Jeffery You (jy9vm)

Lab 5 Report

VHDL Source:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY LargerFloat IS
   GENERIC (width : POSITIVE := 8); -- Number of floating point number bits
    PORT (A, B : IN std_logic_vector(width -1 DOWNTO 0);
    Larger, Smaller : OUT std_logic_vector(width - 1 DOWNTO 0));
END ENTITY LargerFloat;

--
ARCHITECTURE Behavior OF LargerFloat IS
BEGIN
  PROCESS(A,B)
  BEGIN
    for i in (width-1) DOWNTO 0 loop
      if A(i)>B(i) then
        Smaller <= B;
        Larger <= A;
        EXIT;
      elsif A(i)<B(i) then
        Smaller <= A;
        Larger <= B;
        EXIT;
      elsif i=0 and A(0)=B(0) then
        Smaller <=A;
        Larger <=B;
        EXIT;
      else
        next;
      end if;
    END LOOP;
  END PROCESS;
END ARCHITECTURE Behavior;
```

VHDL code of the finding the larger of two floating point numbers

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;


ENTITY AddSubtract IS
   GENERIC (width : POSITIVE := 8); -- Number of fixed point number bits
    PORT (Left, Right : IN std_logic_vector(width - 1 DOWNTO 0); -- Inputs
    Sum : OUT std_logic_vector(width DOWNTO 0); -- Output
    AddSub : IN std_logic); -- Control to choose Add or Subtract
END ENTITY AddSubtract;

--
ARCHITECTURE Behavior OF AddSubtract IS
BEGIN
  PROCESS(Left, Right, AddSub)
  Begin
    if AddSub='1' then
      Sum<=std_logic_vector(unsigned('0'& Left)-(unsigned('0' & Right)));
    else
      Sum<=std_logic_vector(unsigned('0' & Right)+(unsigned('0' & Left)));
    end if;
  End Process;
END ARCHITECTURE Behavior;


```

VHDL code of the fixed point adder and subtractor

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

ENTITY BarrelShifter IS
  GENERIC (width : POSITIVE := 8; -- Number of input and output bits
    ControlBits : POSITIVE := 4);
  PORT (A : IN std_logic_vector(width -1 DOWNTO 0);
    Y : OUT std_logic_vector(width -1 DOWNTO 0);
    RightShifts : IN std_logic_vector(ControlBits -1 DOWNTO 0));
END ENTITY BarrelShifter;

--
ARCHITECTURE Behavior OF BarrelShifter IS
BEGIN
  PROCESS(A, RightShifts)
  BEGIN
    if (RightShifts(ControlBits-1)='0') then
      Y<=A srl to_integer(unsigned(RightShifts));
    else
      Y<=A sll (to_integer(unsigned(not RightShifts))+1);
    end if;
  END PROCESS;
END ARCHITECTURE Behavior;
```

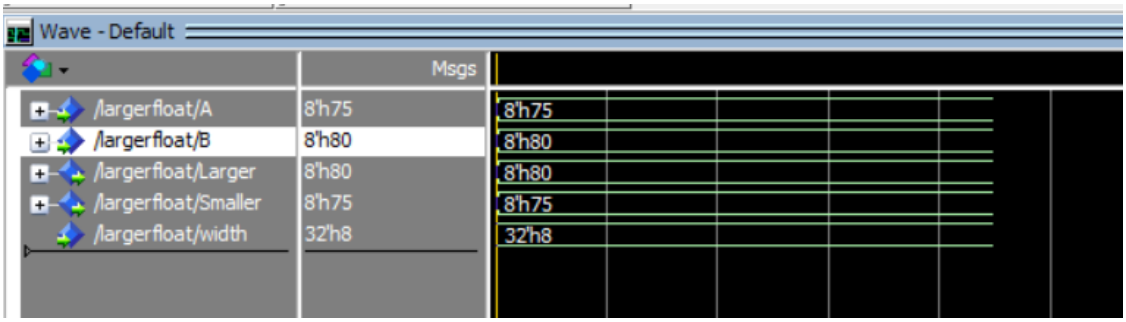
VHDL code of the variable-bit shifter

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.numeric_std.all;

ENTITY NormalShiftCount IS
   GENERIC (width : POSITIVE := 8; -- Number of mantissa bits
    Shifts : POSITIVE := 3); -- Number of bits in shift count
   PORT (A : IN std_logic_vector(width - 1 DOWNTO 0);
    S : OUT std_logic_vector(shifts -1 DOWNTO 0);
    NoOnes : OUT std_logic); -- NoOnes is asserted if the mantissa is all zeros
END ENTITY NormalShiftCount;

--
ARCHITECTURE Behavior OF NormalShiftCount IS
BEGIN
  Process(A)
    begin
    for i in (width-1) DOWNTO 0 loop
      if A(i)='1' then
        S<= std_logic_vector(to_signed((i -(width-2)), S'length));
        NoOnes<='0';
        EXIT;
      end if;
      if (A = (A'range => '0')) then
        NoOnes<='1';
        S<=(others=>'0');
        EXIT;
      end if;
    END loop;
  end process;
END ARCHITECTURE Behavior;
```

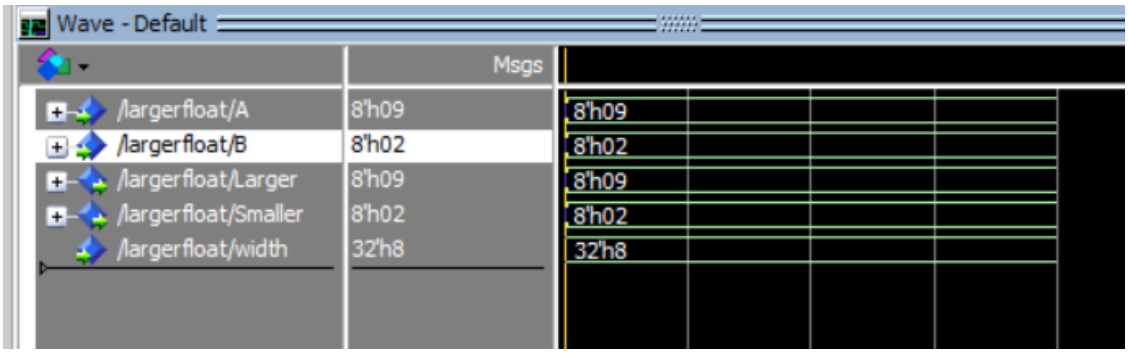
VHDL code of finding the most significant ‘1’

Test and Results:

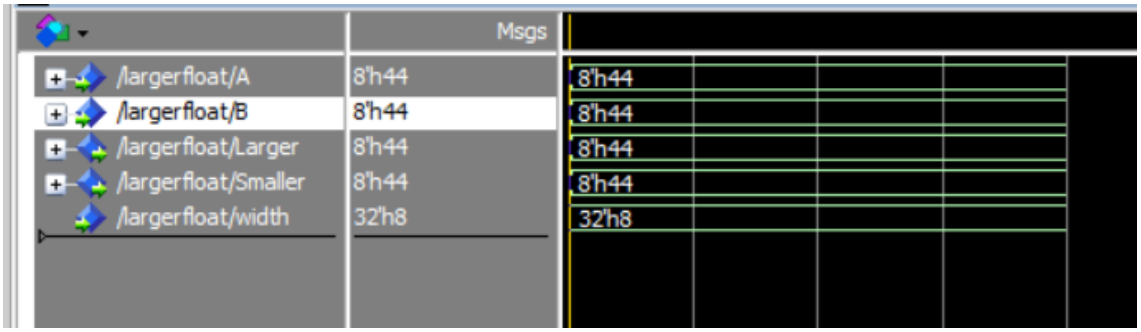
Finding the Larger of Two Floating Point Numbers



The variable A is set to 75 and the variable B is set to 80. The simulation determined that the larger value is 80 and the smaller value is 75, as seen in the figure above.

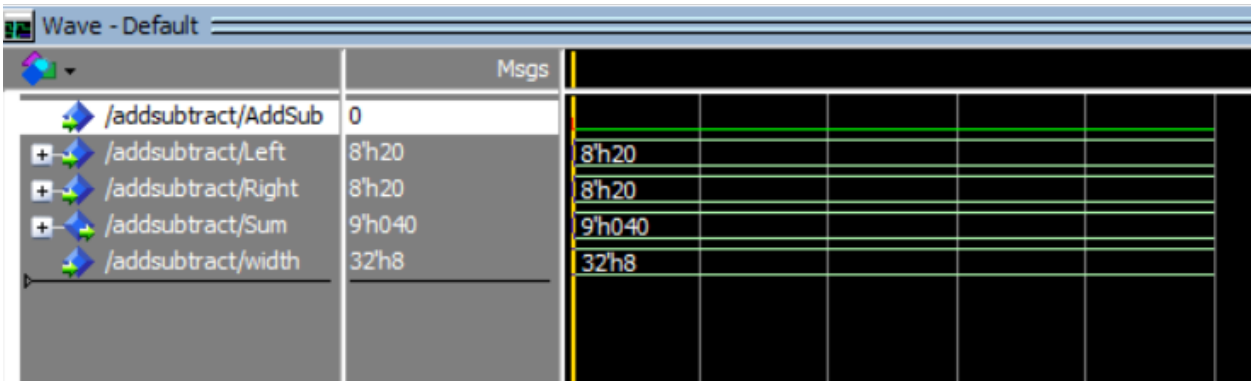


The variable A is set to 9 and the variable B is set to 2. The simulation determined that the larger value is 9 and the smaller value is 2, as seen in the figure above.

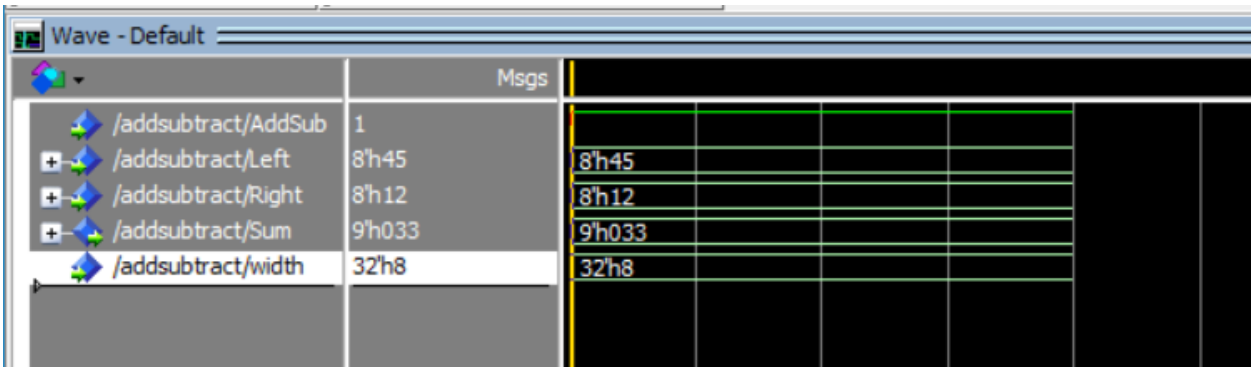


Both A and B were set to 44. The simulation determined that the larger and smaller value were both 44.

A Fixed Point Adder/Subtractor

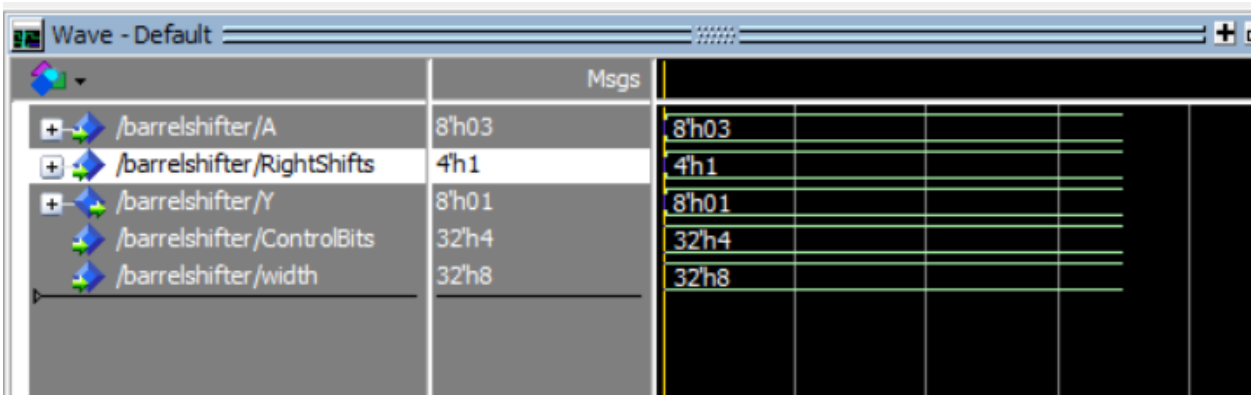


The AddSum variable is asserted '0' which means to add the left input to the right input. With the left input set to 20 and the right input set to 20, the sum is 40, which matches the simulation.

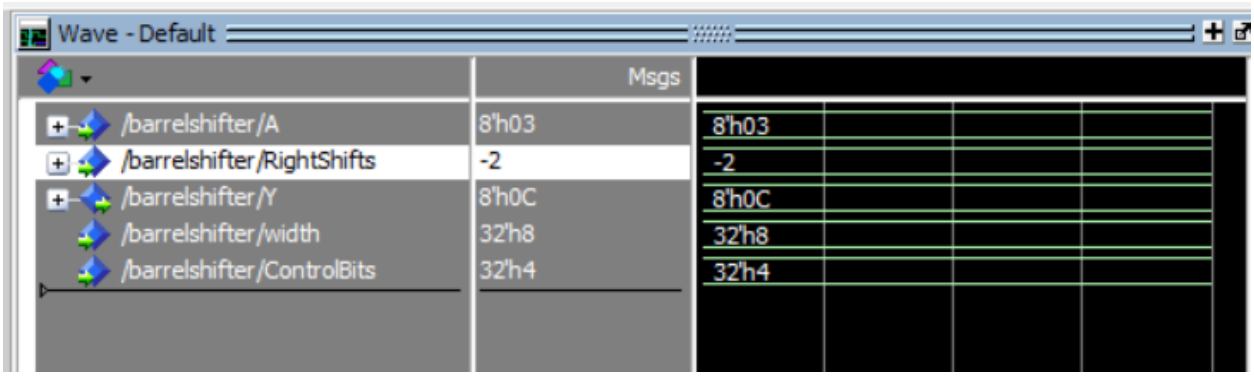


The AddSum variable is asserted '1' which means to subtract the right input from the left input. With the left input set to 45 and the right input set to 33, the subtraction is 33, which matches the value assigned to the sum variable.

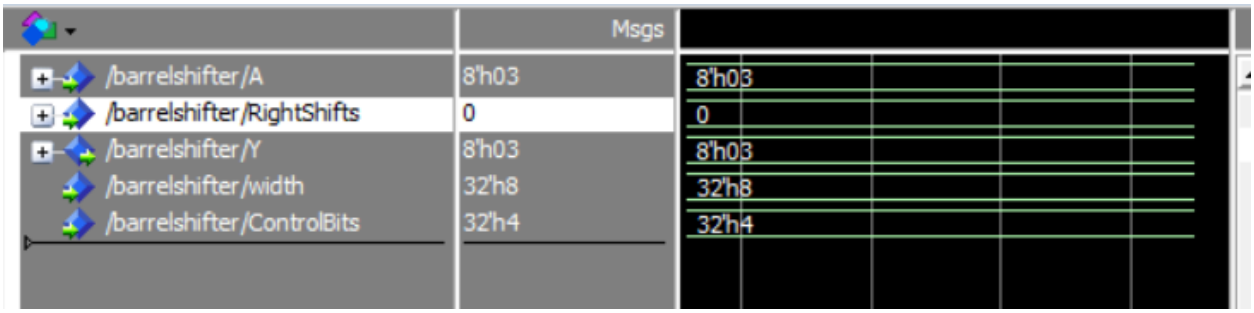
A Variable Bit-Shifter



The input variable A was set to 3. The RightShifts variable was set to 1. This moves A one bit to the right, which results in Y being equal to 1.

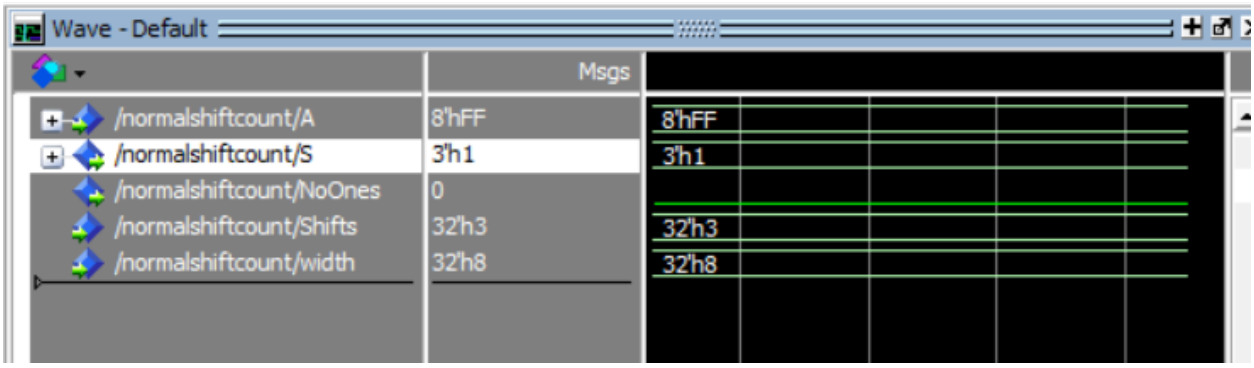


The input variable A was set to 3. The RightShifts variable was set to -2. This moves A 2 bits to the left, which results in Y being equal to 12. 12 is C in hex and this is correctly displayed in the figure above.

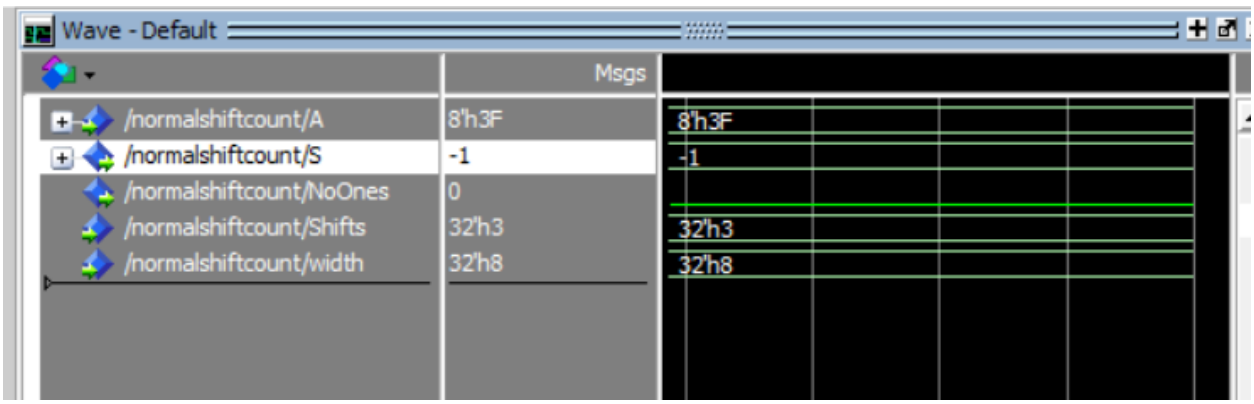


The input variable A was set to 3. Since the RightShifts variable was set to 0, A does not move any bits. This results in Y being equal to 3.

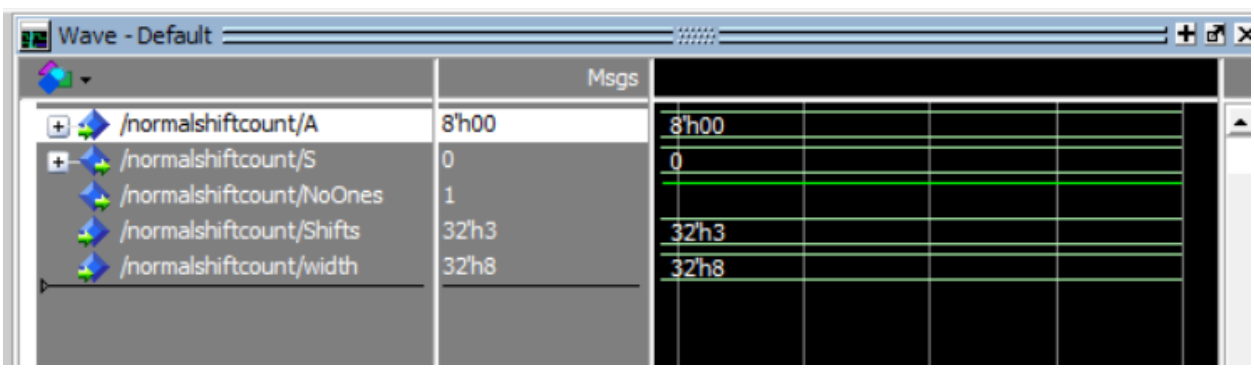
Finding the Most Significant '1'



The input variable A is set to FF, which means that the most signficant bit is the 8th one. As a result, S outputs 1, indicating how much should be shifted to the right.

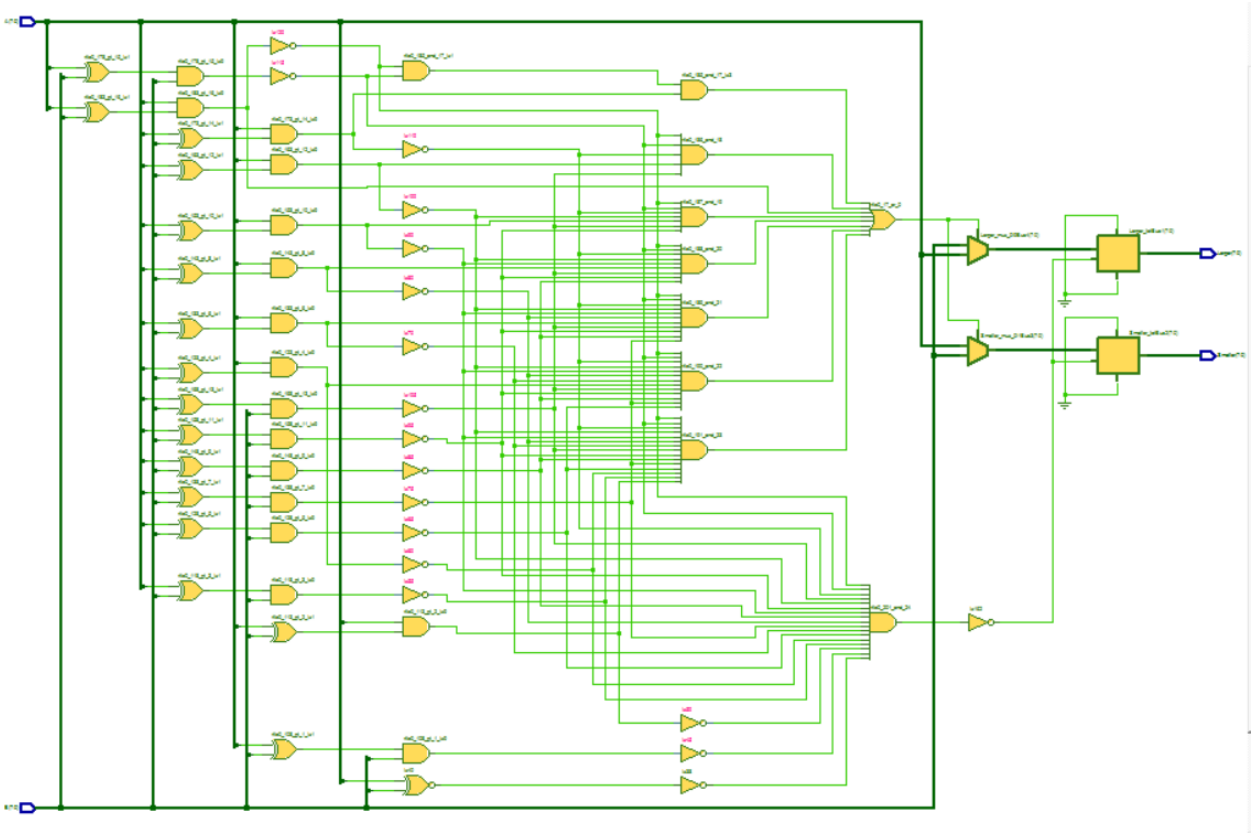


The input variable A is set to 3F, which means that the most significant bit is the 6th one. As a result, S outputs -1, as the amount that needs to be shifted to the right.

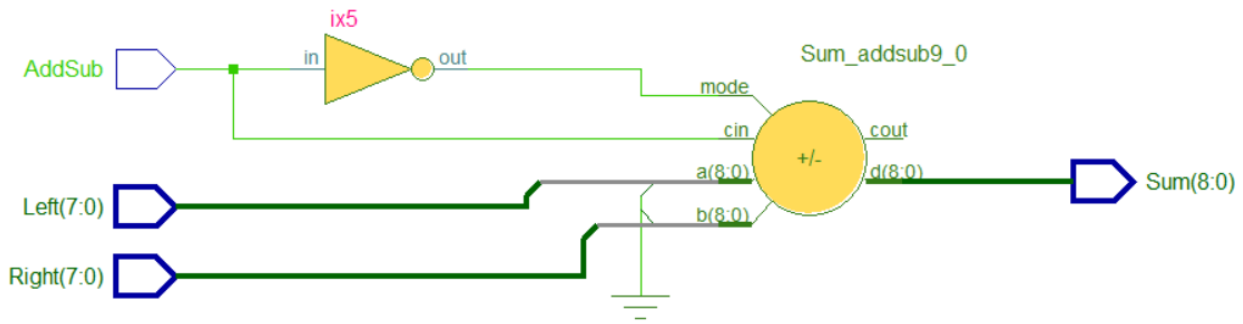


The input variable is set to 0, which means that there are no '1's. As a result, S outputs 0, and the NoOnes variable asserts '1'.

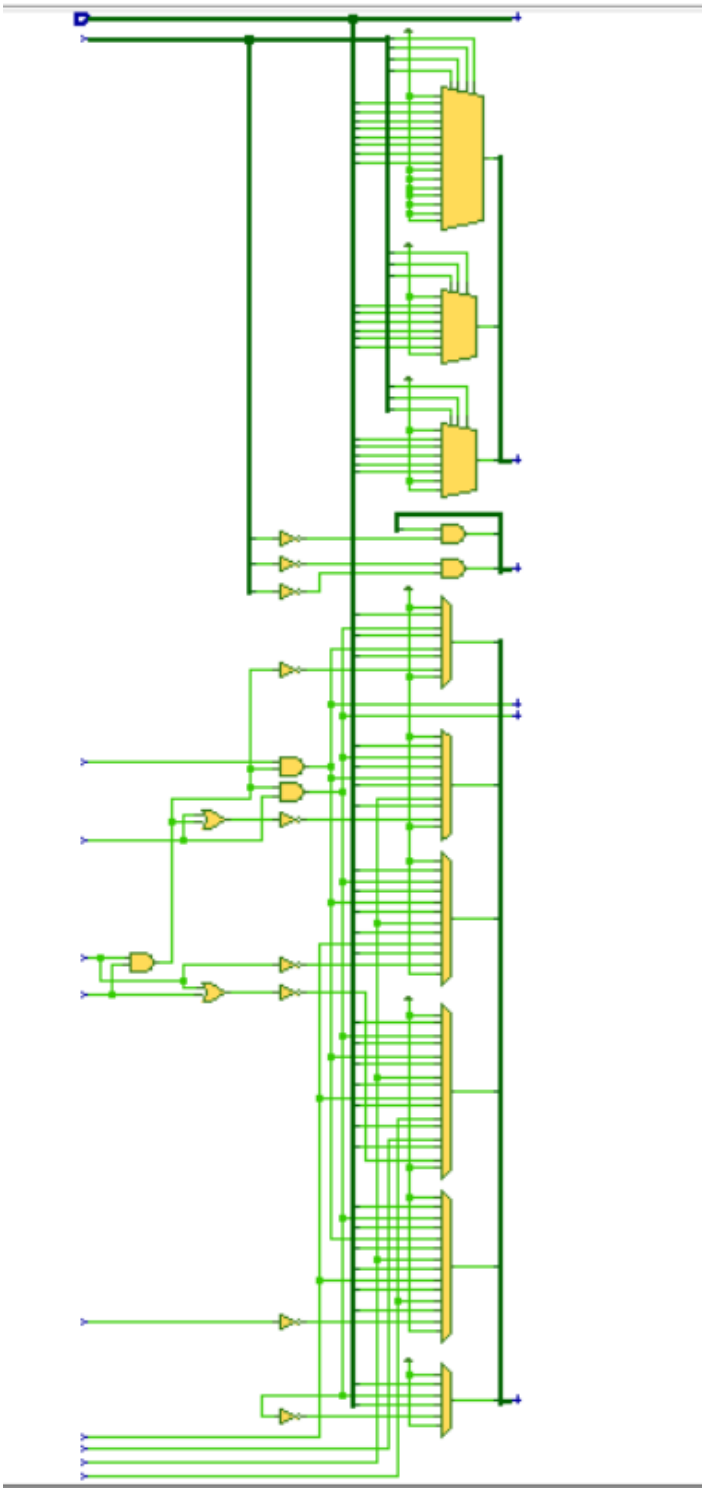
Schematics:



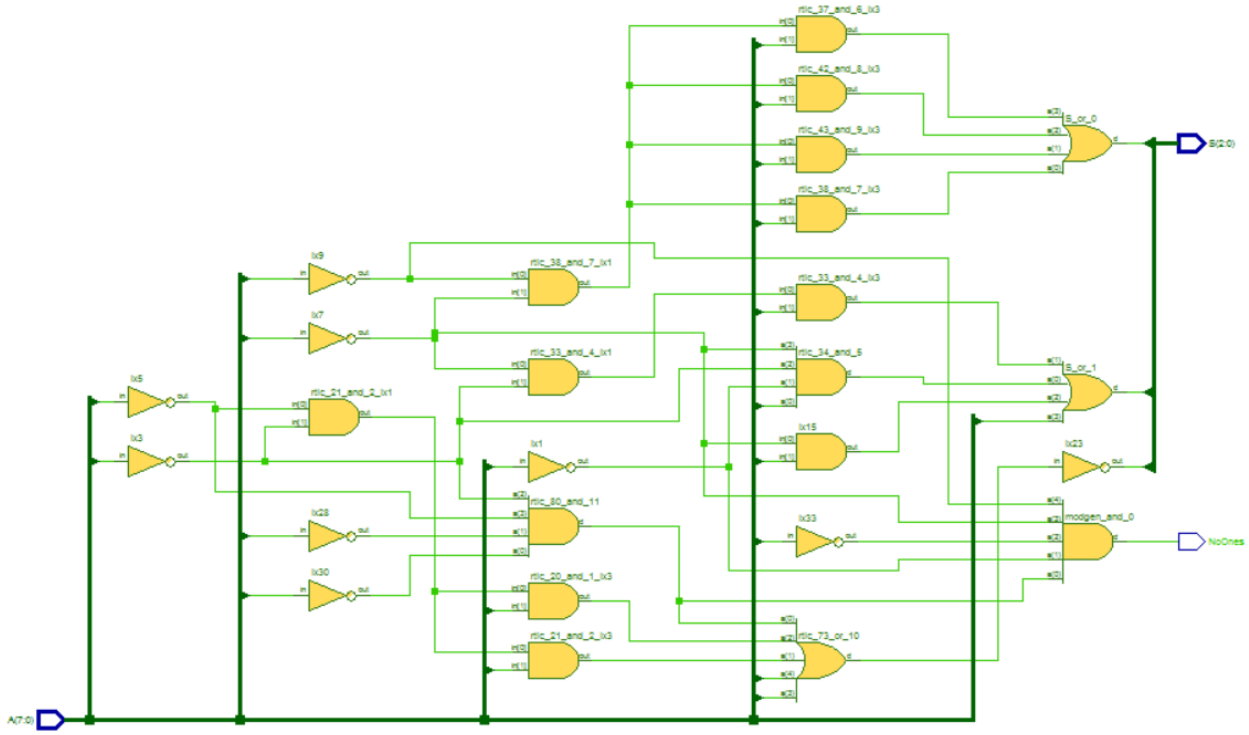
RTL view of finding the larger of two floating point numbers



RTL view of the fixed-point adder/subtractor



RTL view of the variable-bit shifter



RTL view of finding the most significant ‘1’