

Floating Point Calculator

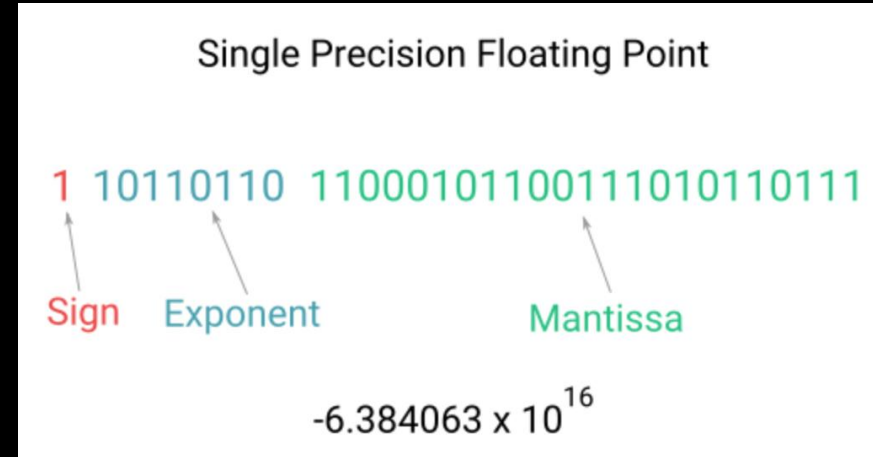
ECE 5736 - Professor Llamocca

Dylan Powell
Brandon Troppens

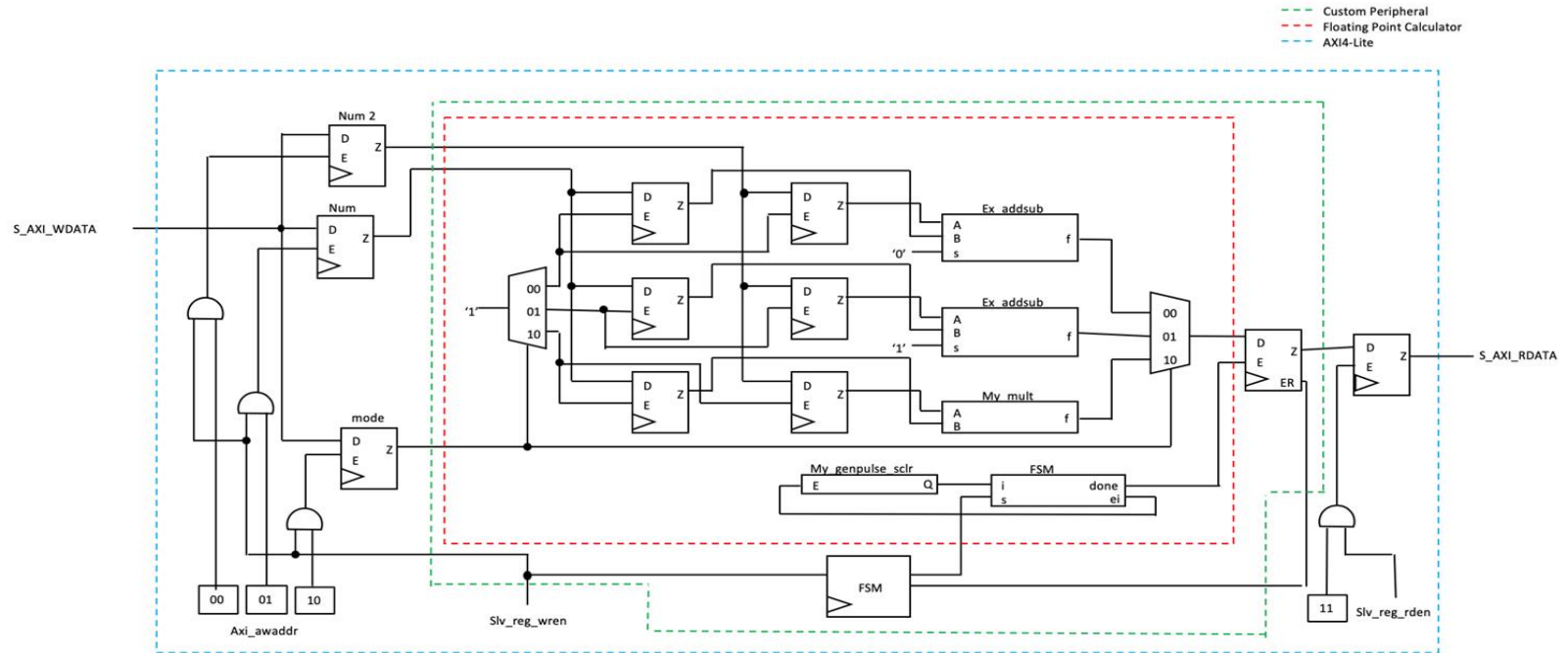


Objective

- Floating Point Calculator
 - Addition
 - Subtraction
 - Multiplication

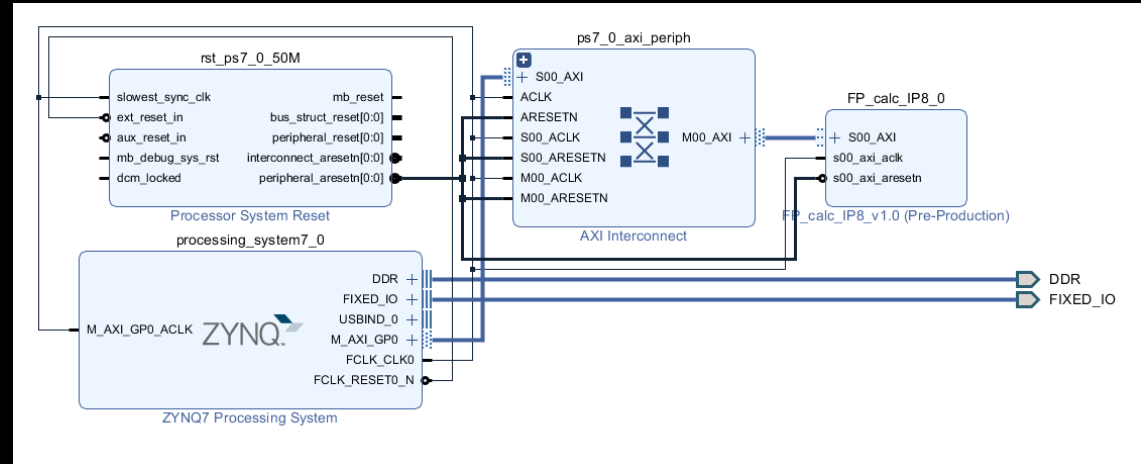
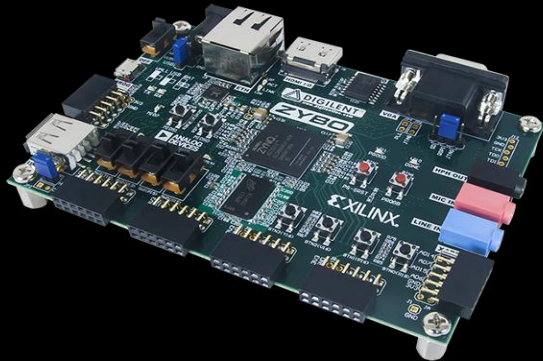


Block Diagram of the Digital System



Bus Interface

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



Software Routine



School of Engineering and
Computer Science

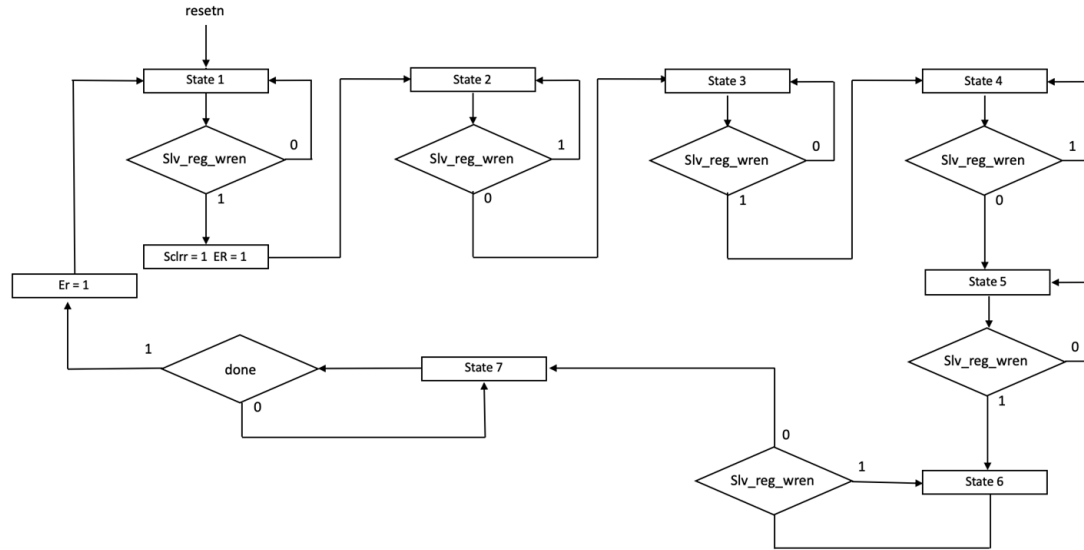
- Using the SDK terminal to read the results over UART

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



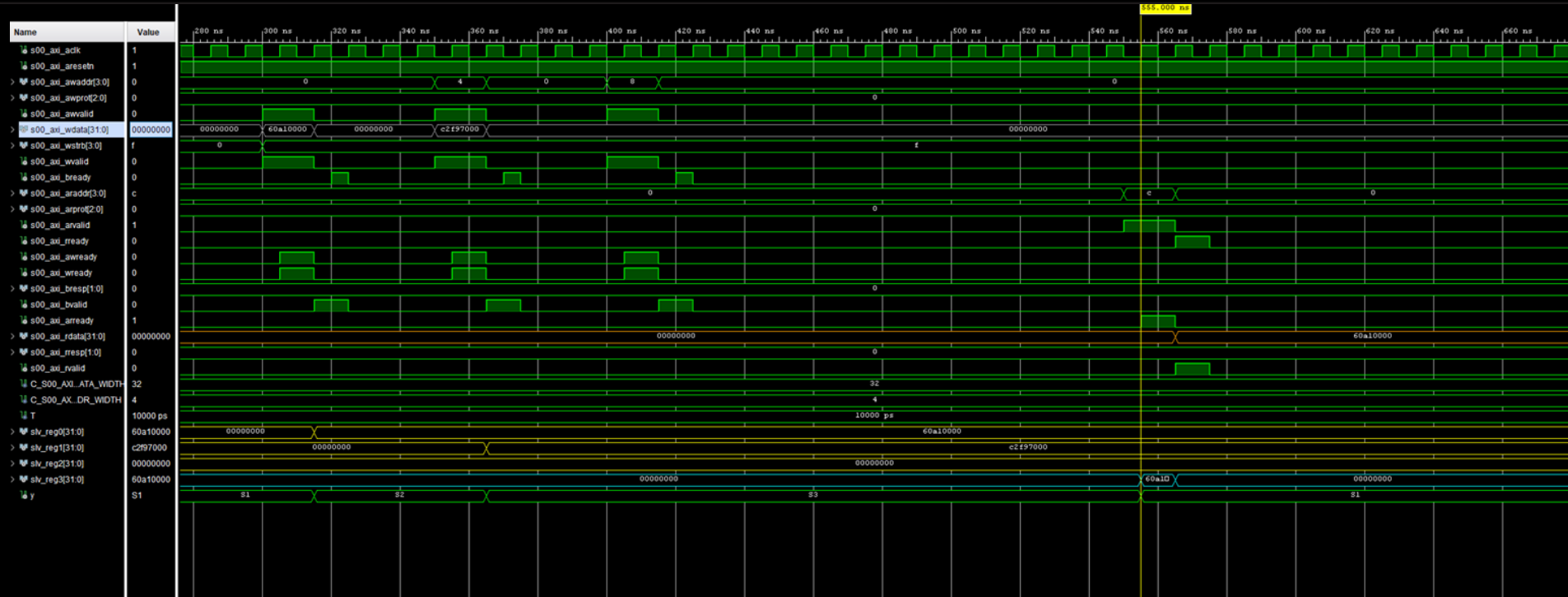
School of Engineering and
Computer Science



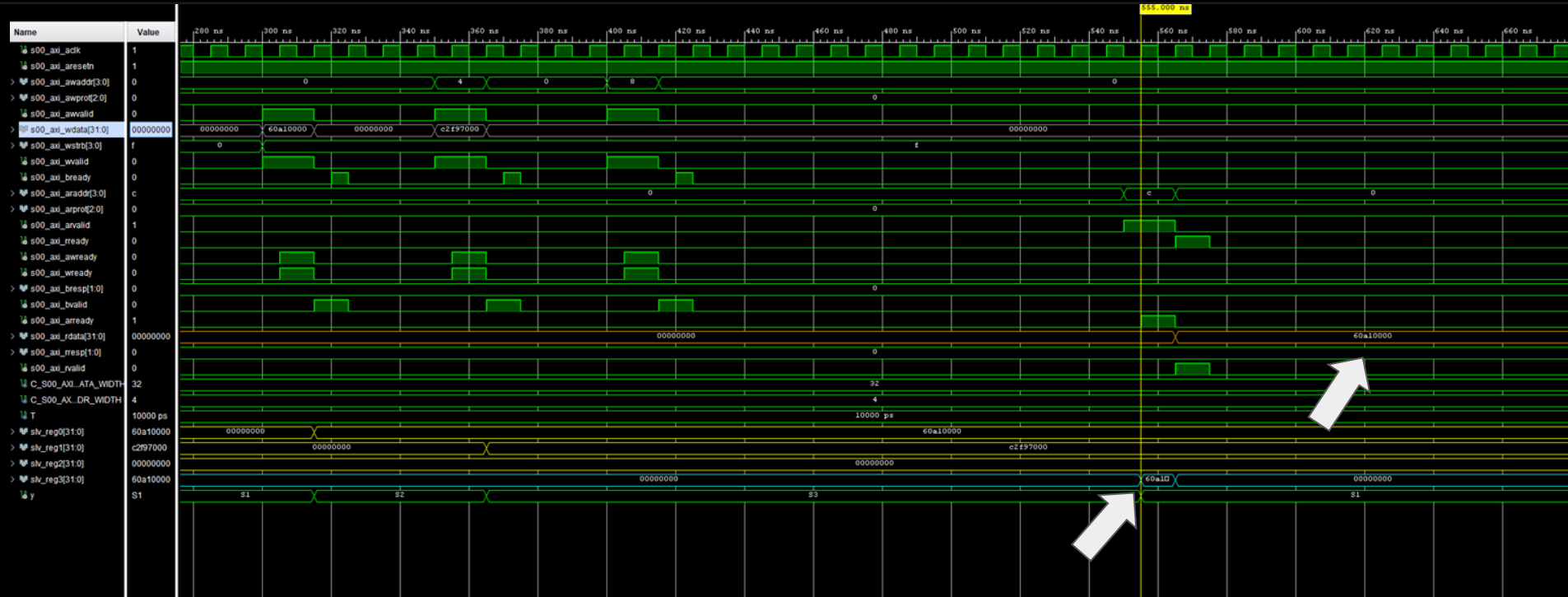
Problems



School of Engineering and
Computer Science



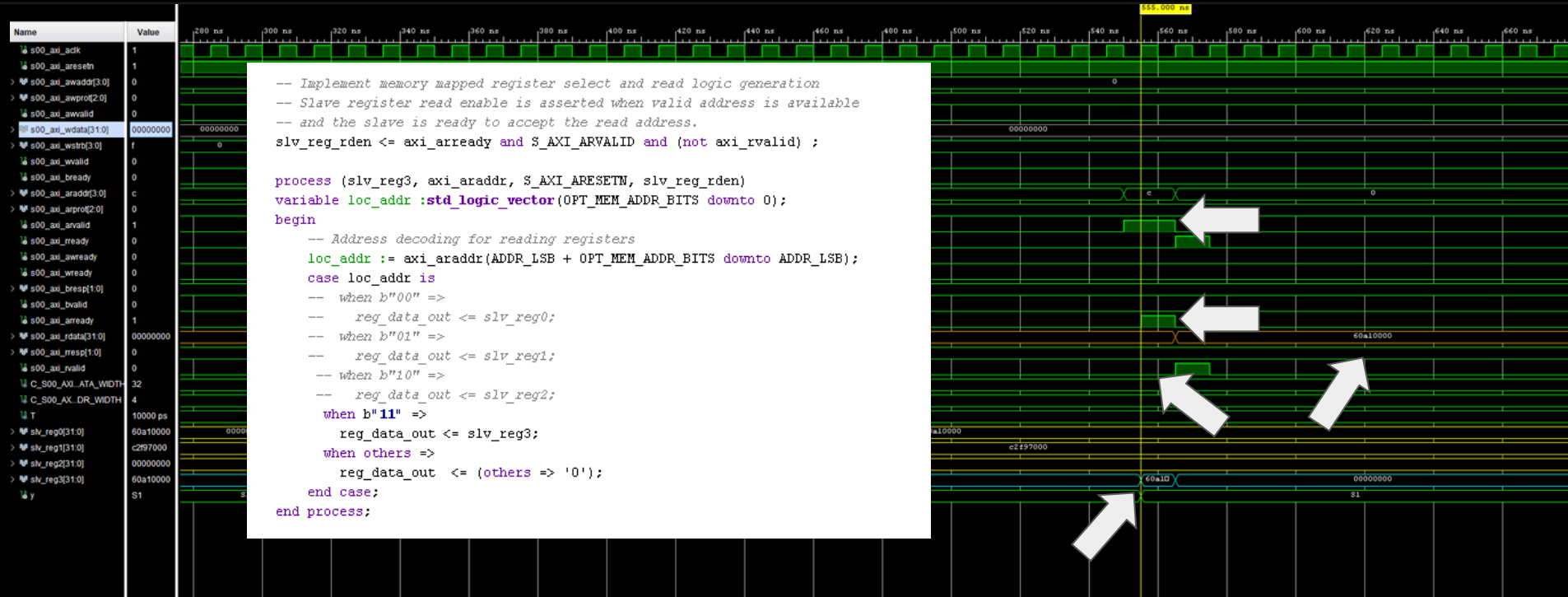
Problems



Problems



School of Engineering and
Computer Science



Problems



School of Engineering and
Computer Science

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

```
system.h  system.mss  myconv2_test.c  FP_calculator_IP1.h
/* Generated driver function for led_controller IP core */
#include "FP_calculator_IP1.h"
#include "parameters.h"
#include "util.h"
#include "util.h"

#define LED_DELAY 32

/* Define the base memory address of the led_controller IP core */
// sometimes the parameters.h is wrongly created with the wrong
// always look at the Address Editor for the correct address

/* main function */
int main(void){
    /* unsigned 32-bit variables for storing current LED value */
    u32 value, MY_HM_BASE;

    volatile int Delay;

    MY_HM_BASE = 0x43C00000;
    // *****

    // =====
    xil_printf("Peripheral Base address is 0x%08lx\n", MY_HM_BASE);
    // Here, you might poll s1v_reg(20) to detect whether data is ready
    // But we will read immediately since time between instructions is enough in this particular case.
    XIL_Out32(MY_HM_BASE + 0*4, 0x00000000);
    XIL_Out32(MY_HM_BASE + 1*4, 0xC2F48000);
    XIL_Out32(MY_HM_BASE + 2*4, 0x00000000);

    for (Delay = 0; Delay < LED_DELAY; Delay++){
        //value = MY_HM_mread32(MY_HM_BASE, 3*4); // Reading from Register 3
        value = XIL_In32(MY_HM_BASE + 3*4);
        xil_printf("1st Test - FP calculator Result: %08X\n", value);
        //0x00000000 + C2F48000 = 0x00000000
        // Result: 0x00000000
        // It matches the AXI Lite Simulation

        // =====
        XIL_Out32(MY_HM_BASE + 0*4, 0x00000000);
        XIL_Out32(MY_HM_BASE + 1*4, 0xC2F48000);
        XIL_Out32(MY_HM_BASE + 2*4, 0x00000000);

        for (Delay = 0; Delay < 50; Delay++){
            //value = MY_HM_mread32(MY_HM_BASE, 3*4); // Reading from Register 3
            value = XIL_In32(MY_HM_BASE + 3*4);
            xil_printf("2nd Test - FP calculator Result: %08X\n", value);
            //0x00000000 + C2F48000 = C2F48000
            // Result: C2F48000
            // It matches the AXI Lite Simulation
        }
    }
}
```

SDKlog 22

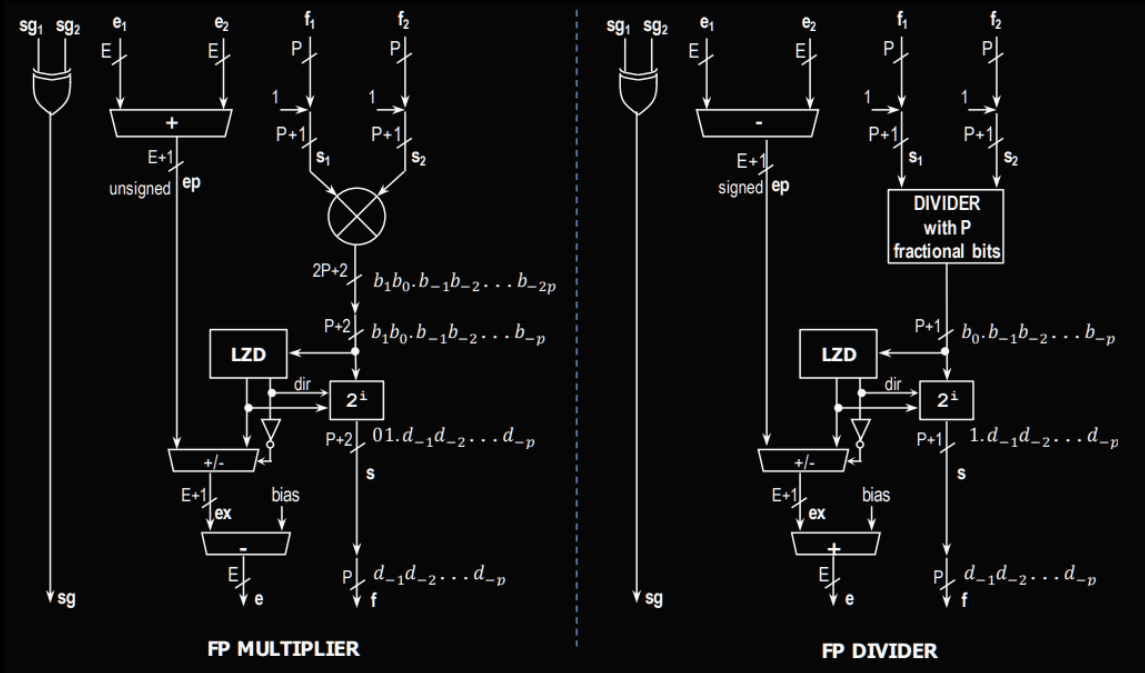
```
ps7_post_config
targets -set -nocase -filter (name == "AXI4") && tag_cable_name == "Dig
rst_processor
targets -set -nocase -filter (name == "AXI4") && tag_cable_name == "Dig
dsw 01/0007350/fin1/PP_calc_block4/PP_calc_block4.s0/PP_calc_v4/Debug/PP
configparams force-mm-access 0
-----End of Script-----
20144:51 INFO : Context for processor "ps7_cortexa9_0" is selected.
20144:51 INFO : "can" command is executed.
20144:51 INFO : -----JTAG Script (After Launch)-----
targets -set -nocase -filter (name == "AXI4") && tag_cable_name == "Dig
600.
-----End of Script-----
20144:51 INFO : Disconnected from the channel tcfchanW29.
```

Problems 1 Task 1 Console Properties SDK Terminal 22

```
Connected to Serial ( COM3, 115200, 8, N)
Already connected to port: COM3@Peripheral Base address is 0x43C00000
1st Test - FP calculator Result: 00000000
2nd Test - FP calculator Result: 00000000
3rd Test - FP calculator Result: 00000000
```


Problems

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



Testbench

- 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

```
system.hdf  system.mss  FINAL_FPCALC.h  myFPCalc_test.c
```

```
/* Generated driver function for led_controller IP core */
#include "FINAL_FPCALC.h"
#include "xparameters.h"
#include <xil_io.h>
#include "xtime_l.h"

int main()
{
    u32 value, MY_HW_BASE;

    MY_HW_BASE = 0x43C00000;

    xil_printf("Peripheral: Base address is 0x%08x\n\r", MY_HW_BASE);

    // Here, you might poll sly_reg2(20)w to detect whether data is ready
    // But we will read immediately since time between instructions is enough i

    // -----Addition-----
    // -----

    xil_printf ("1st Test - Addition: 60A10000 + C2F97000 = \r\n");
    //60A10000 + C2F97000 = 60A10000
    Xil_Out32(MY_HW_BASE + 0*4, 0x60A10000); // Write to Register 0
    Xil_Out32(MY_HW_BASE + 1*4, 0xC2F97000); // Write to Register 1
    Xil_Out32(MY_HW_BASE + 2*4, 0x00000000); // Write to Register 2

    value = FINAL_FPCALC_mReadReg(MY_HW_BASE, 3*4); // Reading from Register 3

    xil_printf ("1st Test - FP calc Result: %08X\r\n", value);

    // Result: 60A10000
    // -----

    xil_printf ("2nd Test - Addition: 40B00000 + C2FA8000 = \r\n");
    //40B00000 + C2FA8000 = C2EF8000
    Xil_Out32(MY_HW_BASE + 0*4, 0x40B00000); // Write to Register 0
    Xil_Out32(MY_HW_BASE + 1*4, 0xC2FA8000); // Write to Register 1
    Xil_Out32(MY_HW_BASE + 2*4, 0x00000000); // Write to Register 2

    value = FINAL_FPCALC_mReadReg(MY_HW_BASE, 3*4); // Reading from Register 3
```

Problems Tasks Console Properties SDK Terminal SDK Log

Connected to Serial (COM9, 115200, 0, 8)

Connected to COM9 at 115200
Peripheral: Base address is 0x43C00000

```
1st Test - Addition: 60A10000 + C2F97000 =
1st Test - FP calc Result: 60A10000
2nd Test - Addition: 40B00000 + C2FA8000 =
2nd Test - FP calc Result: C2EF8000
3rd Test - Addition: 42FA8000 + C0E00000 =
3rd Test - FP calc Result: 42EC8000
4th Test - Subtraction: 10DAD000 - 90FAD000 =
4th Test - FP calc Result: 116AD000
5th Test - Subtraction: 3DE38866 - B300D959 =
5th Test - FP calc Result: 3DE3886A
6th Test - Subtraction: 60A10000 - 60A1F000 =
6th Test - FP calc Result: DCF00000
7th Test - Multiplication: 7AD9C300 * 0BEFF000 =
7th Test - FP calc Result: 4680A35F
8th Test - Multiplication: 7AD9C300 * 0BEE0000 =
8th Test - FP calc Result: C6801080
9th Test - Multiplication: B0C00000 * FAD00000 =
9th Test - FP calc Result: 3C1C0000
```

Send Clear



School of Engineering and
Computer Science

Lessons Learned



School of Engineering and
Computer Science

- Importance of testbench simulations
- Time constraints



Questions?