RC Servo and Stepper Motor Control using Verilog HDL

Kingshuk Chowdhury #1, Rumana Hossain*2, Saif Ahmed #3, Iqbal Rahman Rokon #4

1,2 Undergraduate Student, Department of Electrical and Computer Engineering, North South University, Bashundhara, Dhaka , Bangladesh

3 Lab Officer, Department of Electrical and Computer Engineering, North South University, Bashundhara, Dhaka , Bangladesh

4 Senior Lecturer, Department of Electrical and Computer Engineering, North South University, Bashundhara, Dhaka , Bangladesh

Abstract

In this paper, an FPGA model of RC servo and Stepper motor control system was built. This system consists of EP2C5T144C8 Cyclone II Mini Board, 9 grams micro servo motor, 28BYJ-48 Stepper Motor, ULN2003 motor driver. This system operates 3 functions parallelly, a simple binary count with the 3 LEDs on the board, from 0 to 7, controlling a RC servo, where the position is dependent on the value in the previous counter. The servo starts when the counter value goes from 000 to 001 for the first time. After counting from 0 to 7 for the first time, the servo goes to its initial position when the counter goes to 000 for all the time and controlling a Stepper motor, where the speed is dependent again on the previous counter. The proposed model is built using Verilog HDL, simulated using Modelsim Altera Starter Edition 10.1d and synthesized using Xilinx ISE 9.2i package, and will be implemented using Cyclone II FPGA Mini Board kit. Implementation and Simulation behavioral model results show that the proposed model satisfies the specified operational requirements.

Keywords— *Servo motor, Stepper motor, Motor Driver, RTL, Verilog*

I. INTRODUCTION

The stepper motor is an electromechanical device; it converts electrical power into mechanical power. Also, it is a brushless, synchronous electric motor that can divide a full rotation into an expansive number of steps. The motor’s position can be controlled accurately without any feedback mechanism, as long as the motor is carefully sized for the application. Stepper motors are similar to switched reluctance motors.

Stepping motors, as discrete state devices, are extremely valuable when precision position or velocity is needed. Being discrete state devices, these motors are particularly well suited to digital system control. Although microcontrollers are suited for this application, programmable logic devices (PLDs) hold several advantages. Being a far more energy efficient and flexible system, a PLD also provides easy on chip integration of communication standards, parallel performance, and high pin count for multiple motor controls. Here, we cover some basic considerations for programming a PLD with very high speed integrated circuit hardware description language (VHDL) to function as a stepping motor controller.

Servo motors (or servos) are self-contained electric devices that rotate or push parts of a machine with great precision. Servos are found in many places: from toys to home electronics to cars and airplanes. If you have a radio-controlled model car, airplane, or helicopter, you are using at least a few servos. In a model car or aircraft, servos move levers back and forth to control steering or adjust wing surfaces. By rotating a shaft connected to the engine throttle, a servo regulates the speed of a fuel-powered car or aircraft. Servos also appear behind the scenes in devices we use every day. Electronic devices such as DVD and Blu-ray DiscTM players use servos to extend or retract the disc trays. In 21st-century automobiles, servos manage the car's speed: The gas pedal, similar to the volume control on a radio, sends an electrical signal that tells the car's computer how far down it is pressed. The car's computer calculates that information and other data from other sensors and sends a signal to the servo attached to the throttle to adjust the engine speed. Commercial aircraft use servos and a related hydraulic technology to push and pull just about everything in the plane.

RC Servo Motor is a type of motor whose position can be controlled externally. They are small, compact and quite inexpensive. So they are mostly used in robotic projects. The axis rotational angle is limited to about 270 degrees.

The servo expects a pulse at every 20 ms in order to gain correct information about the angle. [1] The width of the pulse dictates the range of the servo's angular motion. i.e we can set the servo to one end position by sending 1 ms pulses and set it to the other position sending 2 ms pulses. Sending 1.5 ms pulse sets the servo motor to the center position. The PWM module includes 4 inputs and 1 output.

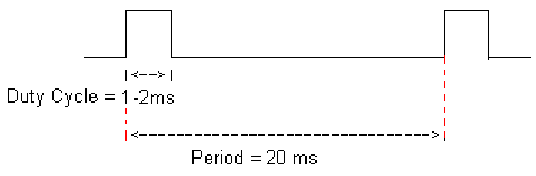

Fig 1: expected input for servo motor

II. INTERNAL DESIGN DETAILS

The following is a breakdown of the internal design details of the hardware simulation

A. Block Diagram and Design

Y

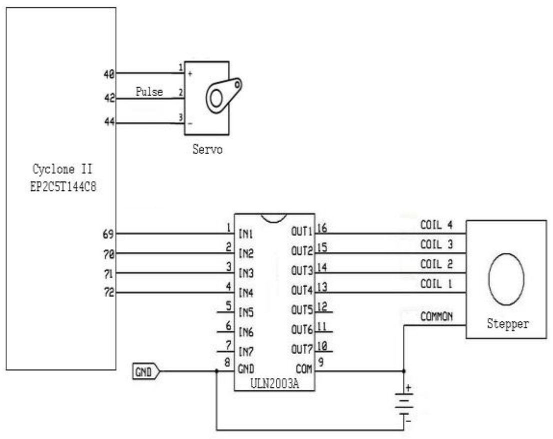

Fig 2: Block Diagram of the System

First of all, the servo motor is connected to the EP2C5T144C8 Cyclone II FPGA Mini Board.[2] The positive (power) pin of the servo motor is connected to the pin 40 of the FPGA, the control pin or signal pin is connected to the pin 42 of the FPGA and the ground pin is connected the pin 44 of the FPGA. Then the motor driver ULN2003A connected to the FPGA. IN1 pin of motor driver is connected to the pin 69 of the FPGA, IN2 is connected to the pin 70, IN3 is connected to pin 71, IN4 is connected to the pin 72 and the GND pin is grounded. After this the connection between stepper motor and the motor driver are given. OUT1, OUT2, OUT3, OUT4 pin of the motor driver is connected respectively to the pin 4, pin3, pin 2, and pin 1 of the stepper motor and COM pin of the motor driver and Common pin of the stepper motor is connected to the power supply. Pins 3, 7, 9 are for 3 LEDs, pin 144 is for reset, pin 17 for clock and pin 135 for enable. Then the required program is loaded to FPGA to do expected operations.

B. Inputs and Outputs

We have used to 4 inputs to the desired outputs. The inputs are clock for clock pulses, reset for taking the counter to its initial condition, enable for enabling the counter for counting and start for enabling the counter which is used for stepper motor control. The outputs are, led for providing the output for 3 bit counter, servoPin for the servo motor control output, stepperPins for the stepper motor control output.

C. Design Hierarchy

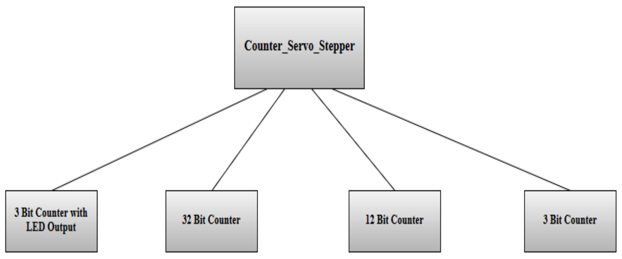

Fig 3: Design Hierarchy

From the figure it can be seen that we have followed the top-down methodology. We first have specified the functionality of the Counter_Servo_Stepper, which is the top-level block. Thus, we broke the bigger block into smaller building sub-blocks. We implemented the LED operation using a 3 bit counter, servo motor control using a 32 bit counter and a 12 bit counter and finally a 3 bit counter for stepper motor control.

III. DETAILED REQUIREMENTS

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

A. List of Functions

Our system will able to perform 3 functions in parallel-

•A simple binary count with the 3 LEDs on the board, from 0 to 7.

•Controlling a RC Servo, where the position is dependent on the value in the previous counter. The servo starts when the counter value goes from 000 to 001 for the first time. After counting from 0 to 7 for the first time, the servo goes to its initial position when the counter goes to 000 for all the time.

•Controlling a Stepper motor, where the speed is dependent again on the previous counter.

B. Hardware Requirements

•EP2C5T144C8 Cyclone II FPGA Mini Board
•9 Grams Micro Servo Motor
•28BYJ-48 Stepper Motor
•ULN2003 Motor Driver

C. Software Requirements

•Modelsim Altera Starter Edition 10.1d
•Xilinx 9.2i

IV. HARDWARE DESCRIPTION

A. *EP2C5T144C8 Cyclone II FPGA Mini Board*

The Altera EP2C5T144C8 Cyclone II FPGA that the board accompanies is a more established gadget, yet is still broadly utilized, and is prepared to do some exceptional applications. It has 4068 logic components, 26 4k RAM blocks giving a sum of 119,898 bits, 13 multipliers, two PLLs, and 89 I/Os. Greatest clock frequency is 300 MHz. The board is fitted with an EPCS4 flash design memory chip, and a 50 MHz oscillator. A large portion of the I/Os are conveyed out to four 2x14 way headers, which may be associated with outer hardware by means of wire jumpers, or PCBs fitted with coordinating attachments. Three LEDs are associated with pins 3, 7 and 9, and a push-catch is joined with pin 144. You may be confused by the absence of a pull-up resistor, yet it is conceivable to empower a powerless pull-up on any pin utilized as an input. The board has two 10-way headers, one for JTAG and one for Advanced Serial programming, to which the USB Blaster can be joined.

If power is applied to the board the power LED should come on, and the three LEDs should flash at 1 Hz. A simple design has been saved in the configuration flash memory, and is loaded into the FPGA when power is applied. Typically, designs are loaded directly into the FPGA during development via JTAG, and only saved to configuration memory when they are working properly, using Advanced Serial programming.

Specifications:
•Number of Logic Elements: 4608
•Number of Logic Array Blocks - LABs: 288
•Number of I/Os: 89 I/O
•Operating Supply Voltage: 1.15 V to 1.25 V
•Maximum Operating Temperature: + 70 C
•Mounting Style: SMD/SMT
•Package / Case: TQFP-144
•Brand: Altera Corporation
•Maximum Operating Frequency: 260 MHz
•Minimum Operating Temperature: 0◦C
•Packaging: Tray
•Series: Cyclone II EP2C5

B. *9 Grams Micro Servo Motor*

Servo motors (or servos) are independent electric gadgets that rotate or push parts of a machine with awesome accuracy. Servos are found in numerous spots: from toys to home hardware to autos and planes. A servo engine is a dc, air conditioning, or brushless dc engine joined with a position detecting gadget (e.g. a computerized decoder). A three-wire DC servo engine joins a DC engine, an apparatus train; breaking point stops past which the pole can't turn a potentiometer for position input and a coordinated circuit for position control. Of the three wires jutting from the motor packaging, one is for power, one is for ground, and one is a control input where a pulse-width signals to what position the motor ought to servo. The 9 grams servo engine is a small scale type. It is impractical to quantify the motor resistance in light of the fact that it is joined with the driver however we can measure the servo current while it's running. It has plastic bushing, 25cm wire, core-less motor, Servo arms and screw included.[3][8]

Specifications:
•Size: 21x12x22 mm / 0.74x0.42x0.78 in
•Voltage: 3v ~ 6v
•Weight: 9g / 0.39oz
•Speed: 0.12 sec/60(4.8V)
•Torque : 1.6kg-cm
•Working Temp : -30C~60C
•Gears: Nylon White type

C. *28BYJ-48 Stepper Motor*

Stepper motors are DC motors that move in discrete steps. They have numerous loops that are sorted out in gatherings called "phases". By invigorating every phase in grouping, the motor will pivot with extra special care. The 28BYJ-48 is a little, cheap, 5 volt geared stepping motors. These stepping motors are clearly generally used to control things like computerized blinds, A/C units and are mass created. Because of the gear reduction proportion of *approximately* 64:1 it offers good torque for its size at rates of around 15 rotations per minute (RPM). "28BYJ-48" does not distinguish a particular model. There are a few unique adaptations connected with this part number. Some portion of the refinement is made by the voltage rating connected with the engine, either "5V" or "12V". In any case, notwithstanding for specific voltage ratings, there are diverse models accessible with, for instance, distinctive winding resistances. There may likewise be models with distinctive gearing. The engine has 4 coils of wire that are controlled in a grouping to make the magnetic motor shaft turn. At the point when utilizing the full-step system, 2 of the 4 coils are powered at every step. The 28BYH-48 datasheet determines that the favored technique for driving this stepper is utilizing the half-step strategy, where we first power coil 1 only, then coil 1 and 2 together, then coil 2 just so on… With 4 curls, this implies 8 distinct signals.[4][7]

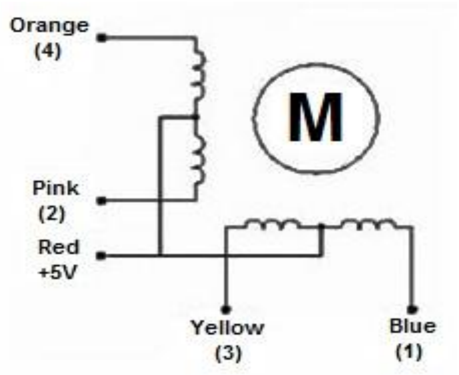

Fig 4: Stepper Motor Circuit

Specifications
•Rated voltage: 5VDC
•Number of Phase: 4
•Speed Variation Ratio: 1/64
•Stride Angle: 5.625° /64
•Frequency: 100Hz
•DC resistance: 50Ω±7%(25℃)
•Idle In-traction Frequency: > 600Hz
•Idle Out-traction Frequency: > 1000Hz
•In-traction Torque: >34.3mN.m(120Hz)
•Self-positioning Torque: >34.3mN.m
•Friction torque: 600-1200 gf.cm
•Pull in torque: 300 gf.cm
•Insulated resistance: >10MΩ(500V)
•Insulated electricity power: 600VAC/1mA/1s
•Rise in Temperature: <40K(120Hz)
•Noise: <35dB(120Hz,No load,10cm)

Table I : Half Step Switching Sequence of the Stepper Motor

Lead Wire Colour	Clockwise Direction (1-2 Phase)							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
4 Orange	-	-						-
3 Yellow		-	-	-				
2 Pink				-	-	-		
1 Blue						-	-	-

D. ULN2003 Motor Driver

The ULN2003 is a solid high voltage and high current Darlington transistor arrays. It comprises of seven NPN Darlington sets that component high-voltage output with common-cathode clamp diode for exchanging inductive burdens. The authority current rating of a solitary Darlington pair is 500mA. The Darlington sets may be paralleled for higher current ability. Applications incorporate relay drivers, hammer drivers, lamp drivers, show drivers (LED gas release), line drivers, and logic buffers. The ULN2003 has a 2.7kΩ series base resistor for each Darlington pair for operation straightforwardly with TTL or 5V CMOS gadgets.[5]
Specifications

- 500mA rated collector current (Single output)
- High-voltage outputs: 50V
- Inputs compatible with various types of logic.
- Relay driver application

V. SIMULATION AND SYNTHESIS

A. Test Environment

Firstly, we compiled the RTL code and the test bench together and found no error. After the successful compilation the RTL code is simulated using test bench. The compilation and simulation is done successfully using Modelism Altera Starter Edition 10.1d. The RTL code is then synthesized using Xilinx 9.2i and the desired output is obtained successfully without any error.

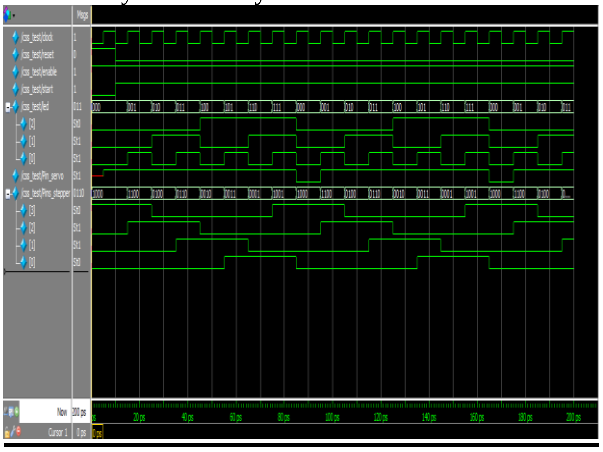
Fig 5: Waveform of output simulation

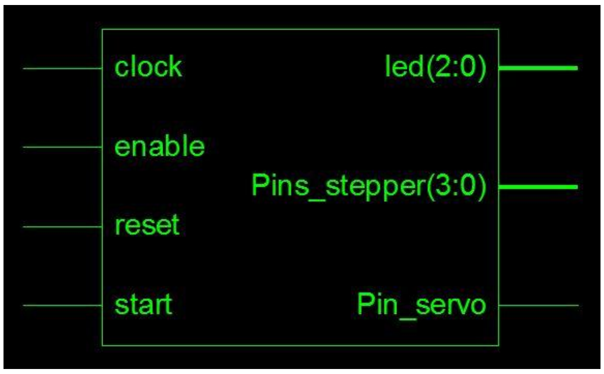

Fig 6: RTL schematic

B. Simulation Code

```
Transcript
vlog -reportprogress 300 -work work {C:/altera/13.0sp1/modelsim_ase/win32aloem/RC_Servo & Stepper/SERVO_STEPPER.v} {C:/altera/13.0sp1/modelsim_ase/win32aloem/RC_Servo & Stepper/SERVO_STEPPER_TB.v}
# Model Technology ModelSim ALTERA vlog 10.1d Compiler 2012.11 Nov  2 2012
# -- Compiling module Counter_Servo_Stepper
# -- Compiling module css_test
#
# Top level modules:
#       css_test
vsim -gui work.css_test
```

```
# vsim -gui work.css_test
# Loading work.css_test
# Loading work.Counter_Servo_Stepper
add wave sim:/css_test/*
# ** Warning: (vsim-WLF-5000) WLF file currently in use: vsim.wlf
#
#          File in use by: Engr. Kingshuk  Hostname: CYBERTRON  ProcessID: 4600
#
#              Attempting to use alternate WLF file "./wlftm33k47".
# ** Warning: (vsim-WLF-5001) Could not open WLF file: vsim.wlf
#
#          Using alternate file: ./wlftm33k47
#
run -all
#                0c=0,r=1, e=1,l=000, servo=x,  strt=0, stepper=1000
#                5c=1,r=1, e=1,l=000, servo=1,  strt=0, stepper=1000
#                10c=0,r=0, e=1,l=000, servo=1, strt=1, stepper=1000
#                15c=1,r=0, e=1,l=001, servo=1, strt=1, stepper=1100
#                20c=0,r=0, e=1,l=001, servo=1, strt=1, stepper=1100
#                25c=1,r=0, e=1,l=010, servo=1, strt=1, stepper=0100
#                30c=0,r=0, e=1,l=010, servo=1, strt=1, stepper=0100
#                35c=1,r=0, e=1,l=011, servo=1, strt=1, stepper=0110
#                40c=0,r=0, e=1,l=011, servo=1, strt=1, stepper=0110
#                45c=1,r=0, e=1,l=100, servo=1, strt=1, stepper=0010
#                50c=0,r=0, e=1,l=100, servo=1, strt=1, stepper=0010
#                55c=1,r=0, e=1,l=101, servo=1, strt=1, stepper=0011
#                60c=0,r=0, e=1,l=101, servo=1, strt=1, stepper=0011
#                65c=1,r=0, e=1,l=110, servo=1, strt=1, stepper=0001
#                70c=0,r=0, e=1,l=110, servo=1, strt=1, stepper=0001
#                75c=1,r=0, e=1,l=111, servo=1, strt=1, stepper=1001
#                80c=0,r=0, e=1,l=111, servo=1, strt=1, stepper=1001
#                85c=1,r=0, e=1,l=000, servo=0, strt=1, stepper=1000
#                90c=0,r=0, e=1,l=000, servo=0, strt=1, stepper=1000
#                95c=1,r=0, e=1,l=001, servo=1, strt=1, stepper=1100
#               100c=0,r=0, e=1,l=001, servo=1, strt=1, stepper=1100
#               105c=1,r=0, e=1,l=010, servo=1, strt=1, stepper=0100
#               110c=0,r=0, e=1,l=010, servo=1, strt=1, stepper=0100
#               115c=1,r=0, e=1,l=011, servo=1, strt=1, stepper=0110
#               120c=0,r=0, e=1,l=011, servo=1, strt=1, stepper=0110
#               125c=1,r=0, e=1,l=100, servo=1, strt=1, stepper=0010
#               130c=0,r=0, e=1,l=100, servo=1, strt=1, stepper=0010
#               135c=1,r=0, e=1,l=101, servo=1, strt=1, stepper=0011
#               140c=0,r=0, e=1,l=101, servo=1, strt=1, stepper=0011
#               145c=1,r=0, e=1,l=110, servo=1, strt=1, stepper=0001
#               150c=0,r=0, e=1,l=110, servo=1, strt=1, stepper=0001
#               155c=1,r=0, e=1,l=111, servo=1, strt=1, stepper=1001
#               160c=0,r=0, e=1,l=111, servo=1, strt=1, stepper=1001
#               165c=1,r=0, e=1,l=000, servo=0, strt=1, stepper=1000
#               170c=0,r=0, e=1,l=000, servo=0, strt=1, stepper=1000
#               175c=1,r=0, e=1,l=001, servo=1, strt=1, stepper=1100
#               180c=0,r=0, e=1,l=001, servo=1, strt=1, stepper=1100
#               185c=1,r=0, e=1,l=010, servo=1, strt=1, stepper=0100
#               190c=0,r=0, e=1,l=010, servo=1, strt=1, stepper=0100
#               195c=1,r=0, e=1,l=011, servo=1, strt=1, stepper=0110
# ** Note: $finish : C:/altera/13.0sp1/modelsim_ase/win32aloem/RC_Servo & Stepper/SERVO_STEPPER_TB.v(53)
#    Time: 200 ps  Iteration: 0  Instance: /css_test
# 1
# Break in Module css_test at C:/altera/13.0sp1/modelsim_ase/win32aloem/RC_Servo & Stepper/SERVO_STEPPER_TB.v line 53[6]
```

C. RTL Code

```
module Counter_Servo_Stepper(clock, reset, enable, led, Pin_servo, start, Pins_stepper);

////////// 3 bit LED counter ///////////

input clock, reset, enable, start;
output [2:0] led;
reg [2:0] count;
always @ (posedge clock or posedge reset)
  if (reset) begin
   count <= 0;
    end
   else
   begin: COUNT
 while (enable) begin
  count <= count + 1;
  disable COUNT;

  end
 end
assign led = count;

 ////////// RC Servo //////////////

///Using a 25MHz clock (40ns period), the first step is to
divide the clock to generate a "tick" of period as close as
possible to 3.9µs.

parameter CLOCK_DIVIDER = 98;  //
25000000/1000/256 = 97.56

 output Pin_servo;
reg Pin_servo;

//Using the "ClkTick", we instantiate a 12-bits counter
that increments at every tick.
//Each tick lasts 3.9µs, so 256 ticks lasts 1ms, and the 12
bits counter "ClkTick" rolls-over every 16ms.
//Just what we need to generate a new pulse regularly.
reg [31:0] clockCount;
reg [11:0] ClkTick = 0;
always @ (posedge clock)
begin
   if(clockCount == CLOCK_DIVIDER-2) // a "tick" has
happened
      begin
          ClkTick <= ClkTick + 1;
          clockCount <= 0;
      end
   else
      clockCount <= clockCount + 1;
end

// Finally, we compare 12 bits to "ClkTick" to generate
the pulse
always @ (posedge clock) Pin_servo <= (ClkTick <
{~count, 5'b00000});

////////// Stepper Motor - Half step method (using 28BYJ-
48) //////////////


output [3:0] Pins_stepper ;
reg [3:0] Pins_stepper ;

reg [2:0] step ;

initial
step = 0; //8 positions for half step//


always @ (posedge clock)
begin
 if (start)
  step <= step + 1;
end

always @ (step) begin
 case (step)
        0: Pins_stepper <= 4'b1000;
        1: Pins_stepper <= 4'b1100;
        2: Pins_stepper <= 4'b0100;
        3: Pins_stepper <= 4'b0110;
        4: Pins_stepper <= 4'b0010;
        5: Pins_stepper <= 4'b0011;
        6: Pins_stepper <= 4'b0001;
        7: Pins_stepper <= 4'b1001;
 endcase
end
endmodule
```

D. Test Bench

```
module css_test;
  reg clock, reset, enable, start;
  wire [2:0] led;
  wire Pin_servo;
  wire [3:0] Pins_stepper;

  Counter_Servo_Stepper css1(clock, reset, enable, led,
Pin_servo, start, Pins_stepper);
    initial
   begin
     $monitor($time,"c=%b,r=%b, e=%b,l=%b, servo=%b,
strt=%b, stepper=%b",clock, reset, enable, led, Pin_servo,
start, Pins_stepper);
   end
initial


begin

#0 clock = 1'b0;

#10 clock = 1'b1;

end

initial

begin

#0 reset = 1'b1;

#10 reset = 1'b0;

end

initial

begin

#0 start = 1'b0;
```

```
#10  start = 1'b1;

end

initial

begin
  #0 enable = 1'b1;


end

 always #5 clock=~clock;
   initial
   #200 $finish;

      ENDMODULE
```

VI. CONCLUSION

As the final output of the test bench shows, the stepper motor works just like the expected output. The use of FPGA implements this servo motor will allow less resources required to operate such devices and will ensure mobility in the module. A controllable system for the servo motor shall allow this to work on multiple platforms with applications of robotics, mechanical devices and many other options.

ACKNOWLEDGMENT

With the name of The Almighty, The Most Gracious and Merciful. Praise to Him for giving us the will and strength to go through the entire project and for giving us the opportunity to complete this project report successfully.

First and foremost, we would like to express our deepest gratitude and appreciation to our respected supervisor, Mr. Iqbalur Rahman Rokon for his guidance, advices, supervision and encouragement in making the project. The valuable and useful ideas that he had shared with us during the project period are very much appreciated.

Special thanks to our beloved parents, family members and friends for their help and supports in producing this report. Lastly, we would like to convey our gratitude to our team members and other people who directly or indirectly help us in the process of finishing this report.

REFERENCES

[1] [Kariyappa B. S; Dr. M. Uttara Kumari (2008). FPGA Based Speed Control of AC Servomotor Using Sinusoidal PWM. IJCSNS International Journal of Computer Science and Network Security. Available at: http:// paper.ijcsns.org/07_book/200810/20081053.pdf

[2] EP2C5T144C8 Cyclone II FPGA, https://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf accessed on 12th November, 2015

[3] 9 grams micro servo, https://solarbotics.com/product/25500/ accessed on 14th November, 2015

[4] 28-BYJ-48 stepper motor, https://robocraft.ru/files/datasheet/28BYJ-48.pdf accessed on 17th November, 2015

[5] ULN2003 Stepper Motor Driver, http://www.elecrow.com/wiki/index.php?title=ULN2003_Stepper_Motor_Driver accessed on 17th November, 2015

[6] http://forum.allaboutcircuits.com/threads/3-bit-up-counter-verilog-code.93600/ accessed on 22nd November, 2015

[7] http://verilogbynaresh.blogspot.com/2013/07/design-of-stepper-motor-driver-full.html accessed on 25th November, 2015

[8] http://www.fpga4fun.com/RCServos.html accessed on 29th November, 2015