Embedded Des Enabling Robotics
Lab 10 report

Shiyu Wang
Hao Jiang
Fall 2016
10th December

10.1

This design is the essentially same as the design we developed in Lab 9. However, instead of counters for controlling the duty cycle, it has constants (which will be mapped to MMRs. It takes 5 uint8 inputs as the position of 5 servo motors and generates the 5 PWM signals. We simply put a constant speed in the program, and tested if the program work with these speed.

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. We tested on different part using the same speed, and we found it worked. The program is attached.

10.2

We downloaded and opened the template design from Blackboard. We compile the design using HDL workflow advisor. In the *Set Target Interface*, we used the AXI4-Lite as inputs. After comparing the two designs, we found 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

This program will grab a box and throw it away. Moving angles of all the parts were tested before concluded. They are also recorded. Both buttons and acceleration values can be read. The program is attached in the appendix as well as in Blackboard.

Grip: (position)
Speed: 25
Base-1: 145
Bicep-2: 100
Elbow: 155
Wrist: 155
Gripper: 100

Throw: (position)
Bicep: 200
Elbow: 200
Gripper: 240

Appendix

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

```
#include <time.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))

/**
 * 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;


/**
 * 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 = ((speed << 8) | 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);
}


int main()
{
```

```
	//Declarations and initialization
	int servo_number = 0;
	int position = 0;
int speed = 0;

	printf("\n----------  ATTENTION ROBOT WILL BE MOVING!  ----------\n\n");
	printf("Please ensure robot power is OFF. Hold it in middle position. Then, turn it on.\n");
	sleep(1);

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

	sleep(2);
	servo_move(1, 145, 25);
	usleep(50000);
	servo_move(2, 90, 20);
	servo_move(3, 145, 25);
	servo_move(4, 145, 25);
	usleep(100000);
	servo_move(5, 100, 25);
	sleep(1);
	servo_move(2,200,50);
	servo_move(3, 200, 50);
	usleep(495000);
	servo_move(5, 240, 50);

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

	// if valid servo number
		if (servo_number != 0) {

			printf("Enter speed (1-50):\n");
			scanf("%d", &speed);  //Take the speed from user

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

			//The selected servo will move to the desired position
```

```
                        servo_move(servo_number, position, speed);
                }
        } while( servo_number != 0 ); // repeat while valid servo number given */

        /* deinitialize servos */
        servo_release();

        return 0;
}
/**
* 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;
}
```