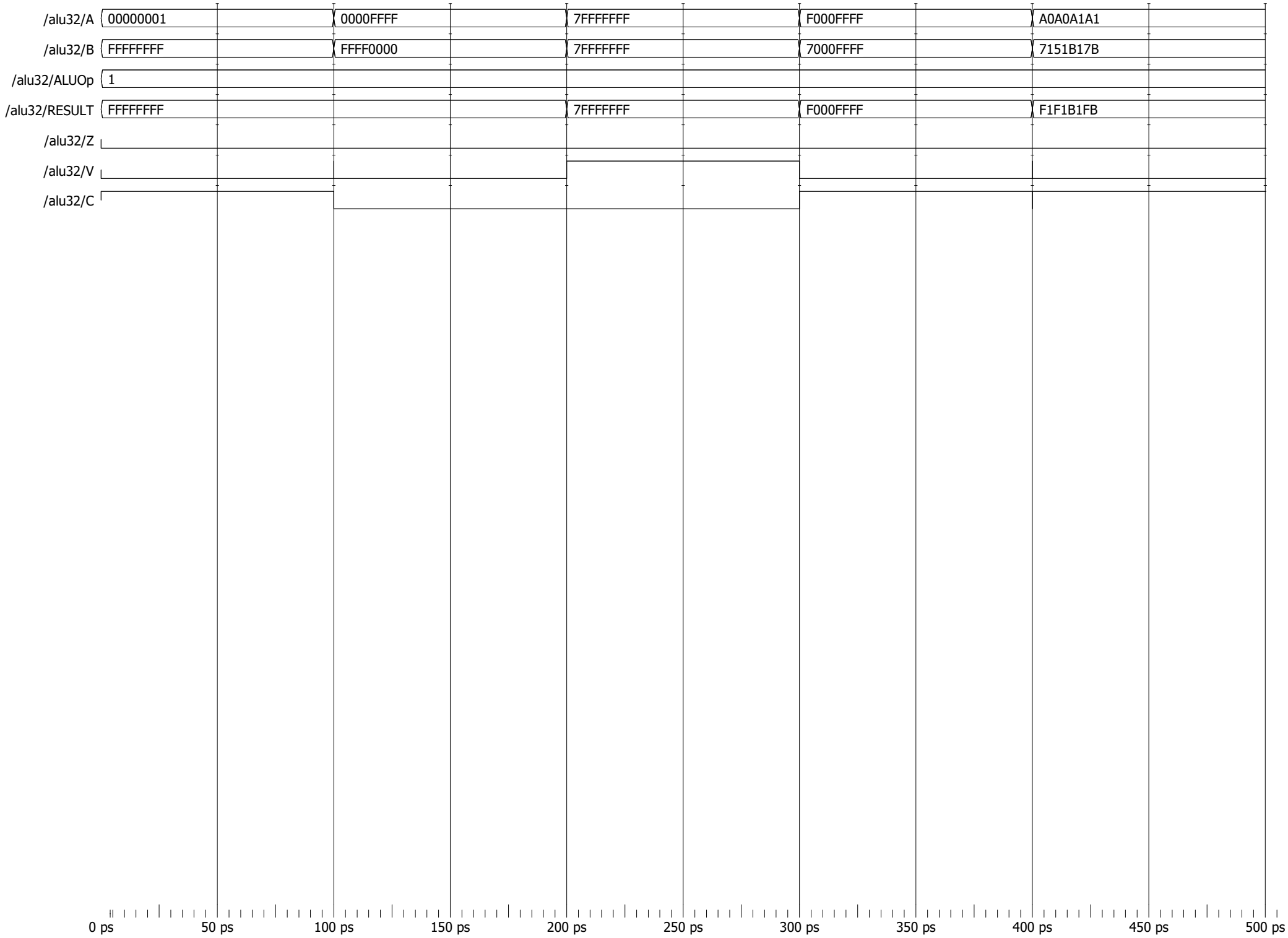
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_misc.or_reduce;
ENTITY alu32 IS
    PORT( A, B: in std_logic_vector (31 downto 0);
        ALUOp: in std_logic_vector (3 downto 0);
        RESULT: out std_logic_vector (31 downto 0);
        Z, V, C: out std_logic );
END alu32;

architecture big_alu of alu32 is
    signal intRes, carry: std_logic_vector(31 downto 0);
    signal set : std_logic;
    component bit_slice port(
        a,b,Less,Ainvert,carryIn        : in std_logic;
        Op                              : in std_logic_vector (2 downto 0);
        result,carryOut                 : out std_logic
     ); end component;
     component MSB_slice port(
        a,b,Less,Ainvert,carryIn        : in std_logic;
        Op                              : in std_logic_vector (2 downto 0);
        result,carryOut,set,overflow    : out std_logic
     ); end component;
begin
    LSB: bit_slice PORT MAP(
        a=>A(0), b=>B(0), Less=>set, Ainvert=>ALUOp(3), carryIn=>ALUOp(2),
        Op=>ALUOp(2 downto 0),
        result=>intRes(0), carryOut=>carry(0) );
    bitSliceSetup: for x in 1 to 30 generate
        BITx: bit_slice PORT MAP(
            a=>A(x), b=>B(x), Less=>'0', Ainvert=>ALUOp(3), carryIn=>carry(x-1),
            Op=>ALUOp(2 downto 0),
            result=>intRes(x), carryOut=>carry(x) );
    end generate bitSliceSetup;
    MSB: MSB_slice PORT MAP(
            a=>A(31), b=>B(31), Less=>'0', Ainvert=>ALUOp(3), carryIn=>carry(30),
            Op=>ALUOp(2 downto 0),
            result=>intRes(31), carryOut=>C,set=>set, overflow=>V );

    --Handle zero flag
    Z <= not or_reduce(intRes);
    --connect result output to signal
    result <= intRes;

end big_alu;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity bit_slice is
  port(
    a,b,Less,Ainvert,carryIn  : in std_logic;
    Op                        : in std_logic_vector (2 downto 0);
    result,carryOut           : out std_logic
  );
end bit_slice;

architecture stdBitSlice of bit_slice is
  signal aMux, bMux, andOut, orOut  : std_logic;
  signal AxorB, AandB, sum, CandXor : std_logic;
begin
  --Inverter muxes
  aMux <= a xor Ainvert;
  bMux <= b xor Op(2);              -- Assuming Binvert is MSB of Op

  --Basic gates
  andOut <= aMux and bMux;
  orOut <= aMux or bMux;

  --Following is adder block
  AxorB <= aMux xor bMux;
  AandB <= aMux and bMux;
  CandXor <= AxorB and carryIn;
  sum <= AxorB xor carryIn ;
  carryOut <= AandB or CandXor;

  --Big mux
  result <= (not Op(1) and not Op(0) and andOut) or
            (not Op(1) and Op(0) and orOut) or
            (Op(1) and not Op(0) and sum) or
            (Op(1) and Op(0) and Less);

end stdBitSlice;

```

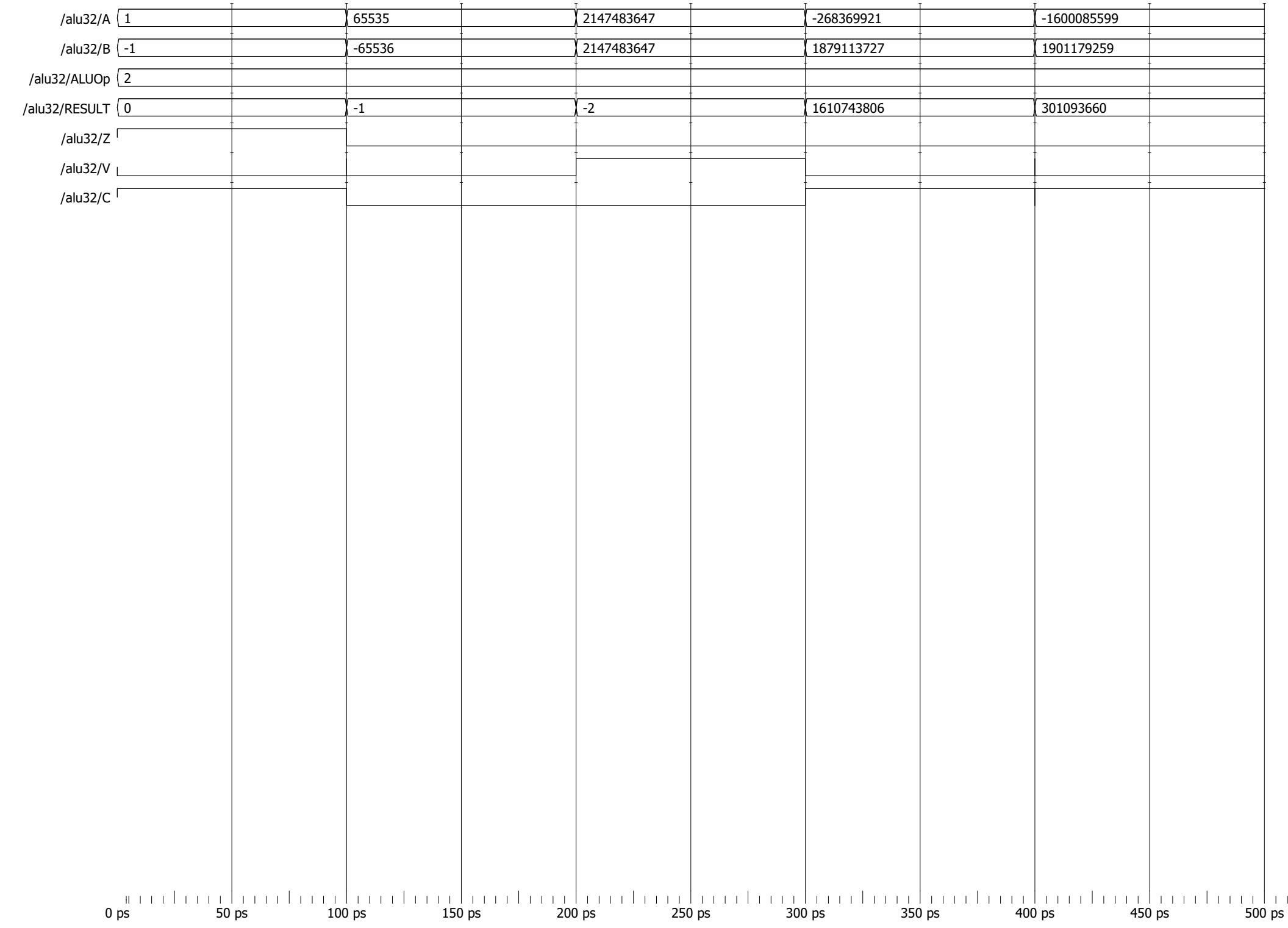
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity msb_slice is
  port(
    a,b,Less,Ainvert,carryIn      : in std_logic;
    Op                            : in std_logic_vector (2 downto 0);
    result,carryOut,set,overflow  : out std_logic
  );
end msb_slice;

architecture MSBBitSlice of msb_slice is
  signal aMux, bMux, andOut, orOut  : std_logic;
  signal AxorB, AandB, sum, CandXor : std_logic;
begin
  --Inverter muxes
  aMux <= a xor Ainvert;
  bMux <= b xor Op(2);              -- Assuming Binvert is MSB of Op

  --Basic gates
  andOut <= aMux and bMux;
  orOut <= aMux or bMux;

  --Following is adder block
  AxorB <= aMux xor bMux;
  AandB <= aMux and bMux;
  CandXor <= AxorB and carryIn;
  sum <= AxorB xor carryIn ;
  carryOut <= AandB or CandXor;

  --Big mux
  result <= (not Op(1) and not Op(0) and andOut) or
            (not Op(1) and Op(0) and orOut) or
            (Op(1) and not Op(0) and sum) or
            (Op(1) and Op(0) and Less);

  --MSB additions
  set <= sum;
  overflow <= carryIn xor (AandB or CandXor);

end MSBBitSlice;
```



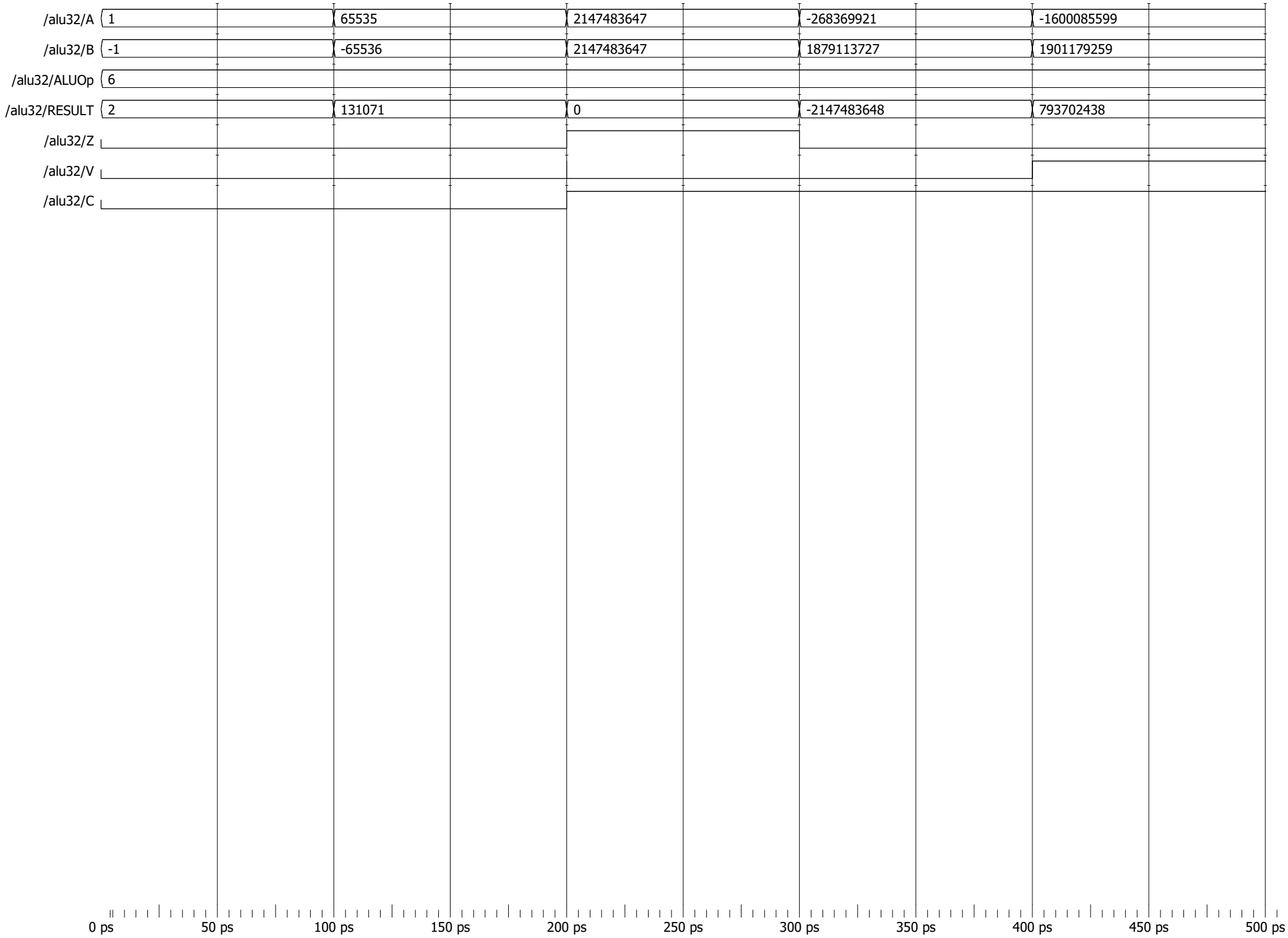

Entity:alu32 Architecture:big_alu Date: Sun Nov 01 13:29:00 CST 2020 Row: 1 Page: 1



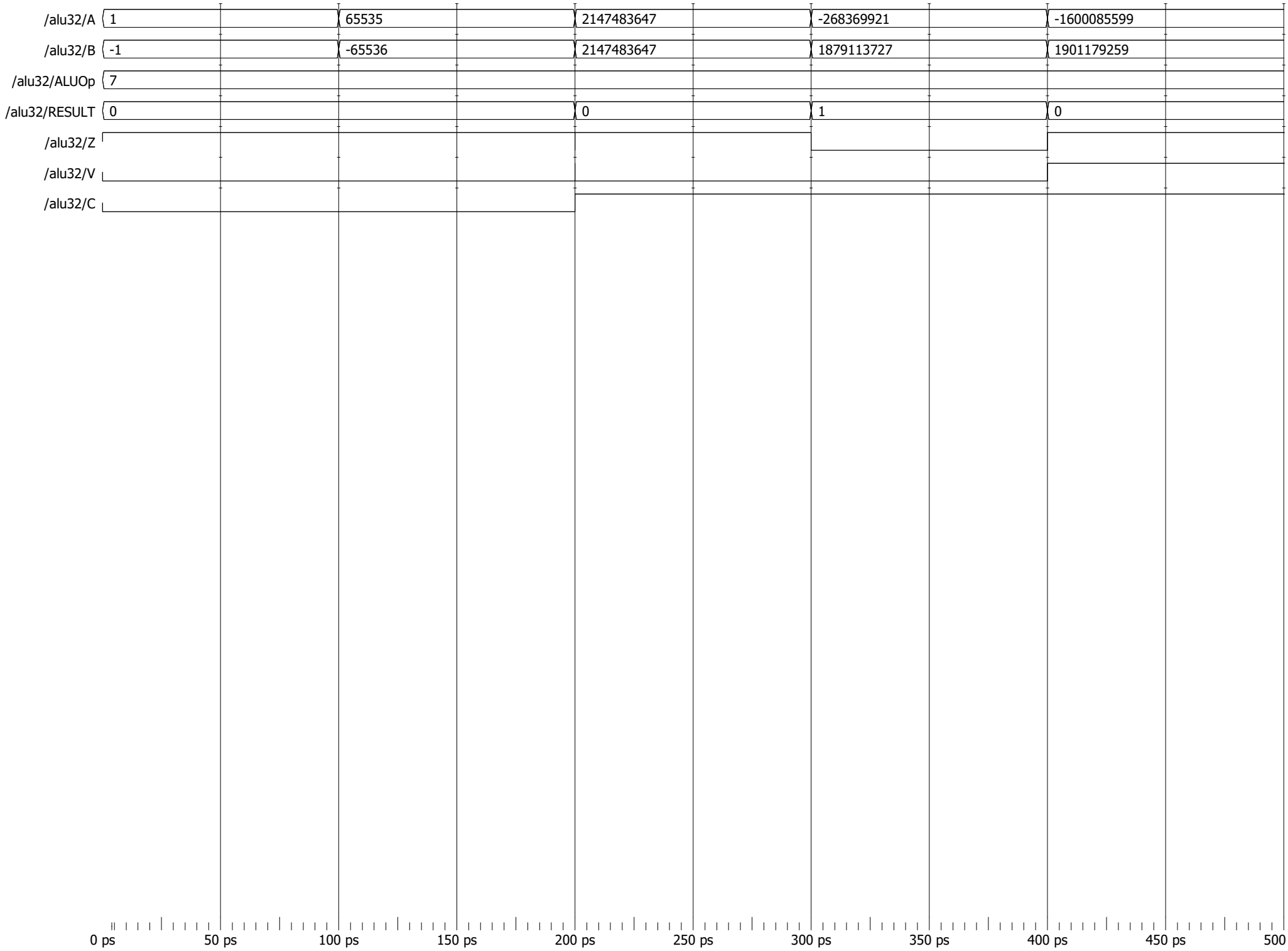

Entity:alu32 Architecture:big_alu Date: Sun Nov 01 13:31:24 CST 2020 Row: 1 Page: 1



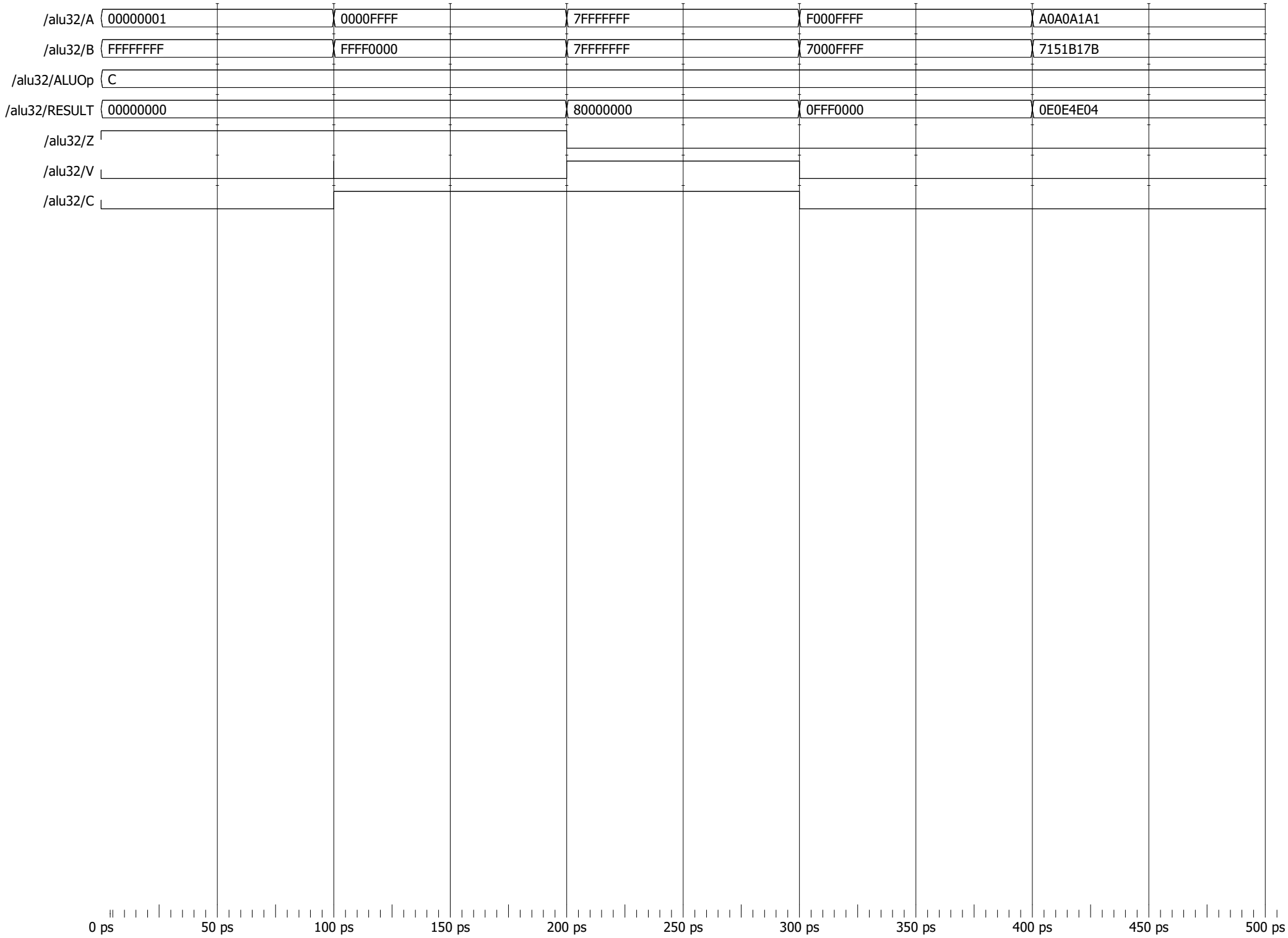

Entity:alu32 Architecture:big_alu Date: Sun Nov 01 13:34:29 CST 2020 Row: 1 Page: 1




Entity:alu32 Architecture:big_alu Date: Sun Nov 01 13:38:35 CST 2020 Row: 1 Page: 1




Entity:alu32 Architecture:big_alu Date: Sun Nov 01 13:40:21 CST 2020 Row: 1 Page: 1




Entity:alu32 Architecture:big_alu Date: Sun Nov 01 13:45:48 CST 2020 Row: 1 Page: 1