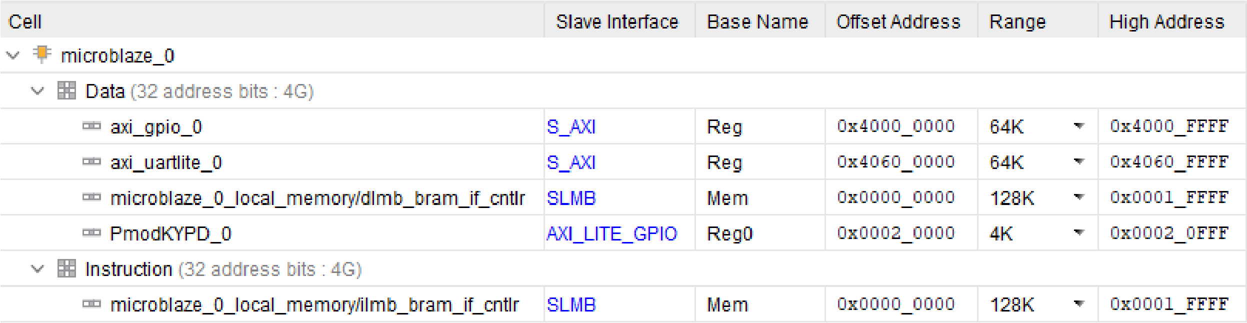
CS 152B Lab 3 Report

Introduction

In this lab, we focus on implementing serial communication between a MicroBlaze processor on an FPGA board and a PC terminal. When engineering serial communication, several key standards and constraints must be considered. The baud rate is an important constraint that determines the rate at which data is transmitted. In this lab, we use a baud rate of 9600 bps. To address memory constraints, we optimize memory usage by replacing functions like printf with xil_printf. The engineering standard of real-time processing was handled in implementing a game of rock-paper-scissors between the FPGA board and PC terminal where our program waits for both inputs before determining the winner.



Cell	Slave Interface	Base Name	Offset Address	Range	High Address
microblaze_0					
Data (32 address bits : 4G)					
axi_gpio_0	S_AXI	Reg	0x4000_0000	64K	0x4000_FFFF
axi_uartlite_0	S_AXI	Reg	0x4060_0000	64K	0x4060_FFFF
microblaze_0_local_memory/dlmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	128K	0x0001_FFFF
PmodKYPD_0	AXI_LITE_GPIO	Reg0	0x0002_0000	4K	0x0002_0FFF
Instruction (32 address bits : 4G)					
microblaze_0_local_memory/ilmb_bram_if_cntlr	SLMB	Mem	0x0000_0000	128K	0x0001_FFFF

Figure 1. Increased memory constraints due to buffer overflow for cout.

Implementation

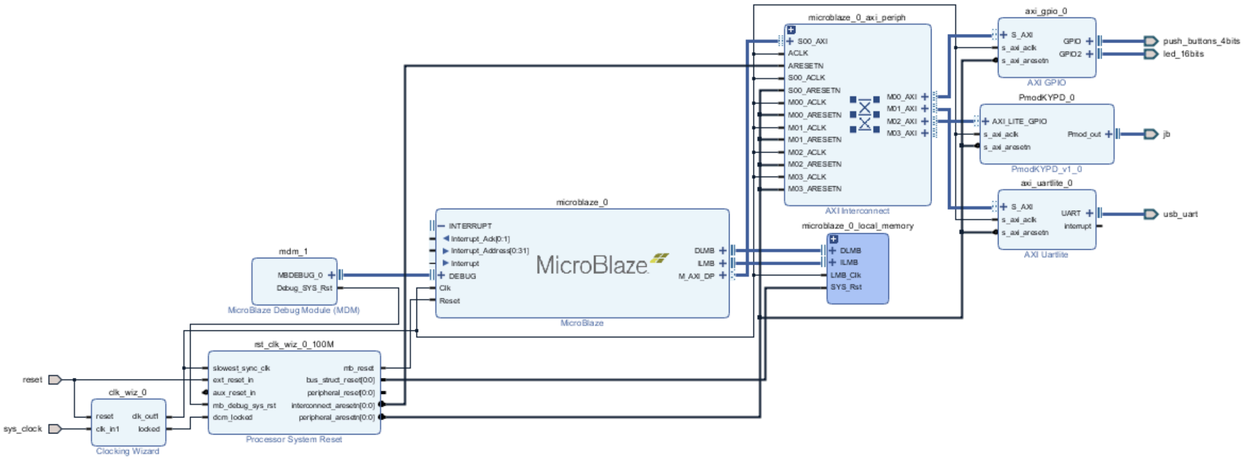


Figure 2. Microblaze block diagram.

Part 2.a:

```
#include "xgpio.h"
#include "xil_printf.h"
#include "xparameters.h"
#include <stdio.h>

#define LED_CHANNEL 1
#define LED_MASK 0x1
```

```
#define THRESHOLD 100

XGpio Gpio;

int main() {
    int num1, num2, product;
    char delimiter;

    XGpio_Initialize(&Gpio, XPAR_GPIO_0_DEVICE_ID);
    XGpio_SetDataDirection(&Gpio, LED_CHANNEL, 0x0);

    xil_printf("MicroBlaze Multiplier Ready!\n\r");

    while (1) {
        xil_printf("\n\rEnter two numbers (format: a/b): ");

        int scanf_output = scanf("%d%c%d", &num1, &delimiter, &num2);

        if (scanf_output == 3 && delimiter == '/') {
            product = num1 * num2;
            xil_printf("\n\rProduct: %d\n\r", product);

            // If product > 100, turn on LED
            if (product > THRESHOLD) {
                XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, LED_MASK);
            } else {
                XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, 0x0);
            }
        } else {
            xil_printf("\n\rInvalid input! Please enter two numbers separated by '/'.\n\r");
            // Clear input buffer to avoid infinite loops
            while (getchar() != '\n');
        }
    }
    return 0;
}
```

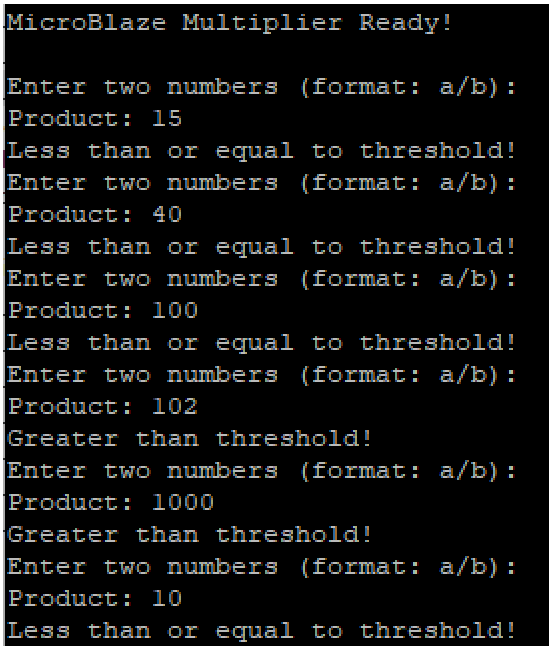
Part 2.b:

```
#include "PmodKYPD.h"
#include "sleep.h"
#include "xil_cache.h"
#include "xparameters.h"

#define DEFAULT_KEYTABLE "0FED789C456B123A"
```

```
PmodKYPD myDevice;

void DemoInitialize();
void DemoRun();
void DemoCleanup();
void DisableCaches();
void EnableCaches();
void DemoSleep(u32 millis);

int main(void) {
    DemoInitialize();
    DemoRun();
    DemoCleanup();
    return 0;
}

// Enter 1 for Rock, 2 for Paper, 3 for Scissors.

const char* determine_winner(int player1, int player2) {
    if (player1 == player2) {
        return "\r\nIt's a tie!";
    }
    if ((player1 == 3 && player2 == 1) || (player1 == 1 && player2 == 2) || (player1 == 2 &&
player2 == 3)) {
        return "\r\nFPGA won!";
    }
    return "\r\nPC won!";
}

void DemoInitialize() {
    EnableCaches();
    KYPD_begin(&myDevice, XPAR_PMODKYPD_0_AXI_LITE_GPIO_BASEADDR);
    KYPD_loadKeyTable(&myDevice, (u8*) "0FED789C456B123A");
}

void DemoRun() {
    u16 keystate;
    XStatus status, last_status = KYPD_NO_KEY;
    u8 key, last_key = 'x';
    int pc_choice, fpga_choice;
    char key_press;

    xil_printf("Rock-Paper-Scissors\r\n");
```

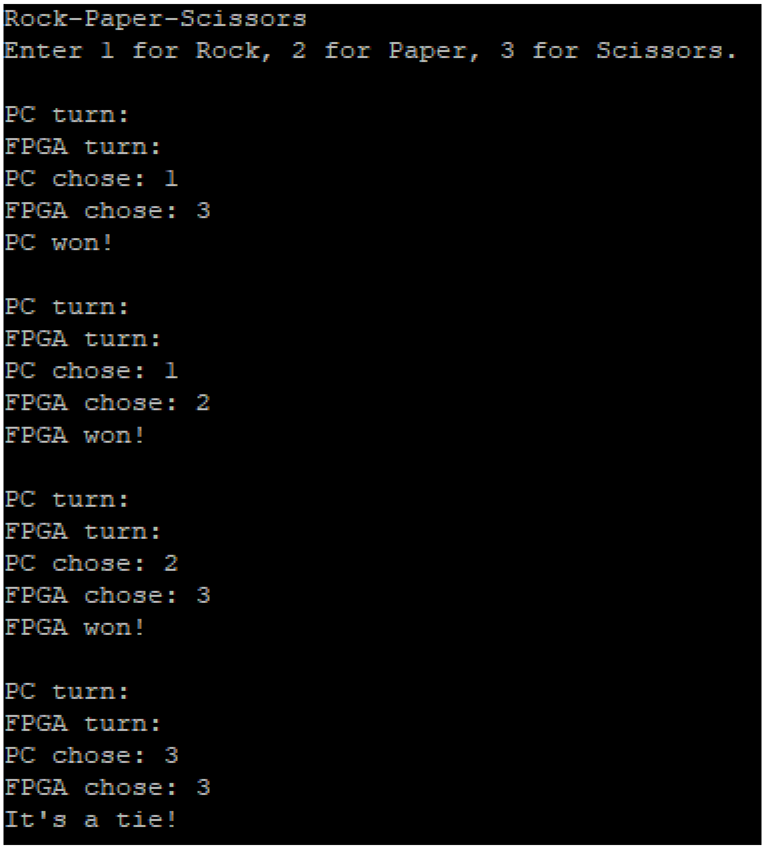
```
  xil_printf("Enter 1 for Rock, 2 for Paper, 3 for Scissors.\r\n");

  while (1) {
    xil_printf("\r\nPC turn:");
    scanf("%d", &pc_choice); // Input from the terminal (PC)

    // Get FPGA input from the keypad
    xil_printf("\r\nFPGA turn:");

    while (1) {
      keystate = KYPD_getKeyStates(&myDevice);

                  status = KYPD_getKeyPressed(&myDevice, keystate, &key);
                  /*xil_printf("KYPD status: %d", status);
                  xil_printf("KYPD expected status: %d", KYPD_SINGLE_KEY);
                  xil_printf("KYPD key press: %s", key_press);*/

                  if (status == KYPD_SINGLE_KEY && (status != last_status || key !=
last_key)) {
                        last_key = key;
                        break;
                  }

                  last_status = status;
    }

    key_press = (char) key;
            if (key_press == '1') {
                  fpga_choice = 1; // Rock
            } else if (key_press == '2') {
                  fpga_choice = 2; // Paper
            } else if (key_press == '3') {
                  fpga_choice = 3; // Scissors
            } else {
                  xil_printf("Invalid input. Please press 1, 2, or 3 on the keypad.\r\n");
                  continue;
            }

            // Display choices
            xil_printf("\r\nPC chose: %d", pc_choice);
            xil_printf("\r\nFPGA chose: %d", fpga_choice);

            // Determine winner
            const char* result = determine_winner(pc_choice, fpga_choice);
            xil_printf("%s\r\n", result);

    usleep(1000); // Wait for 1 second before next round

    // Reinitialize variables
    status, last_status = KYPD_NO_KEY;
    key, last_key = 'x';
  }
}

void DemoCleanup() {
  DisableCaches();
}

void EnableCaches() {
#ifdef __MICROBLAZE__
#ifdef XPAR_MICROBLAZE_USE_ICACHE
  Xil_ICacheEnable();
#endif
#ifdef XPAR_MICROBLAZE_USE_DCACHE
  Xil_DCacheEnable();
#endif
#endif
}

void DisableCaches() {
#ifdef __MICROBLAZE__
#ifdef XPAR_MICROBLAZE_USE_DCACHE
  Xil_DCacheDisable();
#endif
#ifdef XPAR_MICROBLAZE_USE_ICACHE
  Xil_ICacheDisable();
#endif
#endif
}
```

Testing



```
MicroBlaze Multiplier Ready!

Enter two numbers (format: a/b):
Product: 15
Less than or equal to threshold!
Enter two numbers (format: a/b):
Product: 40
Less than or equal to threshold!
Enter two numbers (format: a/b):
Product: 100
Less than or equal to threshold!
Enter two numbers (format: a/b):
Product: 102
Greater than threshold!
Enter two numbers (format: a/b):
Product: 1000
Greater than threshold!
Enter two numbers (format: a/b):
Product: 10
Less than or equal to threshold!
```

Figure 3. Terminal for part 2.a.



```
Rock-Paper-Scissors
Enter 1 for Rock, 2 for Paper, 3 for Scissors.

PC turn:
FPGA turn:
PC chose: 1
FPGA chose: 3
PC won!

PC turn:
FPGA turn:
PC chose: 1
FPGA chose: 2
FPGA won!

PC turn:
FPGA turn:
PC chose: 2
FPGA chose: 3
FPGA won!

PC turn:
FPGA turn:
PC chose: 3
FPGA chose: 3
It's a tie!
```

Figure 4. Terminal for part 2.b.

Challenges

- One challenge we faced was following the tutorial for setting up Microblaze on the FPGA board 🙁. Directions are hard apparently.
- Another challenge we faced was the integration of the Pmod KYPD IP Core into the Microblaze block design. This module had specific functionalities that deviated from the standard way of setting up an IP Core, such as the lack of an interrupt input or a set clock frequency. To resolve this, we had to read the Basys3 Pmod reference manual for any special requirements or nuances the KYPD contained.
- A significant issue we faced was the strict memory requirements of the basys3 board. Even changing all memory-intensive functions like printf to their Xilinx equivalents (e.g. xil_printf), simple code could not run on default settings. To resolve this, we had to manually increase the data memory region and reprogram the FPGA board to utilize this hardware configuration.

Participation
105817312 - setting up Microblaze, coding 2a and 2b, debugging
905699244 - coding 2a and 2b, debugging

152B Lab 1~4 Report Rubric

Implementation: 50

Report: 25

- Introduction: 5
    - 5/5 - Relevant engineering standards and constraints were discussed
    - 4/5 - Both were vaguely discussed
    - 3/5 - Only one of standards or constraints were discussed
    - 2/5 - Introduction is present, but neither engineering standards nor constraints were discussed
    - 0/5 - Introduction missing
- Schematics and code: 6 (8)
    - 8/8 - schematics and code are clear, concise, and complete
    - 7~4/8 - schematics and code are not related to the descriptions or project
    - 0/8 - no code/schematics
- Description of components: 6 (8)

    - 8/8 - descriptions are clear, concise, and complete
    - 7~4/8 - unclear or incorrect descriptions
    - 0/8 - no description
- Challenges: 3 (4)
    - 1 (1.25) points per thoughtful challenge
- Questions: 5 (0)
    - Will vary with each lab

Test: 15

- 15/15 - tests validate all cases
- 10/15 - 1 or 2 edge cases missing
- 5/15 - many cases missing or main functionality not shown
- 0/15 - no waveform

Participation: 10