Floating Point Calculator

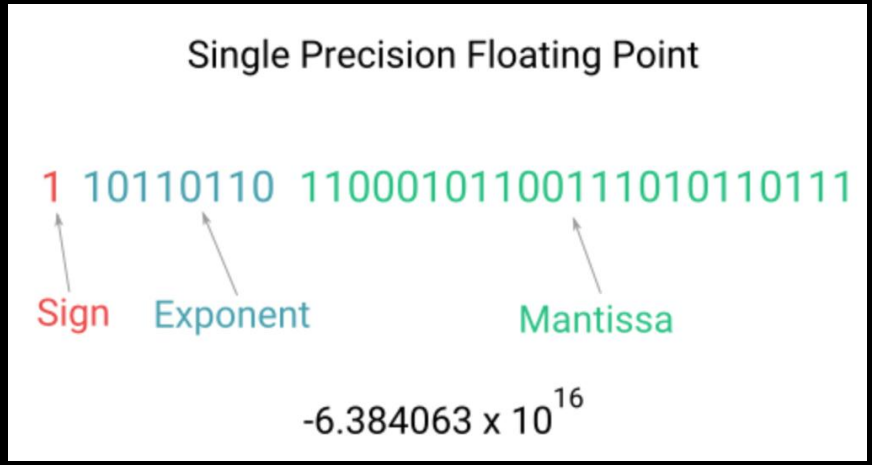
ECE 5736 - Professor Llamocca

Dylan Powell
Brandon Troppens

Objective

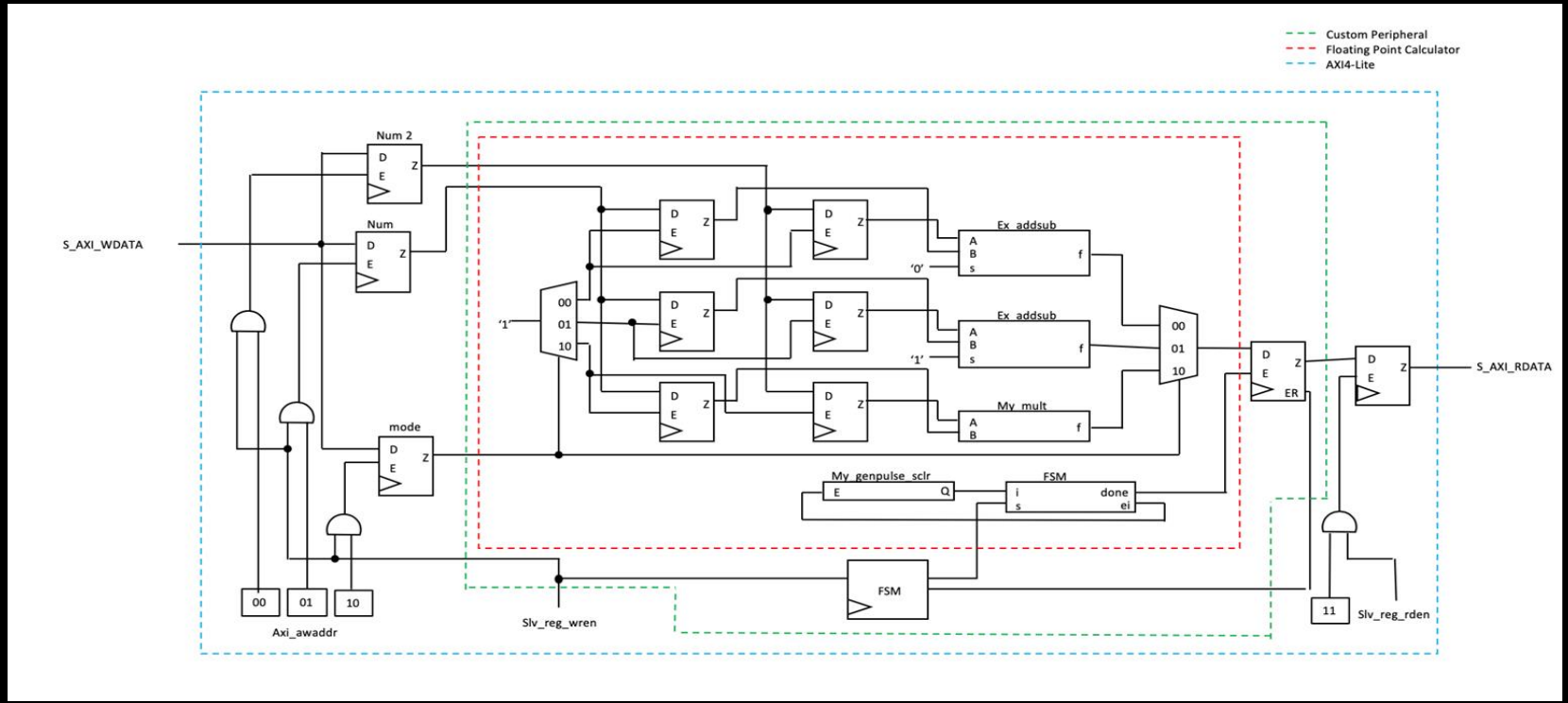
School of Engineering and Computer Science

- Floating Point Calculator
  - Addition
  - Subtraction
  - Multiplication



Block Diagram of the Digital System

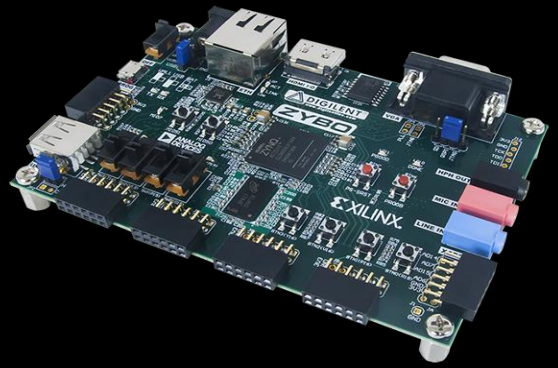
School of Engineering and Computer Science

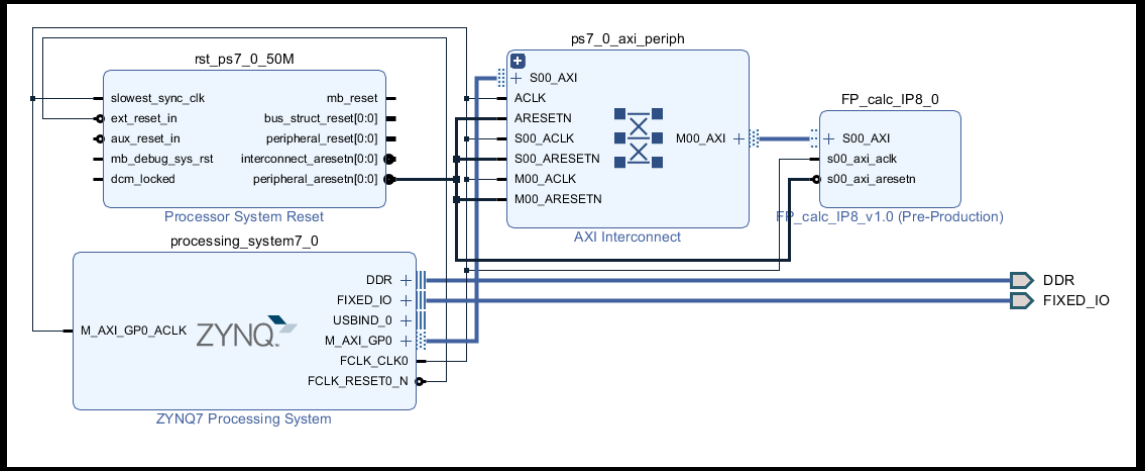


Bus Interface

School of Engineering and Computer Science

- Connects the digital system to the ARM processor on the ZYBO Z-10





Software Routine

School of Engineering and Computer Science

- Using the SDK terminal to read the results over UART

1. Define the base address.

2. Write a 32-bit word to slave_register_0. This is number 1.

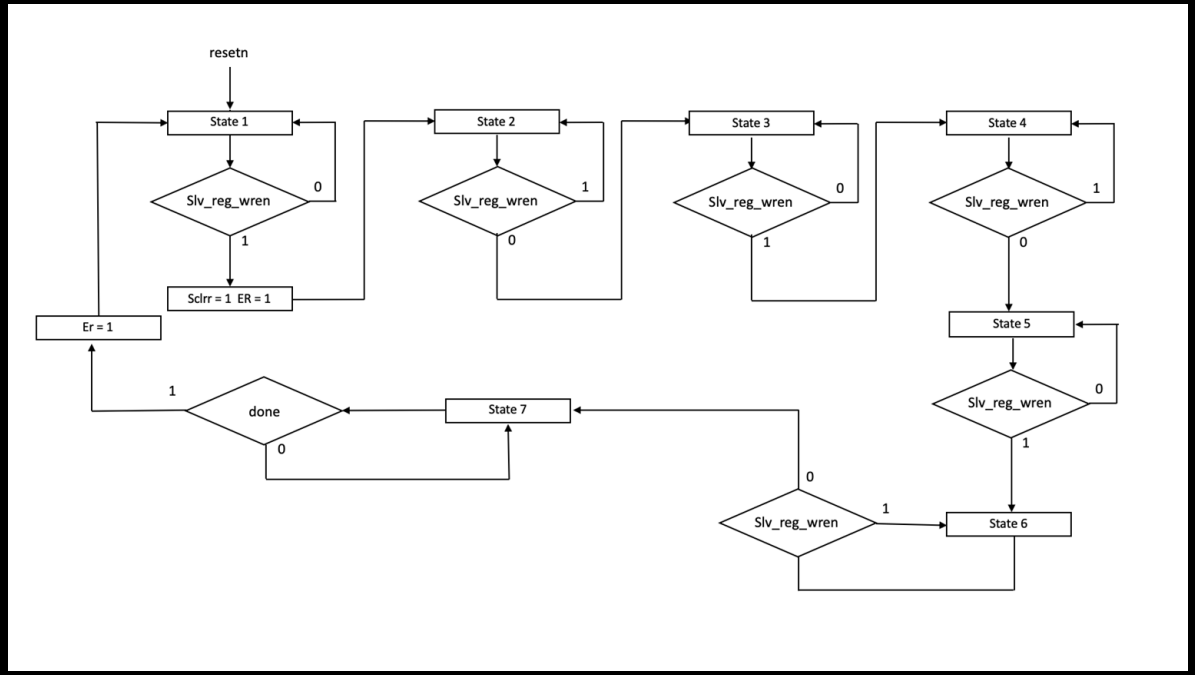
3. Write a 32-bit word to slave_register_1. This is number 2.

4. Write a 32-bit word to slave_register_2. This is the operation.

5. Display the calculated value from the output register, slave_register_3.

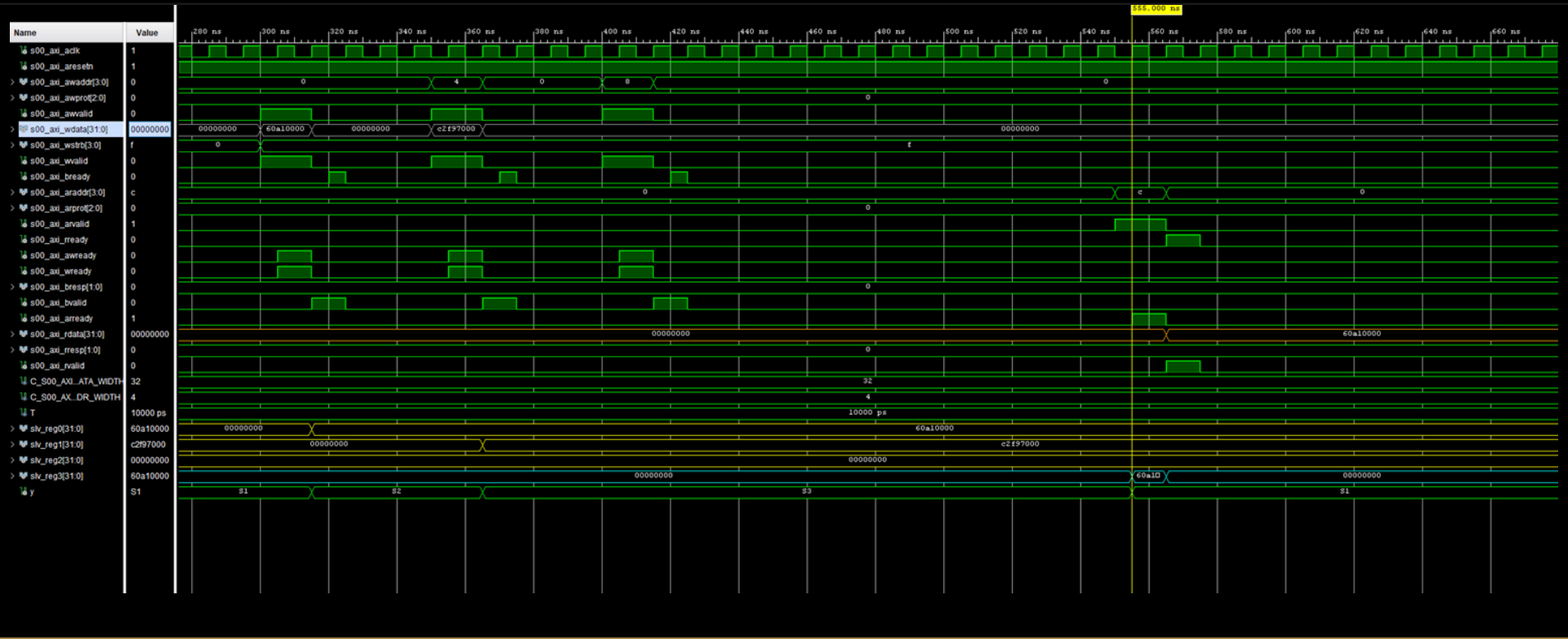
FSM

School of Engineering and Computer Science



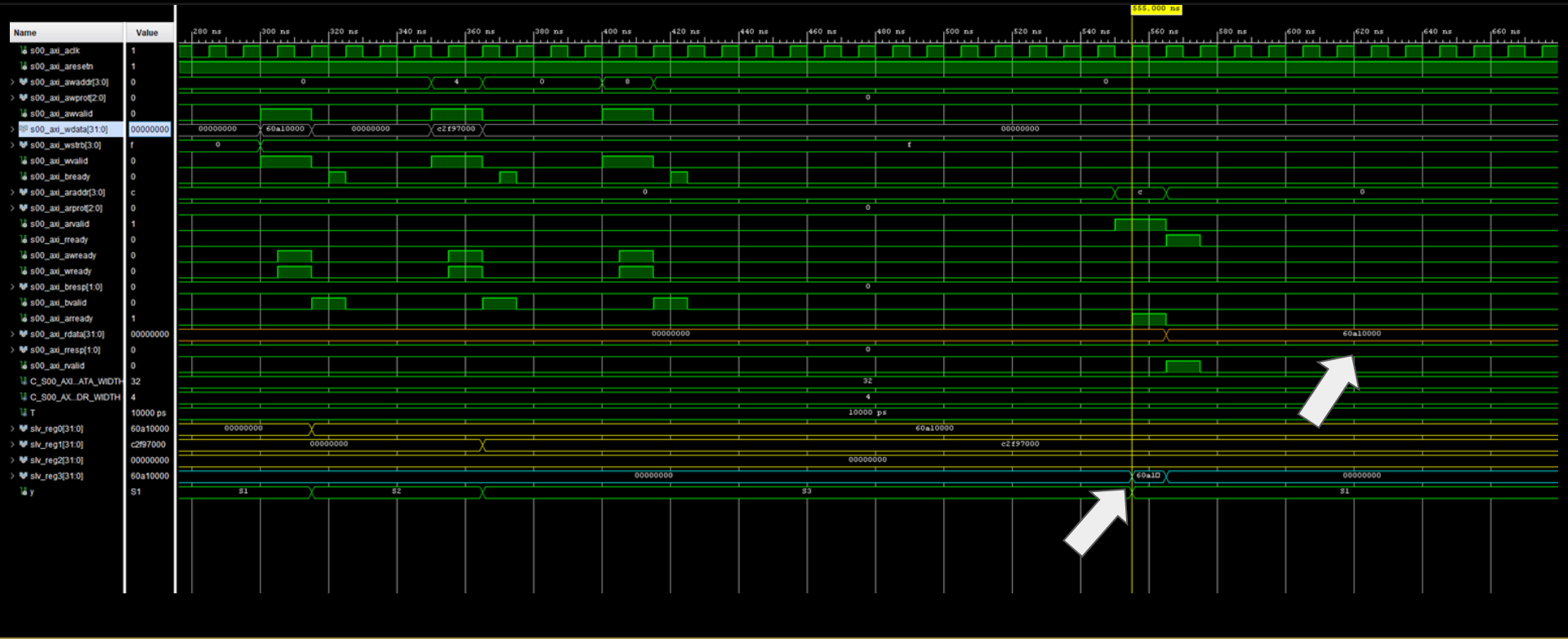
Problems

School of Engineering and Computer Science



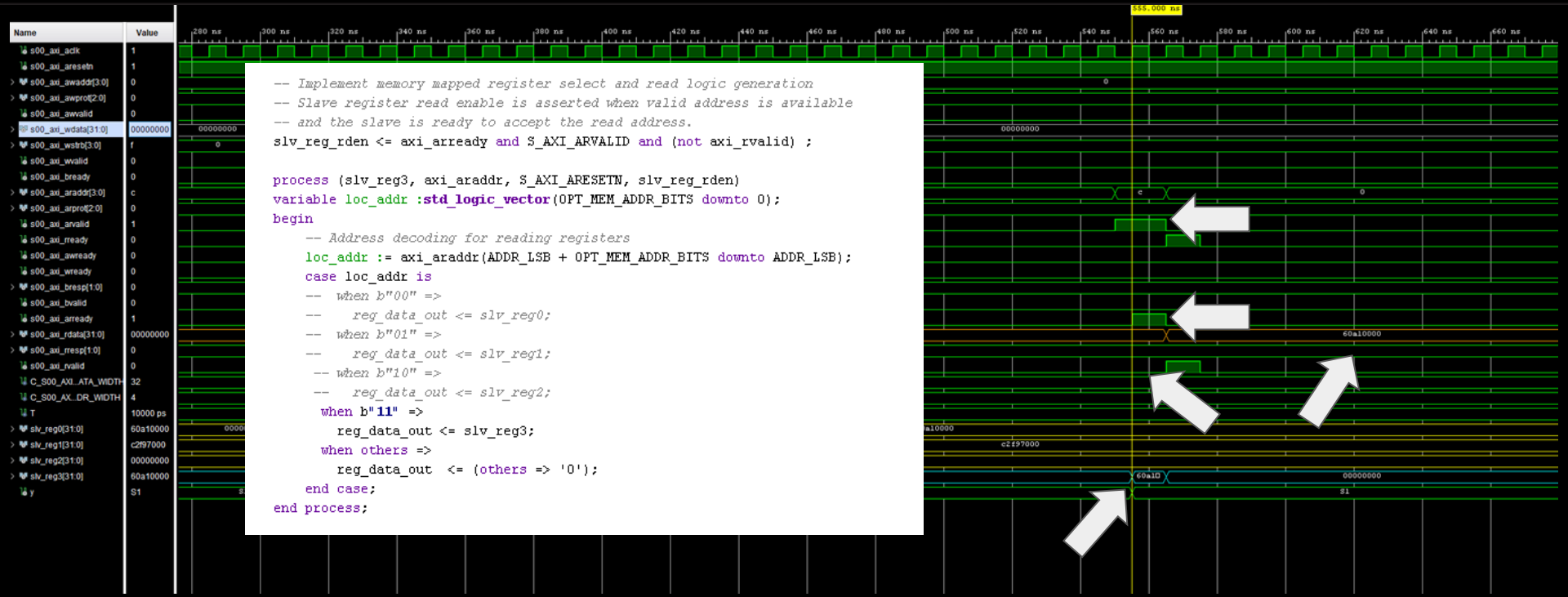
Problems

School of Engineering and Computer Science



Problems

School of Engineering and Computer Science

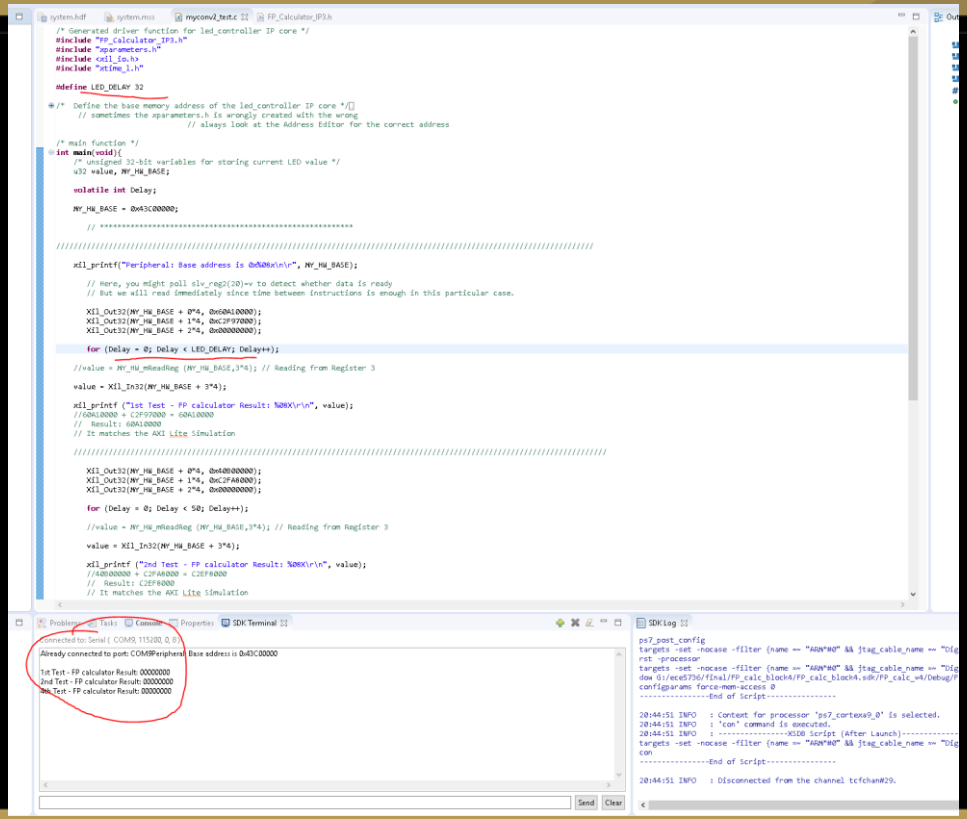



```
-- Implement memory mapped register select and read logic generation
-- Slave register read enable is asserted when valid address is available
-- and the slave is ready to accept the read address.
slv_reg_rden <= axi_arready and S_AXI_ARVALID and (not axi_rvalid) ;

process (slv_reg3, axi_araddr, S_AXI_ARESETN, slv_reg_rden)
variable loc_addr :std_logic_vector(OPT_MEM_ADDR_BITS downto 0);
begin
    -- Address decoding for reading registers
    loc_addr := axi_araddr(ADDR_LSB + OPT_MEM_ADDR_BITS downto ADDR_LSB);
    case loc_addr is
    --  when b"00" =>
    --    reg_data_out <= slv_reg0;
    --  when b"01" =>
    --    reg_data_out <= slv_reg1;
     -- when b"10" =>
     --   reg_data_out <= slv_reg2;
      when b"11" =>
        reg_data_out <= slv_reg3;
      when others =>
        reg_data_out  <= (others => '0');
    end case;
end process;
```

Problems

School of Engineering and Computer Science

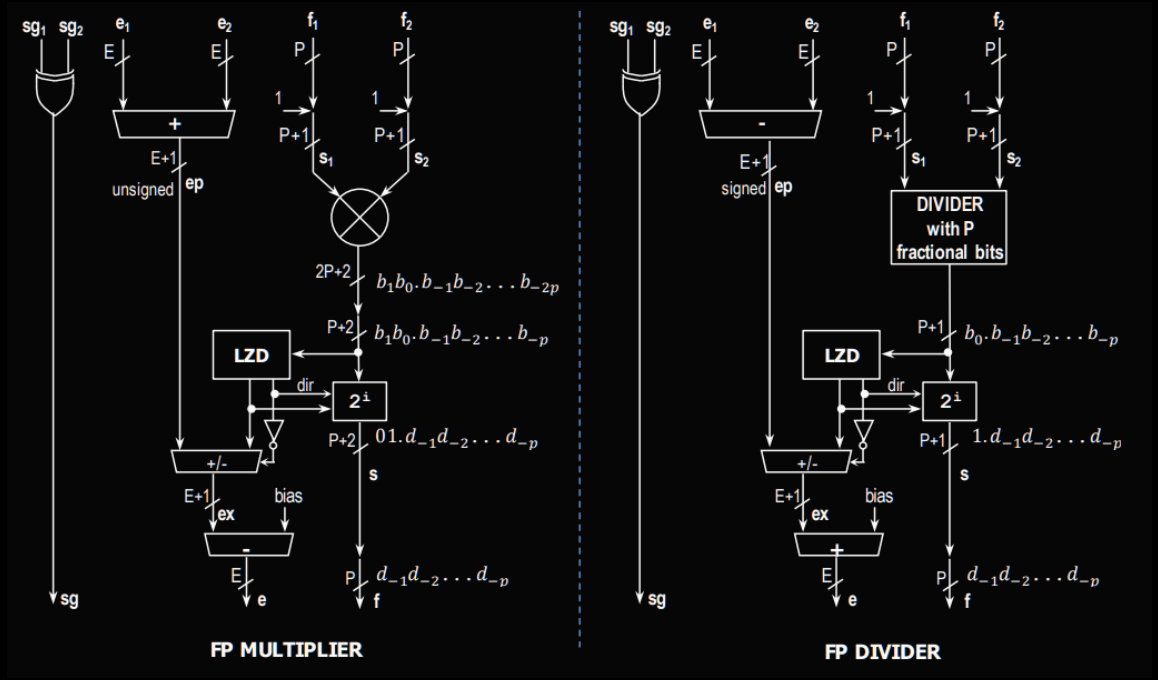


- Improper implementation of the FSM caused data to be read incorrectly in the software routine

Problems

School of Engineering and Computer Science

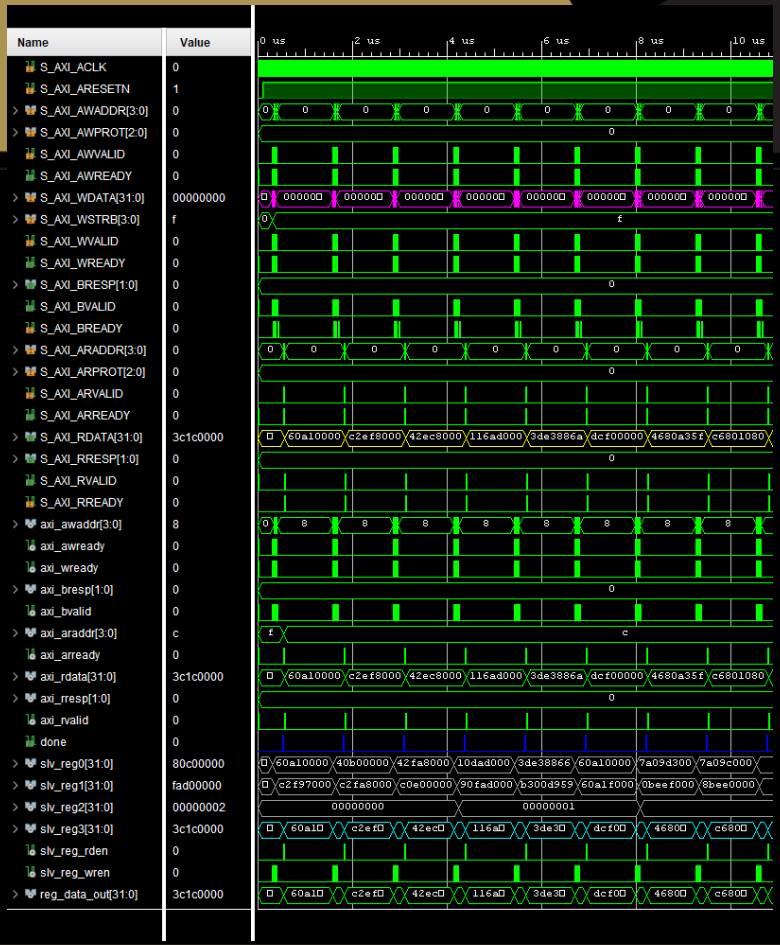
- Unable to properly implement the unsigned floating point divider
- Overflow and underflow conditions for the multiplier



Testbench

School of Engineering and Computer Science

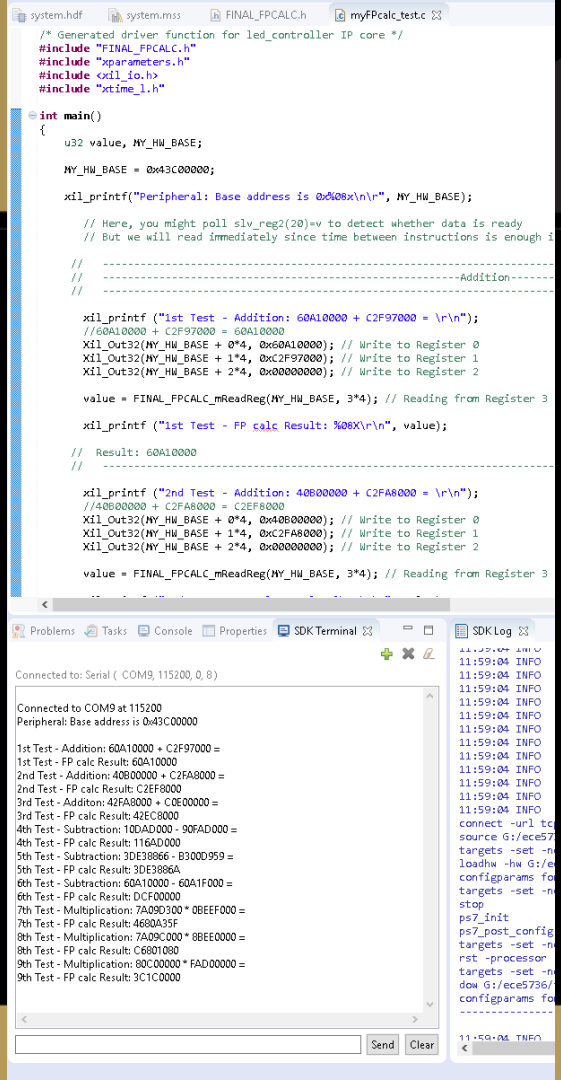
- Implemented simulation tests at the FP calculator and AXI level



- Magenta = write data
- Gray = input to slave registers
- Dark Blue = done bit sent to FSM
- Light Blue = output to slave register
- Yellow = read data

Results

- Successful writing and reading of the circuit



School of Engineering and Computer Science

Lessons Learned

School of Engineering and Computer Science

- Importance of testbench simulations
- Time constraints

Questions?