ECE 424
DUE: Oct 10 in class

Team Design #1: ALU Bitslice (50 pts)

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_____

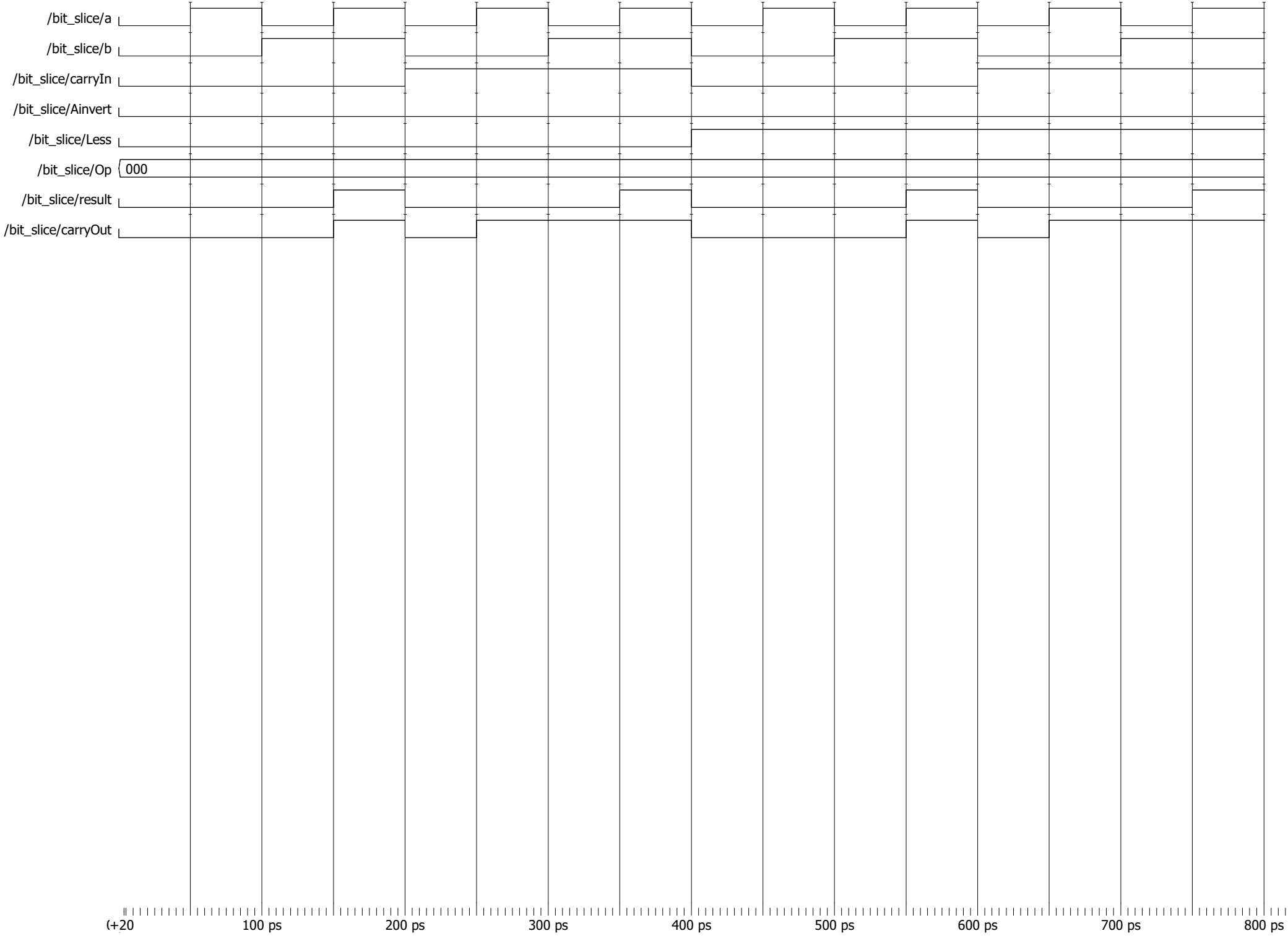
Honor Code: _____ I have neither given or received, nor have I tolerated others' use _____

_____ of unauthorized aid._____

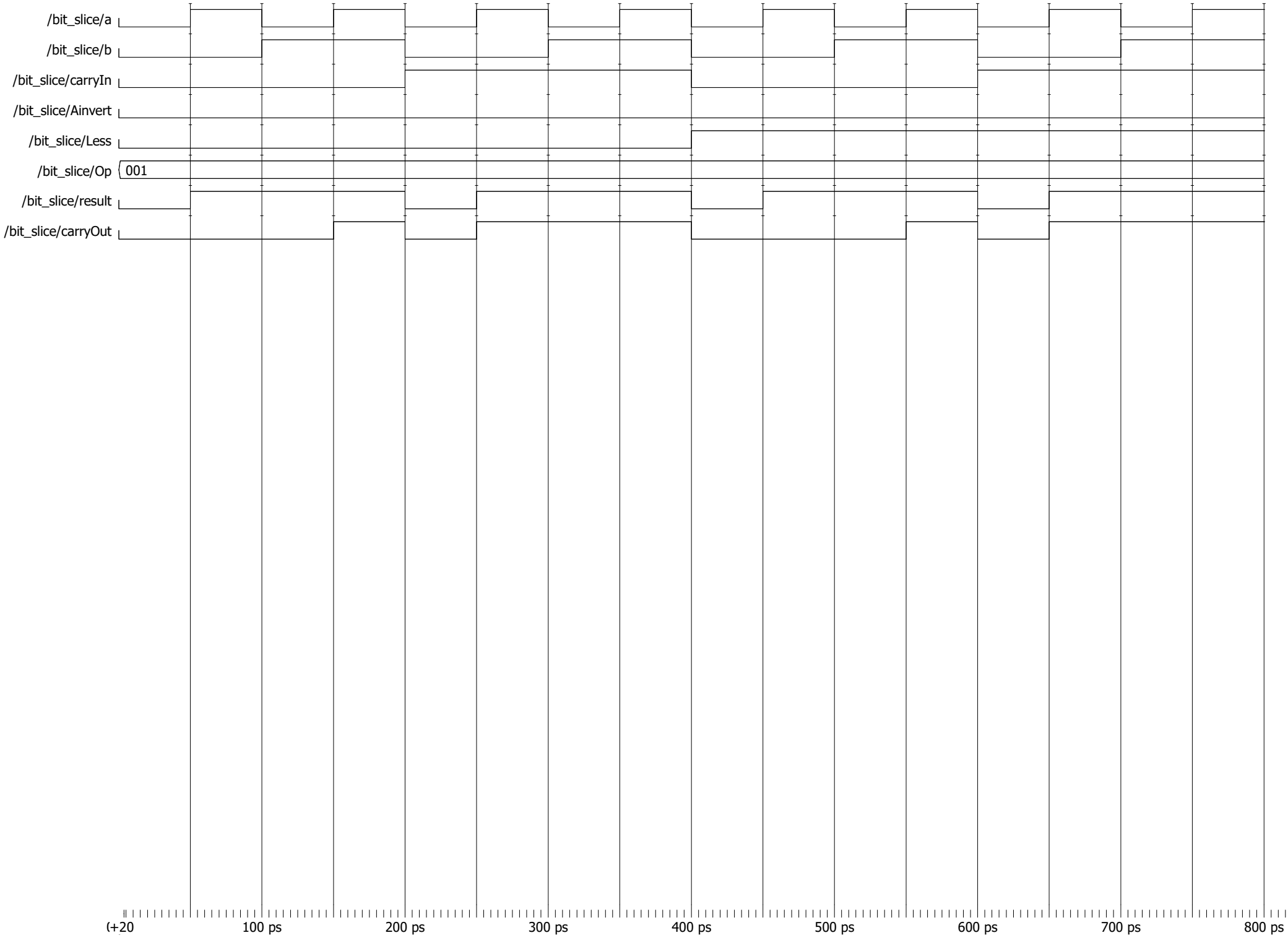
Name:_____Jon Bayert_____ Signature: __Jonathan Bayert____

Include:
0) This coverpage stapled on the front
1) Your VHDL code for part 1
2) Your waveform results for part 1: 5 pages, one for each function
3) Your VHDL code for part 2
4) Your waveform results for part 2: 5 pages, one for each function

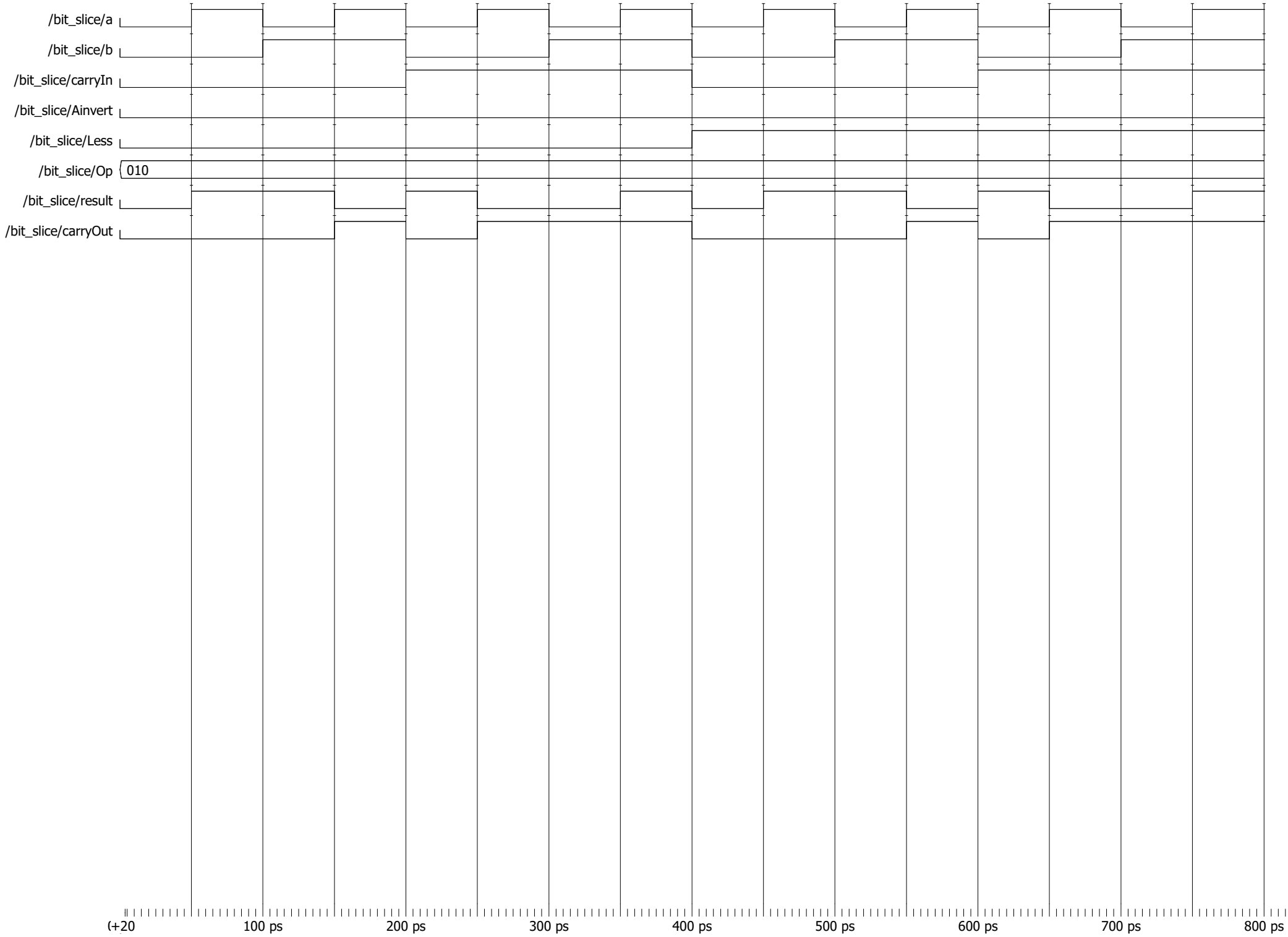
```vhdl
library ieee;
use ieee.std_logic_1164.all;

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;

```



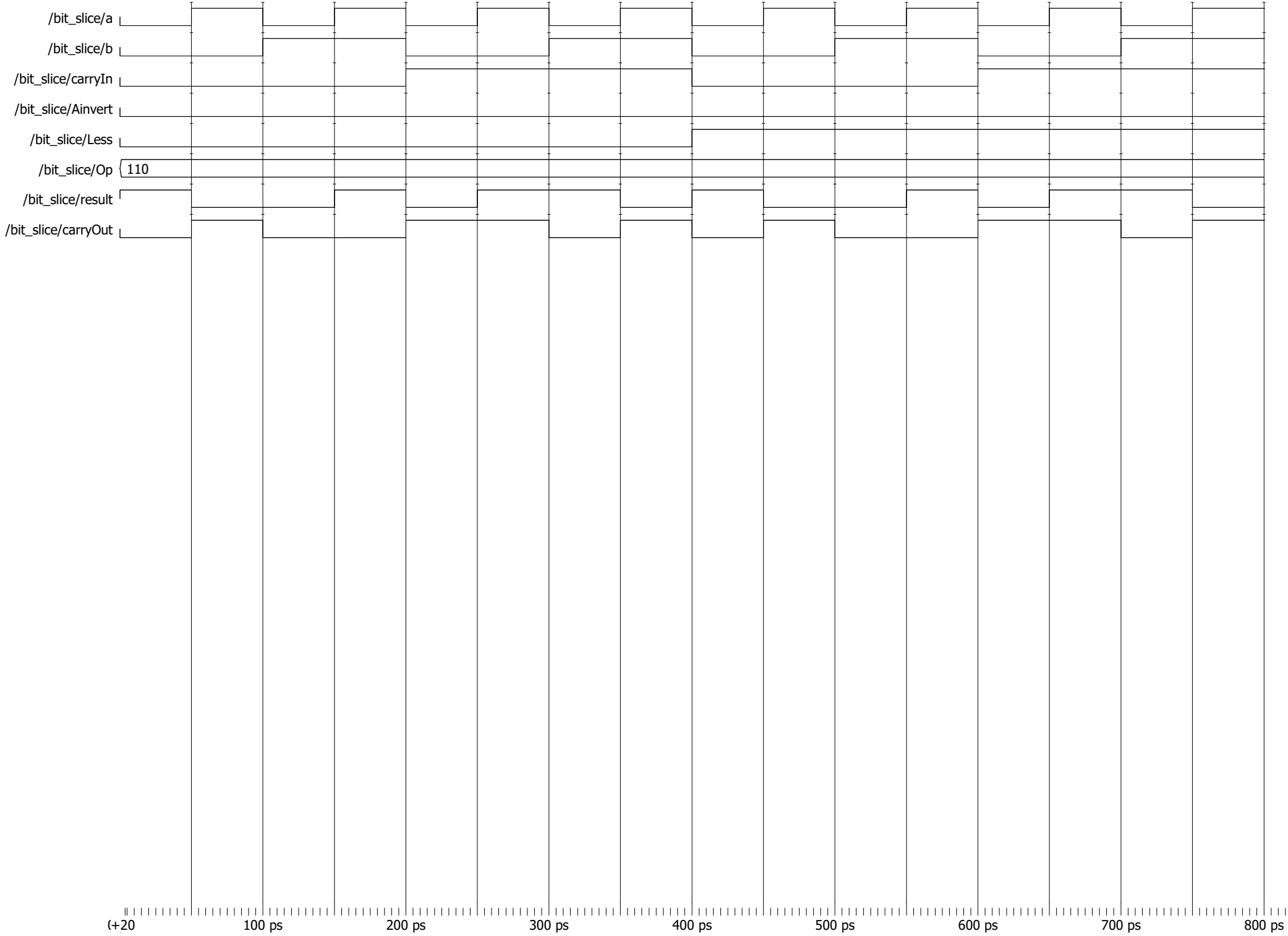
Entity:bit_slice Architecture:stdbitslice Date: Tue Oct 20 18:31:55 CDT 2020 Row: 1 Page: 1



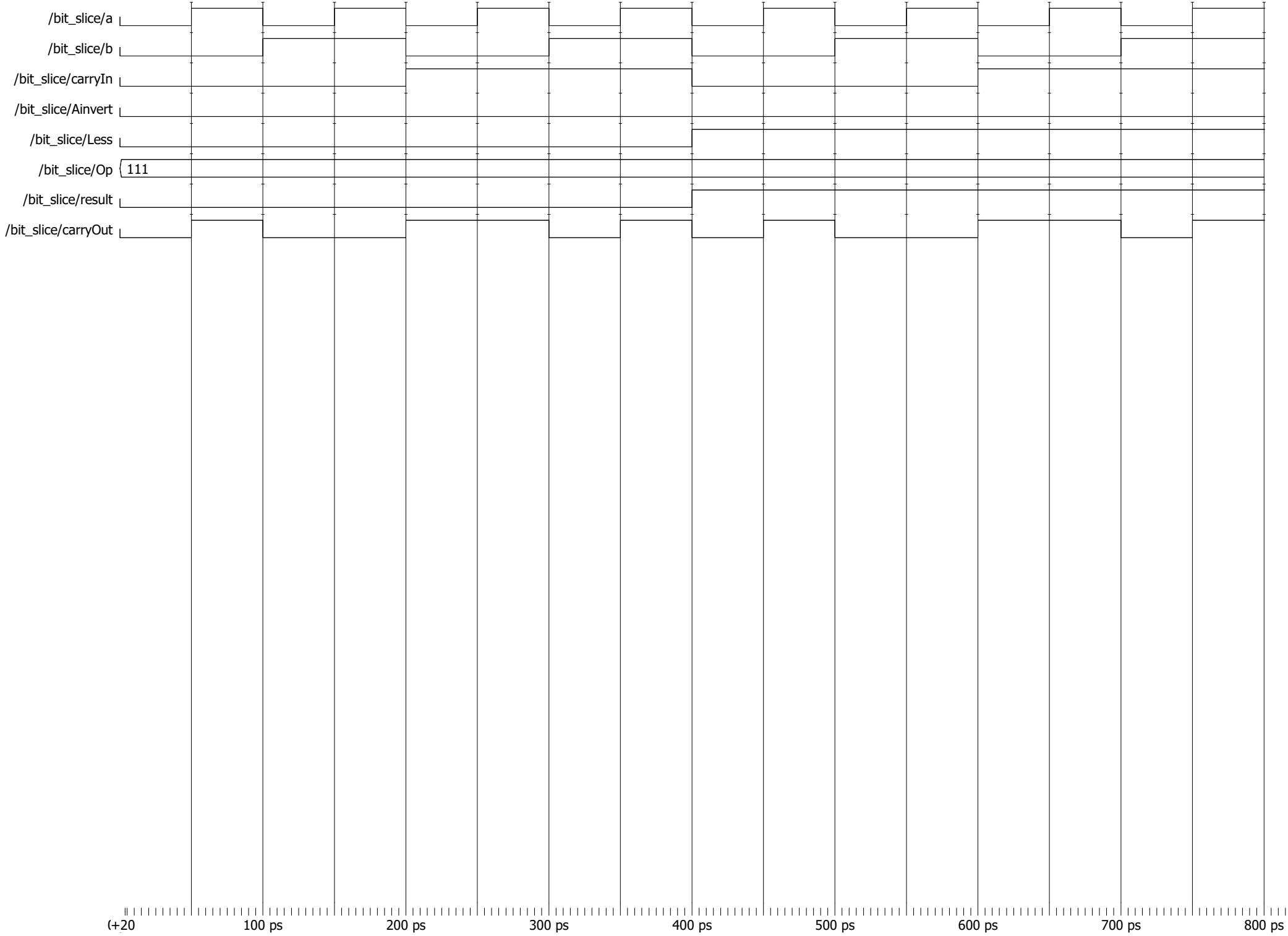
Entity:bit_slice Architecture:stdbitslice Date: Tue Oct 20 18:33:26 CDT 2020 Row: 1 Page: 1



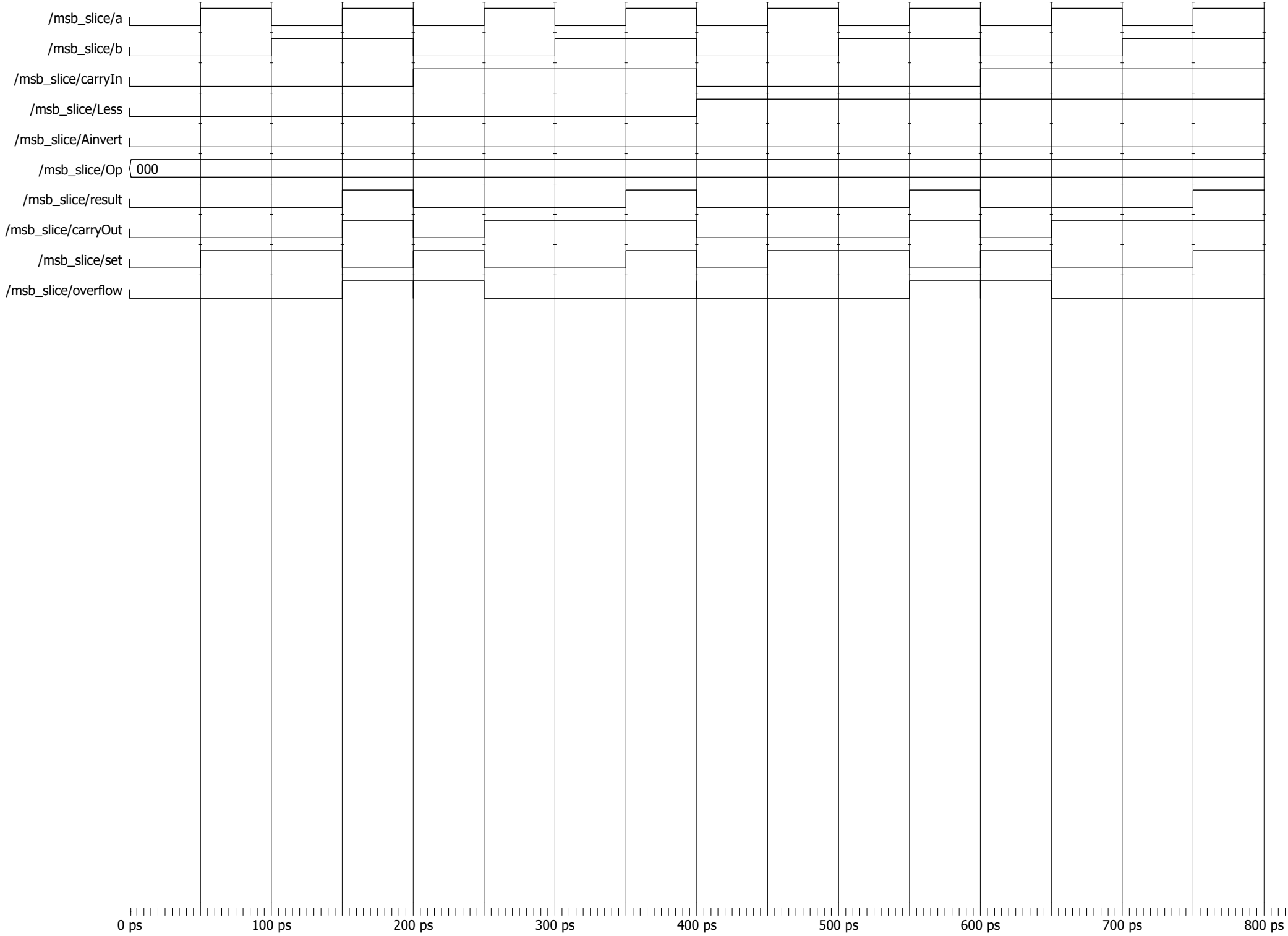
Entity:bit_slice Architecture:stdbitslice Date: Tue Oct 20 18:34:33 CDT 2020 Row: 1 Page: 1



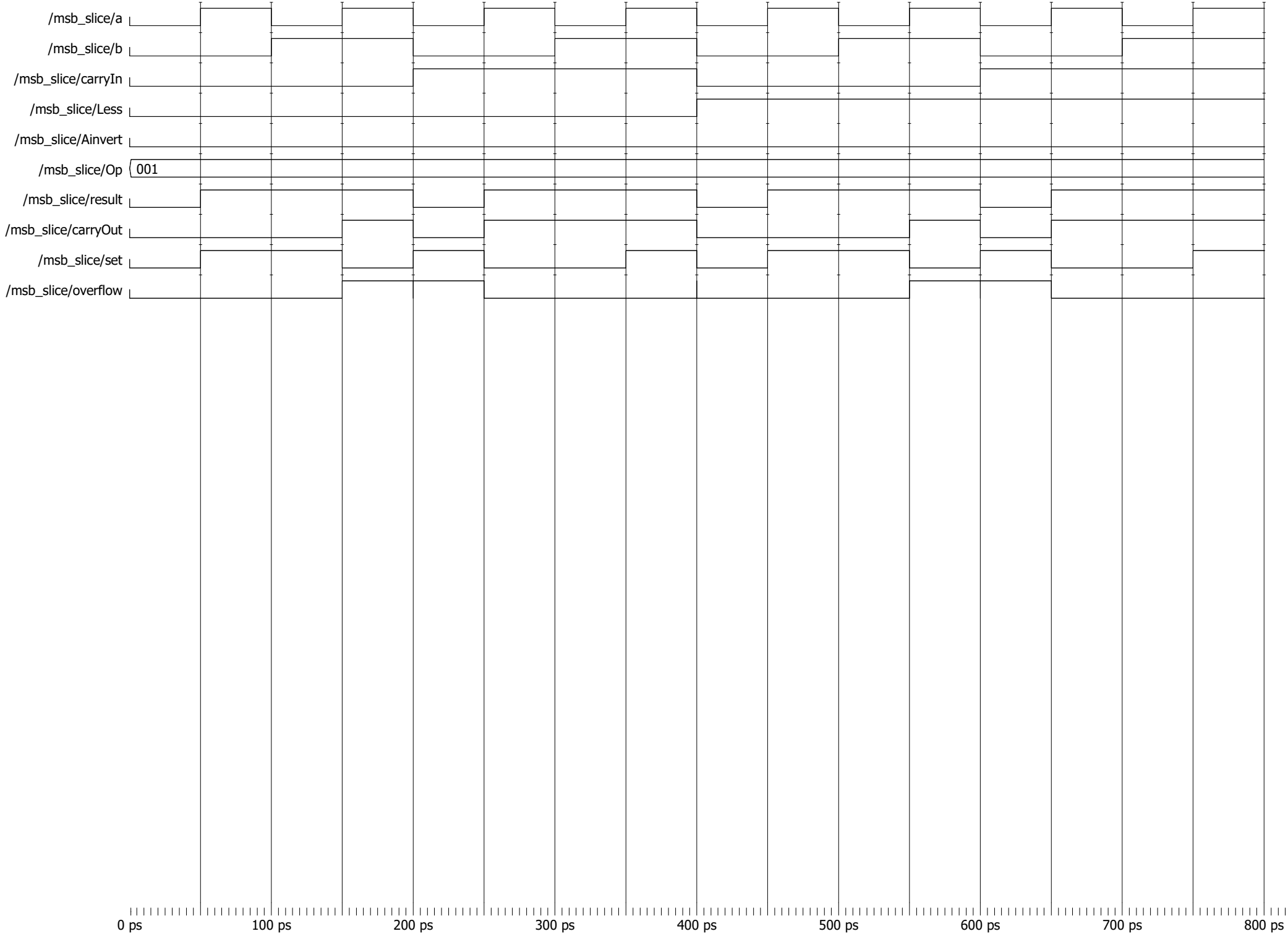
Entity:bit_slice Architecture:stdbitslice Date: Tue Oct 20 18:36:04 CDT 2020 Row: 1 Page: 1


/bit_slice/a
/bit_slice/b
/bit_slice/carryIn
/bit_slice/Ainvert
/bit_slice/Less
/bit_slice/Op
111
/bit_slice/result
/bit_slice/carryOut
(+20
100 ps
200 ps
300 ps
400 ps
500 ps
600 ps
700 ps
800 ps

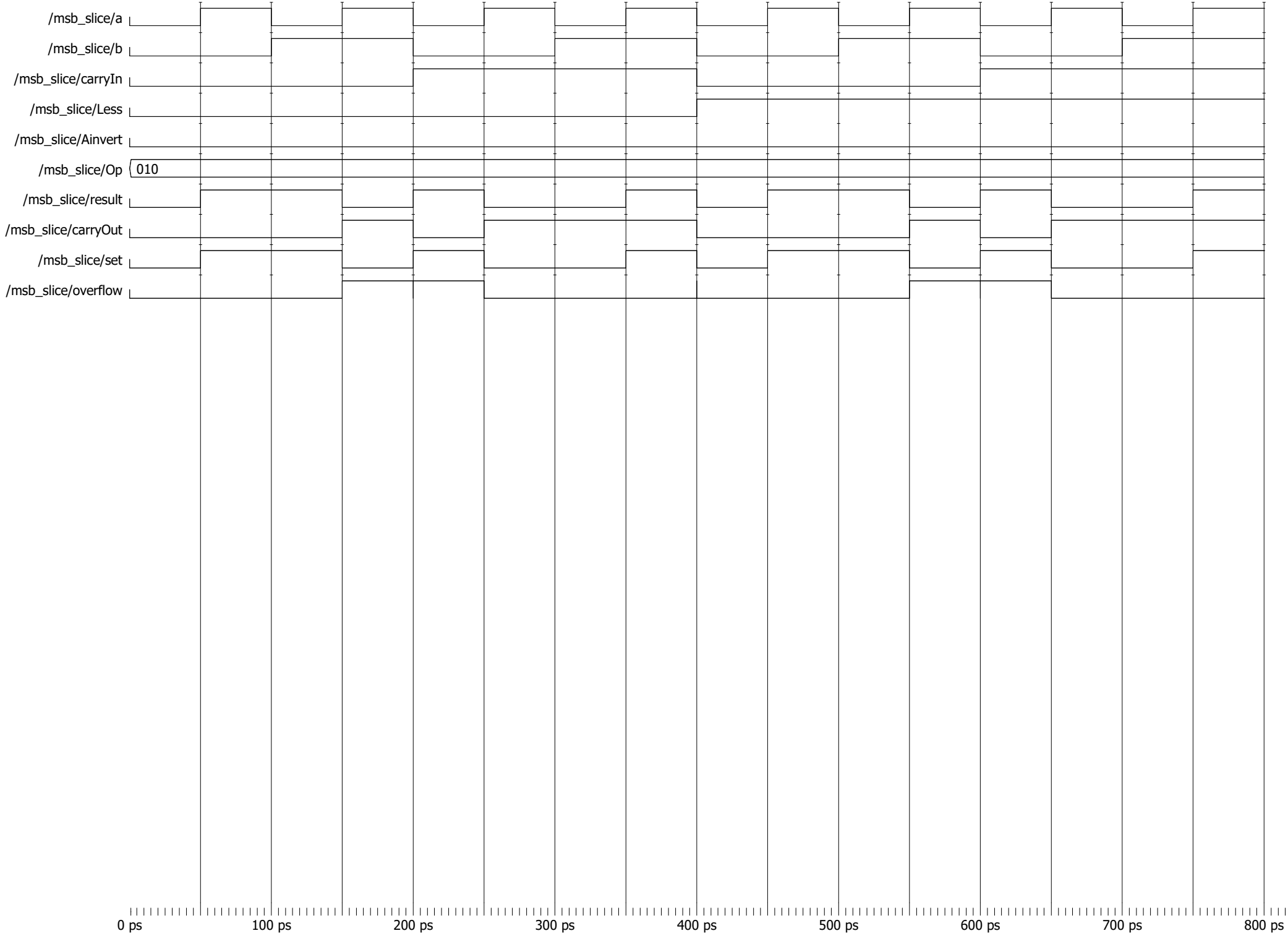
```vhdl
library ieee;
use ieee.std_logic_1164.all;

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;
```



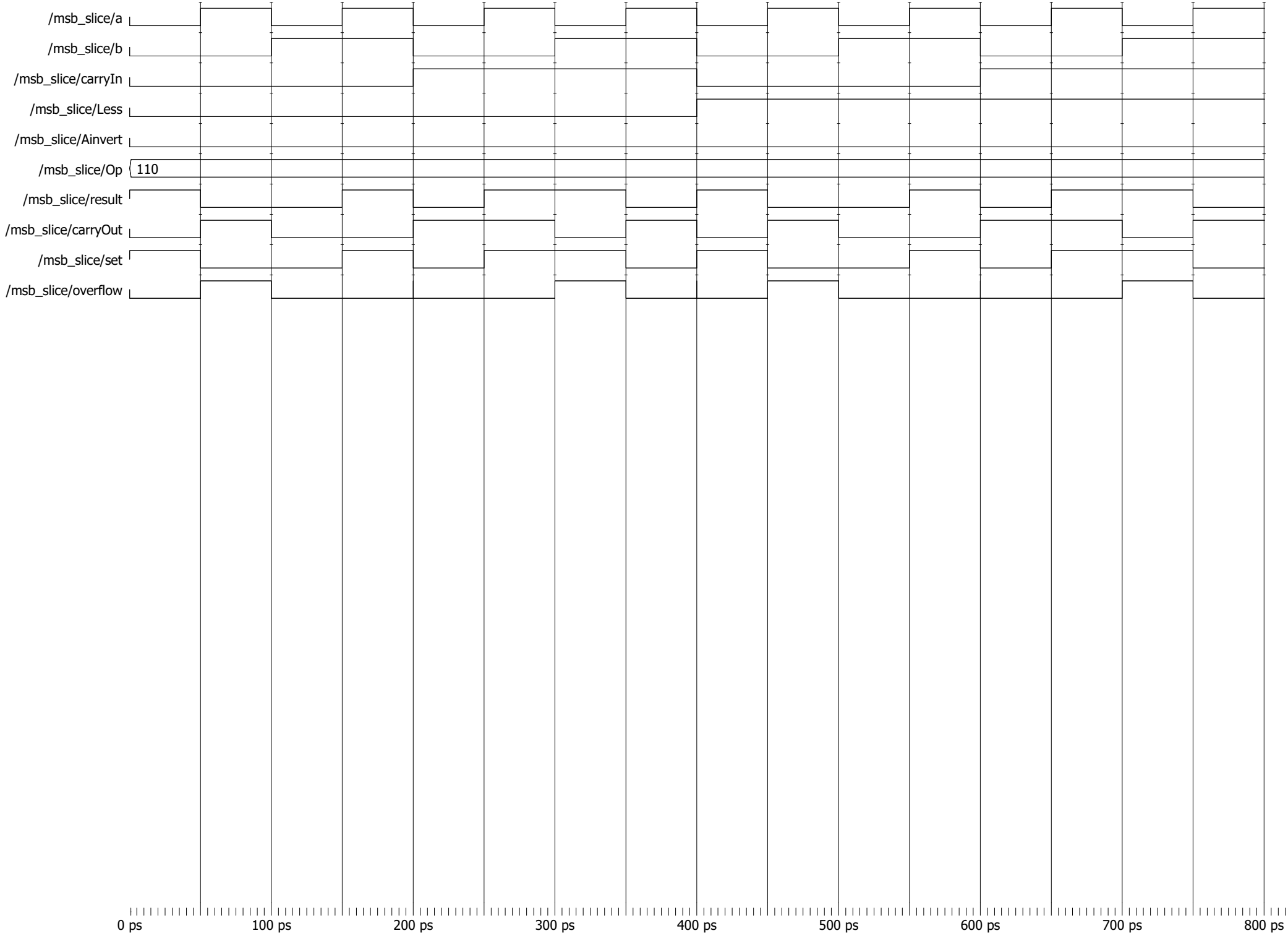
Entity:msb_slice Architecture:msbbitslice Date: Tue Oct 20 19:06:01 CDT 2020 Row: 1 Page: 1



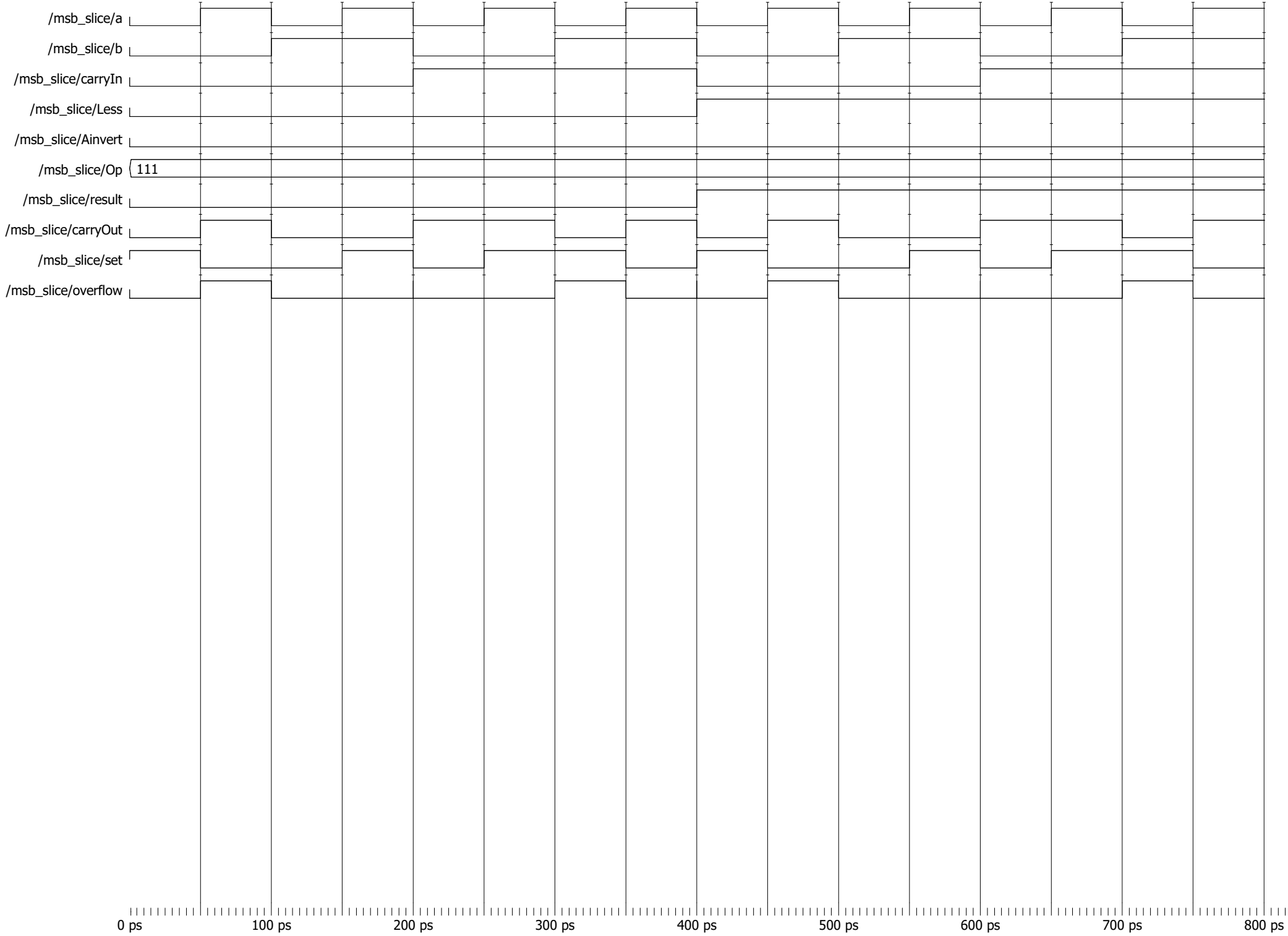
Entity:msb_slice Architecture:msbbitslice Date: Tue Oct 20 19:07:05 CDT 2020 Row: 1 Page: 1



Entity:msb_slice Architecture:msbbitslice Date: Tue Oct 20 19:08:39 CDT 2020 Row: 1 Page: 1



Entity:msb_slice Architecture:msbbitslice Date: Tue Oct 20 19:09:13 CDT 2020 Row: 1 Page: 1



Entity:msb_slice Architecture:msbbitslice Date: Tue Oct 20 19:09:50 CDT 2020 Row: 1 Page: 1