Title:

HDL Modeling (System Verilog HDL or VHDL) and Sysnthesis of a Stepper Motor Controller

Introduction:

The main objective of this experiment is to understand the role of digital controllers in Stepper Motor control, how to control the rotation of the stepper motor by supplying appropriate signals to the different winding of the motor.

Theory and Methodology:

A controller for a uni-polar stepper motor has to be designed that allows a user to control the direction of rotation of the motor through two input signals CW and CW. The CW and CCW inputs have equal priority. When CW and CCW have the same value, the motor should hold its current position. If CW is high and CCW is low, the motor should rotate in clock-wise direction. If CW is low and CCW is high, the motor should rotate in the counter clock-wise direction. The clock period is 1 ms. The motor should receive each binary pattern in its windings (through the Motor_Drive output of the controller) for 1 ms. In other words, the controller should generate a particular pattern for 1 ms (otherwise it will be too fast to affect the motor) before deciding whether to hold the motor position, rotate the motor in the clock-wise direction or rotate it in the counter- clock-wise direction. Remember, a pattern means an individual pattern (say, 1001), not a set of patterns (say, 1010, 1001, 0101, 0110). Use Gray state encoding. The FSM should have an asynchronous reset signal that should take the machine to the initial state (state 0) when active.

Apparatus:

1. A Windows-based (XP or 7 or 10) PC with standard word processors (i.e. Microsoft Office) and PDF readers (i.e. Adobe Acrobat Reader/Writer, Foxit Reader/Phantom) installed.
2. Necessary EDA tools such Active-HDL, Precision RTL, ISE WebPack.

Precautions:

A PC with a standard Anti-Virus program installed was used.

Simulation and Measurement:

```
//Package Declaration

timeunit 1ns;
timeprecision 1ps;

package SMC_PKG;
parameter STATE_REG_WIDTH=3;
parameter MAX_OUTPUT_DURATON=8;
parameter OUTPUT_SIZE=4;
localparam COUNTER_WIDTH=$clog2(MAX_OUTPUT_DURATON);
endpackage:SMC_PKG

// Top Level Design

timeunit 1ns;
timeprecision 1ps;

import SMC_PKG::*;

extern module SMC_TOP ( input logic CW, input logic CCW, input logic SYS_CLK, input logic FSM_A_RESET, output logic [OUTPUT_SIZE-1:0] MOTOR_DRIVE);

module SMC_TOP(.*);

//internal signal

wire SCLR, INC, COUNT_LT_8;

FSM_MOD
F(.CW(CW),.CCW(CCW),.SYS_CLK(SYS_CLK),.FSM_A_RESET(FSM_A_RESET),.M
```

```
OTOR_DRIVE(MOTOR_DRIVE),.SCLR(SCLR),.COUNT_LT_8(COUNT_LT_8),.INC(I
NC));
DATAPATH_MOD
D(.SYS_CLK(SYS_CLK),.SCLR(SCLR),.INC(INC),.COUNT_LT_8(COUNT_LT_8));

endmodule: SMC_TOP
```

//FSM design

```
timeunit 1ns;
timeprecision 1ps;

import SMC_PKG::*;

extern module FSM_MOD ( input logic SYS_CLK, input logic CW, input logic
CCW, input logic FSM_A_RESET, inout logic COUNT_LT_8, output logic
[OUTPUT_SIZE-1:0] MOTOR_DRIVE, output logic SCLR, output logic INC);

module FSM_MOD(.*);

const logic[3:0] A=4'b1010;
const logic[3:0] B=4'b1001;
const logic[3:0] C=4'b0101;
const logic[3:0] D=4'b0110;
logic [STATE_REG_WIDTH-1:0] P_STATE, N_STATE;

always_comb
begin: NSOL

begin:NSL

N_STATE='bx;

case(P_STATE)

3'b000: N_STATE=3'b001;
3'b001: begin
```

```
if(COUNT_LT_8 || CW==CCW)
N_STATE=P_STATE;
else
begin
if(CW)
N_STATE=3'b011;
else
N_STATE=3'b110;
end
end

3'b011: begin
if(COUNT_LT_8 || CW==CCW)
N_STATE=P_STATE;
else
begin
if(CW)
N_STATE=3'b010;
else
N_STATE=3'b001;
end
end
3'b010: begin
if(COUNT_LT_8 || CW==CCW)
N_STATE=P_STATE;
else
begin
if(CW)
N_STATE=3'b110;
else
N_STATE=3'b011;
end
end

3'b110: begin
if(COUNT_LT_8 || CW==CCW)
N_STATE=P_STATE;
```

```
else
begin
if(CW)
N_STATE=3'b001;
else
N_STATE=3'b010;
End
end
default: N_STATE=3'b000; //assign FSM_A_RESET='b1;

endcase

end: NSL

begin: OL
{SCLR,INC}='b0;
case(P_STATE)

3'b000: SCLR=1'b1;
3'b001: begin
MOTOR_DRIVE=A;
INC=1'b1;
if((!COUNT_LT_8) && CW!=CCW)SCLR=1'b1;
end
3'b011: begin
MOTOR_DRIVE=B;
INC=1'b1;
if((!COUNT_LT_8) && CW!=CCW)SCLR=1'b1;
end
3'b010:begin
MOTOR_DRIVE=C;
INC=1'b1;
if((!COUNT_LT_8) && CW!=CCW)SCLR=1'b1;
end
3'b110:begin
MOTOR_DRIVE=D;
INC=1'b1;
```

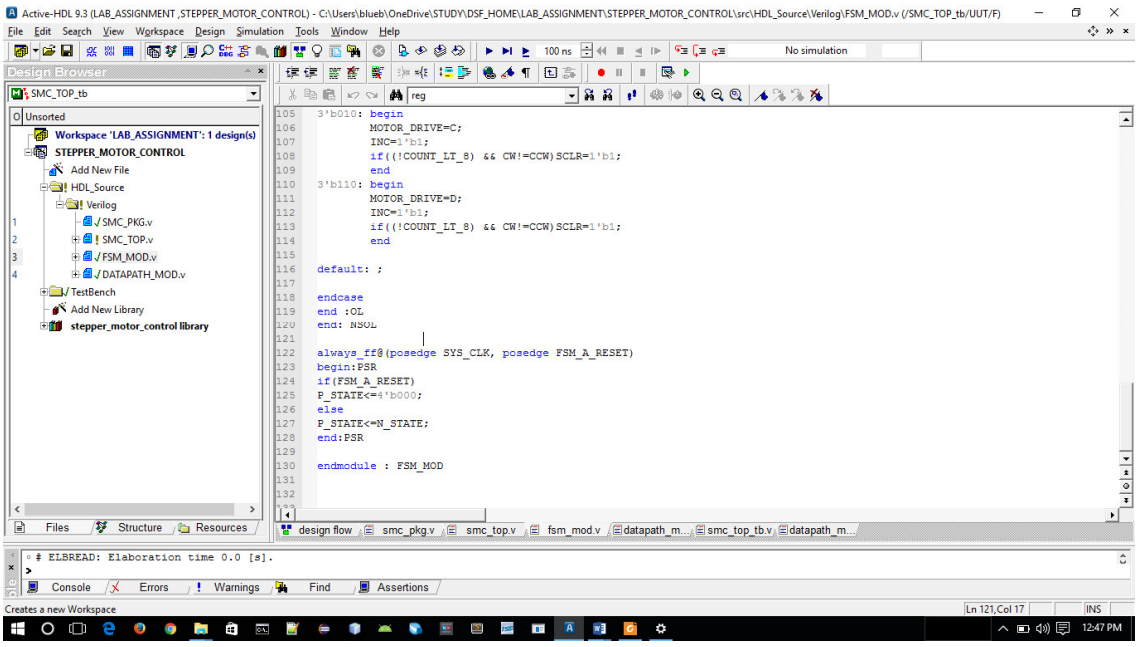
```
if((!COUNT_LT_8) && CW!=CCW)SCLR=1'b1;
end

default: ;

endcase
end :OL
end: NSOL

always_ff@(posedge SYS_CLK, posedge FSM_A_RESET)
begin:PSR
if(FSM_A_RESET)
P_STATE<=4'b000;
else
P_STATE<=N_STATE;
end:PSR

endmodule :FSM_MOD

//DATA_PATH

timeunit 1ns;
timeprecision 1ps;

import SMC_PKG::*;

extern module DATAPATH_MOD (
                        input logic SYS_CLK,
                        input logic SCLR,
                        input logic INC,
                        output logic [3:0] COUNT,
                        output logic COUNT_LT_8
);

module DATAPATH_MOD(.*);

//logic [COUNTER_WIDTH-1:0] COUNT;
```

always_ff @(posedge SYS_CLK)
begin: COUNTER
if(SCLR)
COUNT<=0;
else if(INC)
COUNT<=COUNT+1;
end: COUNTER

always_comb
begin: COMPARATOR
if(COUNT<'d8)COUNT_LT_8='b1;

else COUNT_LT_8='b0;
end :COMPARATOR

endmodule: DATAPATH_MOD



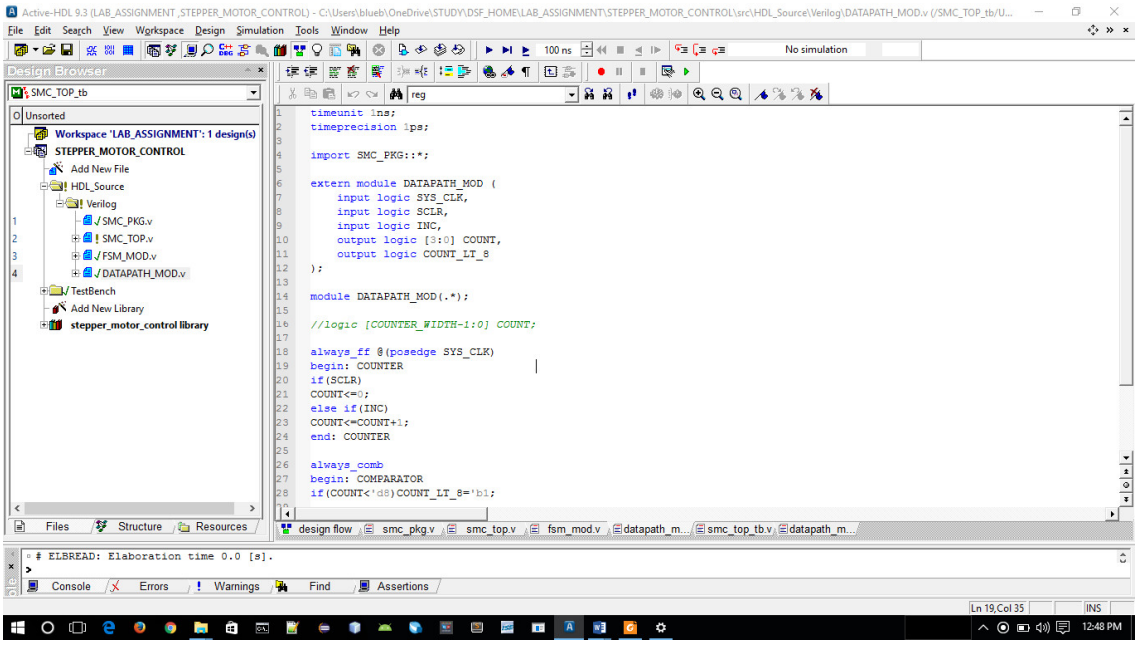


Figure 1: Screenshot of the design module

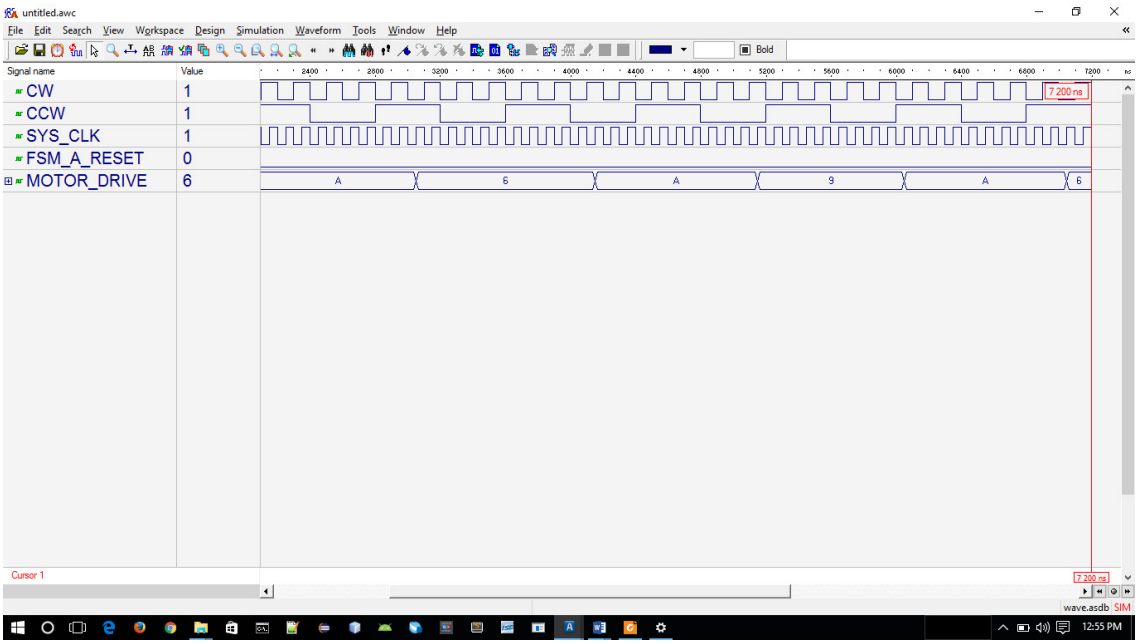


Figure 3: Functional Analysis

Reference:

1. *D.J. Smith, HDL Chip Design: A Practical Guide for Designing, Synthesizing & Simulating ASUC & FPGA using VHDL or Verilog , Madison, AL, USA, Doone Publications. 1996, 6th Printing-1999(minor revisions and code updates for FPGA synthesis)*