**Lab Assignment 10**

Hao Jiang

Shiyu Wang

**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.

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;

}