Lab Assignment 10

Controlling a Robotic Arm via a WiiMote through

the ARM CPU and the FPGA

Xiang Fan

Fan.x@husky.neu.edu

Bo Pang

Pang.b@husky.neu.edu

Submit date: 12/12/2015
Due Date: 12/12/2015

Abstract

For this lab the goal was set to bring together all of the elements you have developed so far in previous labs, and combine them in a hardware/software co-designed implementation.

Introduction

The purpose of this laboratory was to bring together all of the elements we have developed so far in previous labs, and combine them in a hardware/software co-designed implementation. In this lab we controlled the Robotic arm using the FPGA (as you did in Lab 9) and useed the ARM processor on the ZedBoard to receive commands from the WiiMote and control the FPGA.

Lab Discussion

Hardware: Zedboard, Dell desktop computer, Robotic arm, WiiMote

The Zedboard is running a version of Ubuntu called Xillinux (Xilinx + Linux).

It's running on an ARM processor.

Software: MobaXterm, ISE Design Suite 14.4

Set up: We first plugged in the Zedboard and connect it to the host via an Ethernet cable. Then we ran MobaXterm on computer to ssh to Zedboard. We Selected menu Sessions → New session → SSH, and fill in the basic SSH settings. The IP address of the ZedBoard is 192.168.1.10, and the default SSH port is 22.

Connect Wiimote to ZedBoard through Bluetooth.

Result and Analysis

10.1

For this part, we replaced constants with Inputs and compiled the design using HDL workflow advisor, then we run the C program which takes two inputs from the user (servo number, and servo position) and then controlled the robotic arm

The lowest speed is 1 degree/sec, Highest speed is about 30 degree/s based on our observation.

10.2

The ServoControlWSpeed.slx can move multiple servos at the same time and at their own speed, but the previous design can't.

Our robotic arm moved to a predefined position and trying to grab a box and throw it away, but can't lift it up. We thought it may because of the unstable of our gripper.

10.4

Both buttons and acceleration values can be read.

Conclusion:

After this lab we learned how to controll the Robotic arm using the FPGA (as we did in Lab 9) and useed the ARM processor on the ZedBoard to receive commands from the WiiMote and control the FPGA.

Appendix

```
#include <stdio.h>

#include <stdlib.h>

#include <fcntl.h>

#include <sys/mman.h>

#include <time.h>

#include "wiimote.h"


#define BASE_ADDRESS 0x400D0000


//Servo motor offsets

#define Base_OFFSET 0x100

#define Bicep_OFFSET 0x104

#define Elbow_OFFSET 0x108

#define Wrist_OFFSET 0x10C

#define Gripper_OFFSET 0x110


#define REG_WRITE(addr, off, val) (*(volatile int*)(addr+off)=(val))

#define REG_READ(addr, off) *((volatile int*)(addr+off))


#define POS(name) ((REG_READ(gServos.test_base, name##_OFFSET)) & 0xFF)
```

```
//Mask away the speed bits



/**

  * data structure for servo instance

  */

typedef struct {

      unsigned char *test_base; /// base address of mapped virtual space

      int fd;                              /// file desrcriptor for memory map

      int map_len;                        /// size of mapping window



} tServo;



/**

  * global variable for all servos

  */

tServo gServos;



int positions[6];



/**
```

```
 * This function takes the servo number and the position, and writes the values in
 * appropriate address for the FPGA
 * @param test_base             base pointer for servos
 * @param servo_number          servo number to manipulate
 * @param position              new postion in degree (0 .. 180)
 * @param speed                 speed to move in degree / 20ms
 */
void servo_move(unsigned char servo_number, unsigned char position, unsigned
char speed);



/**
 * Initialize servos
 * @return 0 upon success, 1 otherwise
 */
int servo_init() {

    //Open the file regarding memory mapped IO to write values for the FPGA
    gServos.fd = open( "/dev/mem", O_RDWR);

    unsigned long int PhysicalAddress = BASE_ADDRESS;
```

```
    gServos.map_len= 0xFF;    //size of mapping window


    // map physical memory startin at BASE_ADDRESS into own virtual memory
    gServos.test_base = (unsigned char*)mmap(NULL, gServos.map_len,
PROT_READ | PROT_WRITE, MAP_SHARED, gServos.fd, (off_t)PhysicalAddress);


    // did it work?
    if(gServos.test_base == MAP_FAILED)  {
        perror("Mapping memory for absolute memory access failed -- Test Try\n");
        return 1;
    }


    //Initialize all servo motors to middle position, go there fast
    servo_move(0, 150, 100);
    servo_move(1, 150, 100);
    servo_move(2, 150, 100);
    servo_move(3, 150, 100);
    servo_move(4, 150, 100);
    servo_move(5, 150, 100);


    return 0;
```

```
}



/**
  * This function takes the servo number and the position, and writes the values in
  * appropriate address for the FPGA
  * @param test_base                base pointer for servos
  * @param servo_number          servo number to manipulate
  * @param position               new postion in degree (0 .. 180)
  * @param speed                    speed to move in degree / 20ms
  */
void servo_move(unsigned char servo_number, unsigned char position, unsigned
char speed) {


      /* writeValue bits 0..7       position
       *                 bits 8..15      speed
       *                 bits 16..31    all 0
       */
    unsigned int writeValue = ((unsigned int) position) + (((unsigned int) speed) << 8);
    positions[servo_number] = position;
```

```
switch (servo_number) {

        case 1:   //Base

                REG_WRITE(gServos.test_base, Base_OFFSET, writeValue);

                break;


            case 2:   //Bicep

                REG_WRITE(gServos.test_base, Bicep_OFFSET, writeValue);

                break;


        case 3:   //Elbow

                REG_WRITE(gServos.test_base, Elbow_OFFSET, writeValue);

                break;


            case 4:   //Wrist

                REG_WRITE(gServos.test_base, Wrist_OFFSET, writeValue);

                break;


            case 5:   //Gripper

                REG_WRITE(gServos.test_base, Gripper_OFFSET, writeValue);

                break;
```

```
                default:
                    break;
        }
}



/**
  * Deinitialize Servos
  */
void servo_release(){
        // Releasing the mapping in memory
        munmap((void *)gServos.test_base, gServos.map_len);
        close(gServos.fd);
}


unsigned int bound(int degree){
    if(degree < 60){
        return 60;
    }
    if(degree > 240){
        return 240;
```

```
    }

    return degree;

}


void main(){

    int btnFile = acquire_btn_file();

    int accelFile = acquire_accel_file();

    int fail = servo_init();


    if(fail != 0 || btnFile == -1 || accelFile == -1){

        printf("Couldn't start\n %d %d %d", fail, btnFile, accelFile);

        exit(1);

    }


    int speed = 1;

    int pressed[512]; //Space for all the button codes

    while(1){

        struct btn_event press = getBtnEvt(btnFile);

        struct accel_event accel = getAccelEvt(accelFile);

        //printf("Evt value: %d\n", press.evtValue);

        pressed[press.evtCode] = press.evtValue;
```

```
//We will divide the accel delta by 2 to get a degree change. This needs to
//be bounded by 60 and 240 (after it is combined with the position in
//degrees for a given servo)
int degreeChange = accel.accel / 2;
if(accel.evtCode != WIIMOTE_EVT0_ACCEL_X){
   continue;
}
if(pressed[WIIMOTE_2] == 1){
   servo_move(1, bound(positions[1] + degreeChange), speed);
   printf("%u %d\n", POS(Base),    degreeChange);
}
if(pressed[WIIMOTE_1] == 1){
   servo_move(2, bound(positions[2] + degreeChange), speed);
   printf("%u\n", bound(POS(Bicep) + degreeChange));
}
if(pressed[WIIMOTE_B] == 1){
   servo_move(3, bound(positions[3] + degreeChange), speed);
   printf("%u\n", bound(POS(Elbow) + degreeChange));
}
if(pressed[WIIMOTE_A] == 1){
```

```
        servo_move(4, bound(positions[4] + degreeChange), speed);
        printf("%u\n", bound(POS(Wrist) + degreeChange));
    }
    if(pressed[WIIMOTE_HOME] == 1){
        servo_move(5, bound(positions[5] + degreeChange), speed);
        printf("%u\n", POS(Gripper));
    }
    if(pressed[WIIMOTE_PLUS] == 1){
servo_move(1, 150, 100);
servo_move(2, 150, 100);
servo_move(3, 150, 100);
servo_move(4, 150, 100);
servo_move(5, 150, 100);
    }
    if(press.evtCode == WIIMOTE_UP_ARROW){
        speed++;
    } else if(press.evtCode == WIIMOTE_DOWN_ARROW){
        speed--;
        if(speed < 1){
speed = 1;
        }
```

```
        } else if(press.evtCode == WIIMOTE_MINUS){
            servo_release();
            close(accelFile);
            close(btnFile);
            break;
        }
    }
}

/**
 * Template for Servo Control from FPGA with Software Controlled Speed
 *
 */
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <time.h>
#include <unistd.h>

#define BASE_ADDRESS 0x400D0000
```

```
//Servo motor offsets

#define Base_OFFSET 0x100

#define Bicep_OFFSET 0x104

#define Elbow_OFFSET 0x108

#define Wrist_OFFSET 0x10C

#define Gripper_OFFSET 0x110


#define REG_WRITE(addr, off, val) (*(volatile int*)(addr+off)=(val))

#define REG_READ(addr, off) (*(volatile int*)(addr+off))


/**
  * data structure for servo instance
  */
typedef struct {
      unsigned char *test_base; /// base address of mapped virtual space
      int fd;                          /// file desrcriptor for memory map
      int map_len;                     /// size of mapping window

} tServo;
```

```
/**
  * global variable for all servos
  */
tServo gServos;


/**
  * Initialize servos
  * @return 0 upon success, 1 otherwise
  */
int servo_init() {

      //Open the file regarding memory mapped IO to write values for the FPGA
      gServos.fd = open( "/dev/mem", O_RDWR);

      unsigned long int PhysicalAddress = BASE_ADDRESS;
      gServos.map_len= 0xFF;    //size of mapping window

      // map physical memory startin at BASE_ADDRESS into own virtual memory
      gServos.test_base = (unsigned char*)mmap(NULL, gServos.map_len,
PROT_READ | PROT_WRITE, MAP_SHARED, gServos.fd, (off_t)PhysicalAddress);
```

```
    // did it work?
    if(gServos.test_base == MAP_FAILED)  {
        perror("Mapping memory for absolute memory access failed -- Test Try\n");
        return 1;
    }

    //Initialize all servo motors
    // I assume this is the "sleep" position
    REG_WRITE(gServos.test_base, Base_OFFSET, 150);
    REG_WRITE(gServos.test_base, Bicep_OFFSET, 190);
    REG_WRITE(gServos.test_base, Elbow_OFFSET, 190);
    REG_WRITE(gServos.test_base, Wrist_OFFSET, 100);
    REG_WRITE(gServos.test_base, Gripper_OFFSET, 150);

    return 0;
}

/**
  * This function takes the servo number and the position, and writes the values in
  * appropriate address for the FPGA
```

```
 * @param test_base              base pointer for servos
 * @param servo_number          servo number to manipulate
 * @param position              new postion
 */
void servo_move(int servo_number, int position) {
    switch (servo_number) {
            case 1:   //Base
                REG_WRITE(gServos.test_base, Base_OFFSET, position);
                break;

            case 2:   //Bicep
                REG_WRITE(gServos.test_base, Bicep_OFFSET, position);
                break;

            case 3:   //Elbow
                REG_WRITE(gServos.test_base, Elbow_OFFSET, position);
                break;

            case 4:   //Wrist
                REG_WRITE(gServos.test_base, Wrist_OFFSET, position);
                break;
```

```
                case 5:    //Gripper
                    REG_WRITE(gServos.test_base, Gripper_OFFSET, position);
                    break;

                default:
                    break;
        }
}


/**
  *Speed is in degrees per second
  */
void servo_moveHelper(int servo_number, int end, int speed){
    int start = REG_READ(gServos.test_base, 0x96 + 0x4 * servo_number);
    int pos;
    if(end < start){
        speed *= -1;
    }
    for(pos = start; (pos - start) < 0 == (start - end) < 0; pos += speed / 5){
```

```
        servo_move(servo_number, pos);
        usleep(200000);
    }
    servo_move(servo_number, end);
}

/**
  * Deinitialize Servos
  */
void servo_release(){
      // Releasing the mapping in memory
      munmap((void *)gServos.test_base, gServos.map_len);
      close(gServos.fd);
}

int main()
{
      //Declarations and initialization
      int servo_number = 0;
      int position = 0;
```

```
int speed = 10;


printf("\n----------    Robot TESTING    ----------\n\n");


/* initialize servos */
if (servo_init() != 0) {
    return -1; // exit if init fails
}


do {
    printf("Enter servo number (1-5) or enter 0 to exit:\n");
    scanf("%d", &servo_number); //Take the servo number from user


    if (servo_number != 0) {


        printf("Enter position (60 - 240): Speed in degrees/s:\n");
        scanf("%d %d", &position, &speed); //Take the position from user


        servo_moveHelper(servo_number, position, speed); //The selected
servo will move to the desired position
    }
```

```
    } while( servo_number != 0 ); // repeat while valid servo number given



    /* deinitialize servos */

    servo_release();



    return 0;

}
```