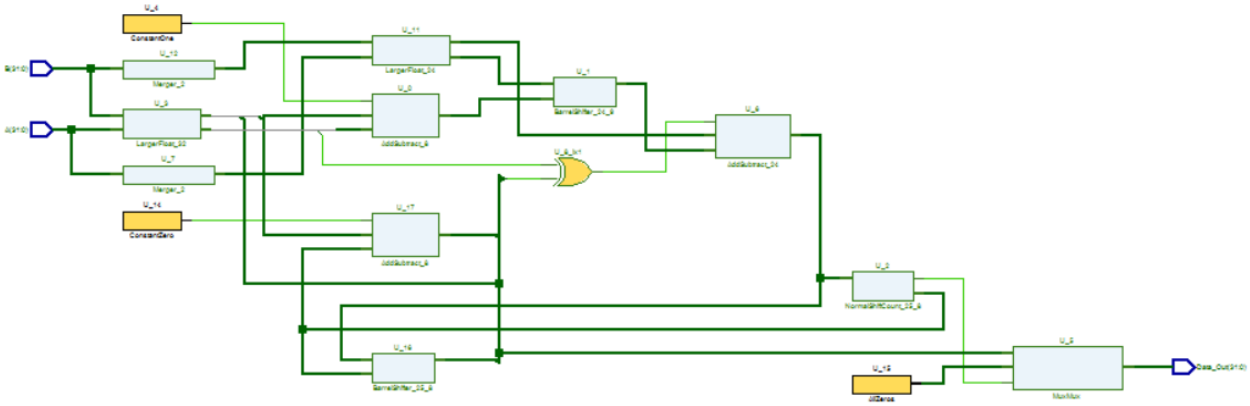
Jeffery You (jy9vm)

Combinational Floating Point Adder

Schematic:



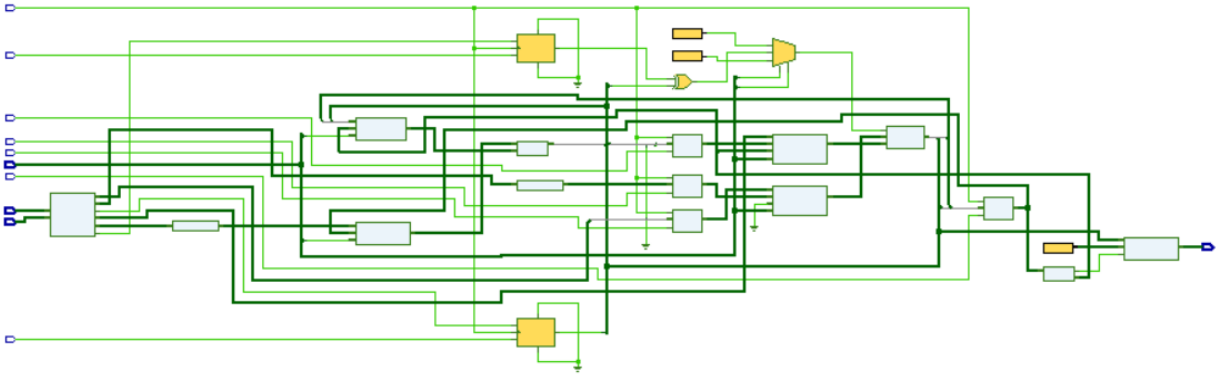
Resource Requirements:

```
**********************************************************************
Device Utilization for 10M08DAF256C7G
**********************************************************************
Resource                              Used    Avail   Utilization
----------------------------------------------------------------------
IOs                                   96      178      53.93%
LUTs                                  1359    8064     16.85%
Registers                             0       8064      0.00%
Memory Bits                           0       387072    0.00%
DSP block 9-bit elems                 0       48        0.00%
----------------------------------------------------------------------
```

There are no registers utilization for the FPGA requirement report for the combinational datapath.

Sequential Floating Point Data Path

Schematic:



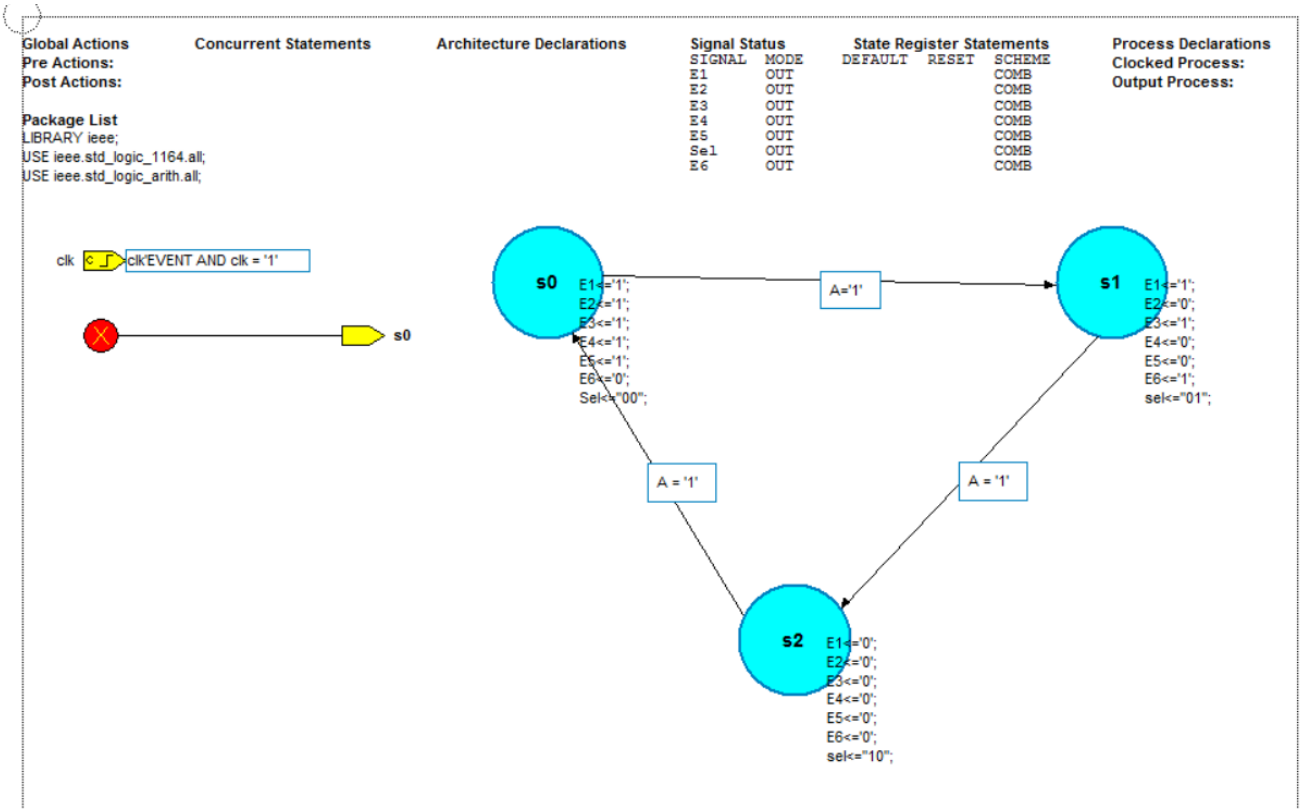
There are a lot of more wires associated with this design. It looks like the RTL made some of the MUXs boxes for some reason.

Resource Requirements:

```
******************************************************************
Device Utilization for 10M08DAF256C7G
******************************************************************
Resource                            Used    Avail   Utilization
------------------------------------------------------------------
IOs                                 105     178      58.99%
LUTs                                999     8064     12.39%
Registers                           83      8064      1.03%
Memory Bits                         0       387072    0.00%
DSP block 9-bit elems               0       48        0.00%
------------------------------------------------------------------
```

There are now registers utilization now for the FPGA device.

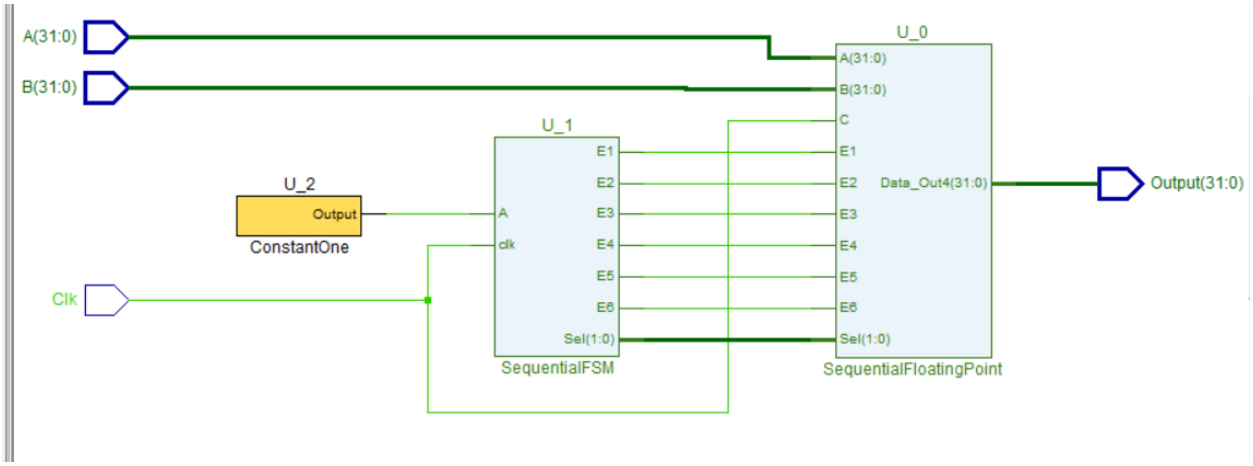
Sequential Control Unit



The FSM machine controls the enables on the registers to determine when the store and when to not store; these are my E's. The Sel variable is the selector for the MUXs. The Sel variable changes depending on the clock cycle. It goes to "00" to "01" to "10" to indicate the cycle.

Sequential Floating Point Adder (Data Path + Control Unit)

Schematic:



This schematic connects the FSM with the Sequential Datapath.

Resource Requirements:

```
**********************************************************************
Device Utilization for 10M08DAF256C7G
**********************************************************************
Resource                              Used    Avail   Utilization
----------------------------------------------------------------------
IOs                                   97      178      54.49%
LUTs                                  1000    8064     12.40%
Registers                             85      8064      1.05%
Memory Bits                           0       387072    0.00%
DSP block 9-bit elems                 0       48        0.00%
----------------------------------------------------------------------
```

Registers are now being used for the FPGA utilization.

Sequential TestBench

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

ENTITY SequentialTestBench IS
END ENTITY SequentialTestBench;

--
ARCHITECTURE Behavior OF SequentialTestBench IS
  SIGNAL clock : std_logic;
  Signal A, B, Output: std_logic_vector(31 DOWNTO 0);
BEGIN
  DUT: ENTITY work.SequentialProcessFlow(Struct)
    PORT MAP(A=>A, B=>B, Output=>Output, clk=>clock);
  PROCESS
    BEGIN
      clock <= '1';
      WAIT FOR 50 ns;
      clock <= '0';
      WAIT FOR 50 ns;
    END PROCESS;
  PROCESS
    BEGIN
      A<="01000000101000000000000000000000";
      B<="01000000111000000000000000000000";
      WAIT FOR 500 ns; --Some kind of slew start affected the initial result
      WAIT FOR 250 ns; -- Check the value in-between the 3rd state
      assert (Output = "01000000101000000000000000000000")
        report "Test 1 Failure";
      WAIT FOR 50 ns;

      A<="01000000100100000000000000000000";
      B<="01000000100010000000000000000000";
      WAIT FOR 300 ns;
      WAIT FOR 250 ns;
      assert (Output = "01000001100110000000000000000000")
        report "Test 2 Failure";
      WAIT FOR 50 ns;

      A<="01000001111110000000000000000000";
      B<="11000001011100000000000000000000";
      WAIT FOR 300 ns;
      WAIT FOR 250 ns;
      assert (Output = "01000001100000000000000000000000")
       report "Test 3 Failure";
      WAIT FOR 50 ns;

      A<="11000001111001011001100110011010";
      B<="01000001010101011001100110011010";
      WAIT FOR 250 ns;
      assert (Output = "11000001011101011001100110011010")
        report "Test 4 Failure";
      WAIT FOR 50 ns;

      A<="01000001001100000000000000000000";
      B<="00000000000000000000000000000000";
      WAIT FOR 250 ns;
      assert (Output = "01000001001100000000000000000000")
        report "Test 5 Failure";
      WAIT FOR 50 ns;
      WAIT;
    END PROCESS;
END ARCHITECTURE Behavior;
```

Testbench worked out well. I made the program report a test failure if there was a problem. I tested 5 different cases. Two different cases for adding two positive floating numbers, two different cases for adding two negative floating point numbers, one adding a negative and positive floating point numbers, and finally one adding a positive floating point number with a 0. I extend WAIT to 500 initially due to some clock skew delay for some reason that causes the first input entry not to register immediately. Within the cases, I extended to 6 clock cycles just in case the new immediate input entries does not register for the 1st 3 clock cycles. But the correct value is definitely given at the end of the 2nd 3 clock cycles. Also, I asserted within the middle of 3rd cycle and not towards the falling edge of the 3rd clock cycle to avoid any output changes when the input changes.