CS223 Laboratory Assignment 4 Using the Step Motor

Lab dates and times:

Section 1:	18.11.2019	Monday	08:40-12:25
Section 2:	19.11.2019	Tuesday	08:40-12:25
Section 3:	18.11.2019	Monday	13:40-17:25
Section 4:	19.11.2019	Tuesday	13:40-17:25
Section 5:	21.11.2019	Thursday	08:40-12:25
Section 6:	22.11.2019	Friday	08:40-12:25

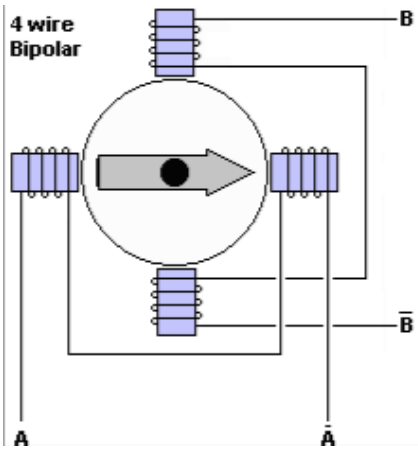
Location: EA Z04 (in the EA building, straight ahead past the elevators)

Groups: Each student will do the lab individually. Group size = 1

Lab Assignment

In this lab we are going to investigate the use of a step-motor. A step motor (or stepper motor) is a brushless DC (direct current) electric motor that divides a full rotation into a number of equal steps. The step motor has permanent magnet rotor (rotating piece, oftentimes the center piece). And some electromagnets are placed at fixed intervals on the stator (stationary piece). The step motor on the experiment set is a permanent magnet, parallel coil step motor. Step motors operate by electrical pulses; when an electrical pulse is applied to its coils, step motor will turn one step. Each step can be a small angle such as 3.75 degree or 7.5 degree etc. They can be controlled digitally and therefore can be used in plotters, printers, some medical instruments, robotics etc. In order to turn the motor one step a current **pulse should be applied**.

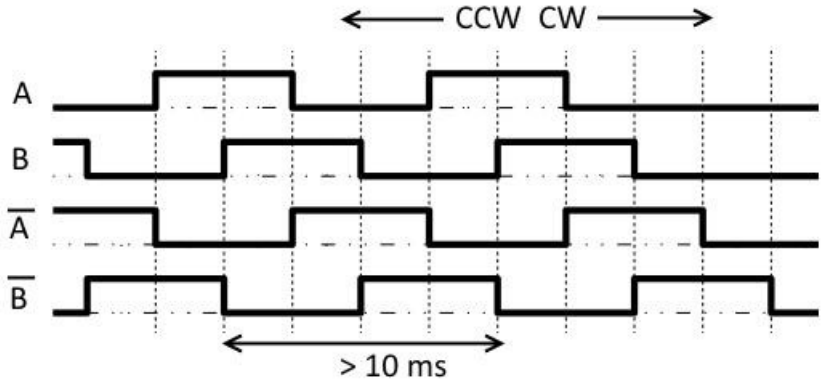
An explanatory figure with 4 stators is given in the figure below. Each step in this step-motor is 45 degrees. **Each time** you apply a **pulse** to one of the **inputs**, the corresponding stator piece will attract the rotor piece to itself and will keep it there as long as the signal stays on. If you apply a pulse to the adjacent input the adjacent stator piece will attract the rotor piece to itself, if you continue with generating pulses continuously than the rotor will have a continuous motion.



The frequency of the pulses determines the speed of the step motor. In the step motor you are going to use, the period of application of the input pulses should be more than 10 milliseconds. The step motor requires 175 mA current which cannot be supplied by the BASYS3 card (Any FPGA pin typically can give maximum of 10-15 mA). Therefore, instead

of connecting the FPGA output ports to the stepper motor, the step motor is driven by a driver circuit containing more powerful transistors (check lab Demo section for connection details). The BASYS3 board outputs can then be connected to inputs of the driver, and then driver outputs to stepper motor inputs.

In order to operate the step motor, following sequence of inputs should be applied to its four drivers periodically. For clockwise rotation (A,B,Ab,Bb) = 1100, 0110, 0011, 1001 and for counter clockwise rotation, apply same sequence in reverse direction. After application of each pulse on 4-bit input, the motor will turn one step.



Note: You need to take one grey thick wire which will be connected to the step motor's part, and you should connect it to Basys3 FPGA board.

Preliminary Report (40 points)

All students should be prepared to submit their Preliminary Design Report at the start of their lab section time. Your report should contain the following items, each starting at the top of a new page:

1. A cover page which include the following: course name and code number, the number of the lab, your name and student ID, date, number of your trainer pack.
2. The FSM for the driver which will drive the step motor. This includes state transition diagram, next state table, output table, state encoding table, output encoding table, reduced next state logic truth table, reduced output logic truth table. And finally simplified Boolean equations for both. Clearly state the inputs and outputs to your FSM. Use binary encoding in your solution.
3. The **SystemVerilog code** for the driver. **Testbench** for your SystemVerilog code.
4. Prepare the SystemVerilog codes for the step motor simulate driving a car. Assume that the step motor is the wheels of your car. Your car operates as follows:
    a. Use a switch on the BASYS board to Start or stop the car. When the car starts, step-motor turns clockwise.
    b. Use 2 switches on the BASYS to specify the speed of the car. your car can have four different speeds: 3(b11) stop, 2(b10) slowest, 1(b01) medium, 0(b00) fastest.
    c. Use another switch to reverse the car. (i.e your step motor should turn counter-clockwise).
    d. Show the speed of your car on the 4-digit 7-segment display (revolutions per minute for the speeds you've chosen). You can use the SystemVerilog code for 7-segment display in Unilica. You can find related FPGA pins for 7-segments in

BASYS3 manual in Figure-16. Do not include this code as a part of your code for MOSS test.

e. Use the push buttons for signalling right or left. The signals will be the LEDs on your BASYS3 board.

5. **Prepare the SystemVerilog codes** for the step motor simulate driving a car.
6. **Prepare testbench** for testing your design of the car.

Note: Remember that the step-motor maximum speed is 40 steps/second.

IMPORTANT NOTE: The SystemVerilog modules are code, and as with any program code, they should be well-structured, well-commented, use meaningful identifiers, use white space and indentation as appropriate to facilitate understanding, and in summary, should be self- documenting. If code is difficult to understand, it is not self-documenting! All pages in the report, with the possible exception of a hand-drawn logic diagram, should be printed. You should make a photocopy of the Preliminary Design Report, for you to use during the lab.

Lab Demo (60 points)

You should come to lab fully prepared. You are advised to arrive with the problem above fully implemented and tested and ready to work. Your grade will be based on the following:

1. You will be asked to do simulation, using the testbench modules you created, for SystemVerilog modules in Preliminary work part (b) and (c).

2. **Use the flat connecter and gray parallel cable to connect the BASYS3 board outputs to step motor input terminal**. Check the step motor schematic (available on Unilica) for pinout. **On BASYS3, the 4 signals (A,B,Ab,Bb) go to FPGA pins (A14, B15, A16, B16) respectively**. Implement each of the SystemVerilog codes you have designed in the preliminary work. Do the Xilinx design flow (Synthesize, Implement, Create bitstream, and Program the device) to realize your design on the Digilent FPGA board. When you are convinced that it works correctly, show the physical implementation results to the TA, who will grade each demo of your designs.

3. Finally, the TA will ask questions to check your knowledge and understanding of the project and SystemVerilog, and you will receive a grade according to your answers.

4. Upload your codes (without the 7-segment codes) before the upload slot is closed.

Submit your code for MOSS similarity testing

Finally, when you are done and before leaving the lab, you need to upload the file *StudentID_SVerilog.txt* created in the Implementation with FPGA part. Be sure that the file contains exactly and only the codes which are specifically detailed above. Don't include the codes which are given to you. If you have multiple files, just copy and paste them in order, one after another inside text file. Check the specifications! Even if you didn't finish, or didn't get the SystemVerilog part working, you must submit your code to the Unilica Assignment for similarity checking. Your codes will be compared against all the other codes in all sections of the class, by the MOSS program, to determine how similar it is (as an indication of plagiarism). So be sure that the code you submit is code that you actually wrote yourself! All students must upload their code to the 'Unilica>Assignment' specific for your section. Check submission time and don't miss it before leaving the lab. After taking a backup of your work, don't forget to delete it from computer. Because students of other sections will work with your system too.

Clean Up!

1. Clean up your lab station, and return all the parts, wires, the Beti trainer board, etc. Leave your lab workstation.
2. CONGRATULATIONS! You are finished with this lab and are one step closer to becoming a computer engineer.

NOTES

- Advance work on this lab, and all labs, is strongly suggested.
- Be sure to read and follow the Policies for CS223 labs, posted in Unilica.

LAB POLICIES

1. There are three computers in each row in the lab. Don't use middle computers, unless you are allowed by lab supervisor.
2. You borrow a Lab-board containing the development board, connectors, etc. in the beginning. The lab supervisor takes your signature. When you are done, return it to her, otherwise you will be responsible and lose points.
3. Each Lab-board has a number. You must always use the same trainer board pack throughout the semester.
4. You must be in the lab, working on the lab, from the time lab starts until you finish and leave. (Bathroom and snack breaks are the exception to this rule). Absence from the lab, at any time, is counted as absence from the whole lab that day.
5. No cell phone usage during lab. Tell friends not to call during the lab hours--you are busy learning how digital circuits work !
6. Internet usage is permitted only to lab-related technical sites. No Facebook, Twitter, email, news, video games, etc--you are busy learning how digital circuits work !
7. If you come to lab later than 20 minutes, you will lose that session completely.
8. When you are done, DO NOT return IC parts into the IC boxes, where you've taken them first. Just put them inside your lab pack box. Lab coordinator will check and return them later.

SevSeg_4digit.sv:

```
`timescale 1ns / 1ps
/     LED positions inside 7-segment
/     A
//  F B
/     G
//  E C
//    D     DP
/     digit positions on Basys3 :
/     in3(left), in2, in1, in0(right)

module SevSeg_4digit(
input clk,
input [3:0] in0, in1, in2, in3,     //  4  values  for  4  digits  (decimal
value)
output a, b, c, d, e, f, g, dp, //individual LED output for the 7-segment
along with the digital point
output [3:0] an   // anode: 4-bit enable signal (active low)
);

/     divide system clock (100Mhz for Basys3) by 2^N using a counter, which
allows us to multiplex at lower speed
localparam N = 18;
logic [N-1:0] count = {N{1'b0}}; //initial value always@ (posedge clk)
count <= count + 1;
logic [3:0]digit_val; // 7-bit register to hold the current data on output
logic [3:0]digit_en; //register for enable vector always_comb

begin
digit_en = 4'b1111; //default
digit_val = in0; //default


case(count[N-1:N-2]) //using only the 2 MSB's of the counter
2'b00 : //select first 7Seg.
begin
digit_val = in0;
digit_en = 4'b1110;
end

2'b01: //select second 7Seg.
begin
digit_val = in1;
digit_en = 4'b1101;
end

2'b10: //select third 7Seg.
begin
digit_val = in2;
digit_en = 4'b1011;
end

2'b11: //select forth 7Seg.
begin
digit_val = in3;
digit_en = 4'b0111;
end
endcase
end
```

//continues on next page

```
//Convert digit number to LED vector. LEDs are active low.

logic [6:0] sseg_LEDs;
always_comb
begin
sseg_LEDs = 7'b1111111; //default
case(digit_val)
4'd0 : sseg_LEDs = 7'b1000000; //to display 0
4'd1 : sseg_LEDs = 7'b1111001; //to display 1
4'd2 : sseg_LEDs = 7'b0100100; //to display 2
4'd3 : sseg_LEDs = 7'b0110000; //to display 3
4'd4 : sseg_LEDs = 7'b0011001; //to display 4
4'd5 : sseg_LEDs = 7'b0010010; //to display 5
4'd6 : sseg_LEDs = 7'b0000010; //to display 6
4'd7 : sseg_LEDs = 7'b1111000; //to display 7
4'd8 : sseg_LEDs = 7'b0000000; //to display 8
4'd9 : sseg_LEDs = 7'b0010000; //to display 9
default : sseg_LEDs = 7'b0111111; //dash
endcase
end

assign an = digit_en;
assign {g, f, e, d, c, b, a} = sseg_LEDs;
assign dp = 1'b1; //turn dp off
endmodule
```