DEPARTMENT
OF
ELECTRONICS & COMMUNICATION ENGINEERING

HDL LABORATORY MANUAL
V Semester (17EC5DLHDL)
Autonomous Course2020

HOD : Dr. Manjunath T C
In -Charge: Madhura.R

Name of the Student	:	
Semester /Section	:	
USN	:	
Batch	:	

Dayananda Sagar College of Engineering
Shavige Malleshwara Hills, Kumaraswamy Layout,
Banashankari, Bangalore-560078, Karnataka
Tel : +91 80 26662226 26661104 Extn : 2731 Fax : +90 80 2666 0789
Web - http://www.dayanandasagar.edu Email : hod-ece@dayanandasagar.edu
(An Autonomous Institute Affiliated to VTU, Approved by AICTE & ISO 9001:2008 Certified)
(Accredited by NBA, National Assessment & Accreditation Council (NAAC) with 'A' grade)

Dayananda Sagar College of Engineering

Dept. of E & C Engg

Name of the Laboratory	**:**	HDL LAB
Semester/Year	**:**	V/ 2020 (Autonomous)
No. of Students/Batch	**:**	20
No. of Equipment's	**:**	20
Major Equipment's	**:**	SPARTAN-6 KIT, OHP XILINX –ISE 14.2 TOOL Digital Storage Oscilloscope Function generator, CROs Fixed & Variable Power Supply HDL kits, PCs with peripherals
Area in square meters	**:**	109 Sq Mts
Location	**:**	Level – 2
Total Cost of Lab	**:**	Rs. 37, 77,535/-

Lab In charge/s: Prof. Chaitra.A
Prof. Navya Holla.K
Dr. Rajagopal
Prof. Manasa.R

Instructor : Mr. Pradyumna & Mr. Naveen

HOD : Dr. T.C. Manjunath, Ph.D. (IIT Bombay)

About the College & the Department

The Dayananda Sagar College of Engineering was established in 1979, was founded by Sri R. Dayananda Sagar and is run by the Mahatma Gandhi Vidya Peetha Trust (MGVP). The college offers undergraduate, post-graduates and doctoral programmes under Visvesvaraya Technological University & is currently autonomous institution. MGVP Trust is an educational trust and was promoted by Late. Shri. R. Dayananda Sagar in 1960. The Trust manages 28 educational institutions in the name of "Dayananda Sagar Institutions" (DSI) and multi – Specialty hospitals in the name of Sagar Hospitals - Bangalore, India. Dayananda Sagar College of Engineering is approved by All India Council for Technical Education (AICTE), Govt. of India and affiliated to Visvesvaraya Technological University. It has widest choice of engineering branches having 16 Under Graduate courses & 17 Post Graduate courses. In addition, it has 21 Research Centres in different branches of Engineering catering to research scholars for obtaining Ph.D under VTU. Various courses are accredited by NBA & the college has a NAAC with ISO certification. One of the vibrant & oldest dept is the ECE dept. & is the biggest in the DSI group with 70 staffs & 1200+ students with 10 Ph.D.'s & 30^{+} staffs pursuing their research in various universities. At present, the department runs a UG course (BE) with an intake of 240 & 2 PG courses (M.Tech.), viz., VLSI Design Embedded Systems & Digital Electronics & Communications with an intake of 18 students each. The department has got an excellent infrastructure of 10 sophisticated labs & dozen class room, R & D centre, etc…

Vision & Mission of the Institute

Vision of the Institute

To impart quality technical education with a focus on Research and Innovation emphasizing on Development of Sustainable and Inclusive Technology for the benefit of society.

Mission of the Institute

- To provide an environment that enhances creativity and Innovation in pursuit of Excellence.
- To nurture teamwork in order to transform individuals as responsible leaders and entrepreneurs.
- To train the students to the changing technical scenario and make them to understand the importance of Sustainable and Inclusive technologies.

Vision & Mission of the Department

Vision:

To prepare the students for global competence, with core knowledge in Electronics and communication engineering having focus on research to meet the needs of industry and society.

Mission:

- To provide in-depth knowledge of Electronics and Communication Engineering, ensuring the effective teaching learning process.
- To train students to take-up innovative projects in group with emerging technology relevant to the industry and social needs.
- To imbibe professional ethics, research culture and development of skills.

Program Educational Objectives

Graduate students must be able to:

PEO1: Have Core knowledge of ECE with strong Teaching learning Process who are ready to apply the state-of-art technology to solve Industry and socially relevant problems.

PEO2: Engage in team with work technical knowledge and effective communication skills to address the practical issues in industry and society.

PEO3: Be technologists who can analyze and design innovative projects through research and emerging technology.

PEO4: Professionals, with capabilities for pursuing higher studies in technical/managerial courses.

Program Specific Outcomes

PSO1: Promote Electronics and Communication engineering in meeting technical, environmental and societal challenges.

PSO2: Integrate design and development of electronic systems and circuits using current practices and standards.

PSO3: Apply knowledge of Electronics and Communication Engineering in building projects, developing good communication skills, incorporating ethical behavior for project management, with financial prudence.

HDL LABORATORY (SYLLABUS)

AUTONOMOUS COURSE

V SEMESTER B. E (E & C)

Course code : 17EC5DLHDL	Credits : 2
L : P : T : S : 1 : 2 : 0 : 0	IA CIE Marks : 50
Exam Hours : 3	SEE Marks : 50

Course Objectives:

1. To develop a skill to write the HDL program for combinational circuits and sequential circuits.
2. Learn how to simulate, analyze the developed program.
3. To know how to debug the written code.
4. Familiarize the implementation of digital functions using FPGA kit.
5. To give exposure on interfacing concepts using FPGA kit.
6. To give an in-depth exposure to the various tools in Xilinx.

Syllabus:

Module	Expt No.	Content of the Lab Module with Expt. Nos.	Hours	COs
Software : Programming Using Verilog				
Part A		NOTE: 1.Design using Verilog ♦ RTL Development 2.Verification using Verilog ♦ Test bench and Test case Development 3.Simulation and Debug Simulation Tools: Xilinx/Modelsim/I Verilog		
	1	Write a HDL code to realize all the logic gates	03	CO 1-4
	2	Write a VHDL code to describe the functions of a Full Adder using three modelling styles.	03	CO 1-4
	3	Write a HDL program for the following combinational designs a. 2 to 4 decoder b. 8 to 3 (encoder without priority & with priority) c. 8 to 1 multiplexer d. 4 bit binary to gray converter Multiplexer, de-multiplexer, comparator	03	CO 1-4
	4	Develop the HDL code for the following flip-flops, a. SR Flip-Flop b. D Flip-Flop c. JK Flip-Flop d. T Flip-Flop	03	CO 1-4
	5	Write a model for 8bit ALU. ALU should use combinational logic to calculate an output based on the 3bit op-code input. ALU should decode the 3 bit op-code according to the given in example below. Opcode (2:0) OPCODE ALU OPERATION i). A + B ii). A – B iii). A Complement iv). A * B	03	CO 1-4

		v). A AND B vi). A OR B vii). A NAND B viii). A XOR B		
	6	Design 4 bit Binary and BCD counters (Synchronous reset and Asynchronous reset) and “any sequence” counters	03	CO 1-4
	7.	Design Mealy and Moore state machine for a given sequence.	03	CO 1-4
	8	Write a HDL code for Serial adder.	03	CO 1-4
Hardware : INTERFACING (using VHDL)				
Part B	7	Write HDL code to control speed, direction of DC and Stepper motor.	03	CO 1-6
	8	Write HDL code to generate different waveforms (Square, Triangle, Ramp etc.,) using DAC.	03	CO 1-6
	9	Write HDL code to control external lights using relays.	03	CO 1-6

Course Outcomes:

At the end of the course, student will be able to

CO1	Gain the knowledge of using XILINX ISE tool.
CO2	Develop the HDL code for combinational circuits and simulate using XILINX ISE tool
CO3	Analyze and debug VHDL and Verilog code for combinational and sequential digital circuits using XILINX tool and ModelSim simulator
CO4	Synthesis of digital circuit with FPGA kit.
CO5	Develop a VHDL program and synthesis of a different hardware interfacing experiments
CO6	Design of Digital system for small real time problems and Implement on FPGA kit.

CO-PO mapping:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	2	2	2	3	-	-	-	-	-	-	2
CO2	3	3	3	3	3	-	-	-	-	-	-	2
CO3	3	3	2	2	3	-	-	-	-	-	-	2
CO4	3	3	3	3	3	-	-	-	-	-	-	2
CO5	3	3	3	3	3	-	-	-	-	-	-	2
CO6	3	3	3	3	3	-	-	-	-	-	-	2

HDL LABORATORY (17EC5DLHDL)

I - CYCLE

1. Write HDL code to realize all the logic gates.

2. Write a HDL code to describe the functions of a Full Adder using three modelling styles.

3. Write a HDL program for the following combinational designs
   a. 2 to 4 Decoder.
   b. 8 to 3 Encoder (without priority & with priority).
   c. 8 to 1 Multiplexer.
   d. De-multiplexer
   e. 4 bit Binary to Gray converter.
   f. Comparator

4. Develop the HDL code for the following flip-flops
   a. SR Flip-Flop
   b. D Flip-Flop
   c. JK Flip-Flop
   d. T Flip-Flop

5. Write a model for 8bit ALU. ALU should use combinational logic to calculate an output based on the 3 bit op-code input. ALU should decode the 3 bit op-code according to the given in example below.
   Opcode (2:0)
   OPCODE ALU OPERATION
   1. A + B
   2. A – B
   3. A Complement
   4. A * B
   5. A AND B
   6. A OR B
   7. A NAND B
   8. A XOR B

6. Design 4 bit Binary and BCD counters(Synchronous reset and Asynchronous reset) and any sequence Counters.

7. Design Mealy and Moore state machine for a given sequence.

8. Write a HDL code for Serial adder.

II – CYCLE

9. Write HDL code to control speed, direction of DC and Stepper motor.

10. Write HDL code to generate different waveforms (Square, Triangle, Ramp etc.,) using DAC.

11. Write HDL code to control external lights using relays.

DO's

- Students should follow the dress code of the laboratory compulsorily.
- Keep your belongings in the corner of the laboratory.
- Students have to enter their name, USN, time in/out and signature in the log register maintained in the laboratory.
- Students are required to enter components in the components register related to the experiment and handle the equipment's smoothly.
- Check the components, range and polarities of the meters before connecting to the circuit.
- Come prepare for the experiment and background theory.
- Before connecting to the circuit refer the designed circuit diagram properly. Debug the circuit for proper output.
- Students should maintain discipline in the laboratory and keep the laboratory clean and tidy.
- Observation book and Record book should be complete in all respects and get it corrected by the staff members.
- Clarify the doubts with staff members and instructors.
- Experiment once conducted, in the next lab, the entire record should be complete in all respects, else the student will lose the marks.
- For programming lab, show the output to the concerned faculty.
- All the students should come to LAB on time with proper dress code and identity cards
- Keep your belongings in the corner of laboratory.
- Students have to enter their name, USN, time-in/out and signature in the log register maintained in the laboratory.
- All the students should submit their records before the commencement of Laboratory experiments.
- Students should come to the lab well prepared for the experiments which are to be performed in that particular session.

- Students are asked to do the experiments on their own and should not waste their precious time by talking, roaming and sitting idle in the labs.
- Observation book and record book should be complete in all respects and it should be corrected by the staff member.
- Before leaving the laboratory students should arrange their chairs and leave in orderly manner after completion of their scheduled time.
- Prior permission to be taken, if for some reasons, they cannot attend lab.
- Immediately report any sparks/ accidents/ injuries/ any other untoward incident to the faculty /instructor.
- In case of an emergency or accident, follow the safety procedure.
- Switch OFF the power supply after completion of experiment.

DONT's

- Do not switch on the power supply before verification of the connected circuits by concerned staff.
- Do not feed higher voltages than rated to the device.
- Do not upload, delete or alter any software on the laboratory PC's.
- Do not write or mark on the equipment's.
- Usage of mobile phone is strictly prohibited.
- Ragging is punishable.
- If student damages the equipment or any of the component in the lab, then he / she is solely responsible for replacing that entire amount of the equipment or else, replace the equipment.
- The use of mobile/ any other personal electronic gadgets is prohibited in the laboratory.
- Do not make noise in the Laboratory & do not sit on experiment table.
- Do not make loose connections and avoid overlapping of wires.
- Don't switch on power supply without prior permission from the concerned staff.
- Never point/touch the CRO/Monitor screen with the tip of the open.

Experiment No: 1 Date: ____________

LOGIC GATES

Aim: Write a HDL code to realize all LOGIC GATES.

Verilog

```
module logicgates(a,b,y);
input a,b;
output[6:0] y;
assign y[0] = ~a;
assign y[1] = a & b;
assign y[2] = a|b;
assign y[3] = ~(a&b);
assign y[4] = ~(a|b);
assign y[5] = a^b;
assign y[6] = ~(a^b);
endmodule
```

Testbench

Results:

Applications:

Remarks :

Signature of Staff Incharge with date:

Experiment No: 2 Date: ____________

FULL ADDER

Aim: Write a HDL code FULL ADDER in all styles.

1. DATA FLOW
2. BEHAVIORAL
3. STRUCTURAL

1. DATA FLOW

VHDL

```
ENTITY fulladder IS
        PORT(a,b,cin:IN std_logic;
      sum,cout:OUT std_logic);
END fulladdder;
ARCHITECTURE fulladder_DF OF fulladder IS
BEGIN
        sum<=a XOR b XOR cin;
        cout<= (a AND b) OR (b AND cin) OR (a AND cin);
END fulladder_DF;
```

Verilog

```
module fulladder (a, b, cin, sum, cout);
input a, b, cin;
output sum, cout;
assign sum = a^b^cin;
assign cout = (a&b)|(b&cin)|(a&cin);
endmodule
```

2. <u>BEHAVIORAL</u>

<u>VHDL</u>

```
ENTITY fulladder IS
        PORT(a, b, cin:IN std_logic;
                sum, cout:OUT std_logic);
END fulladdder;
ARCHITECTURE fulladder_BH OF fulladder IS
SIGNAL x:std_logic_vector(2 downto 0);
BEGIN
        x<= a & b & cin;
        PROCESS(x)
        BEGIN
        IF(x="001"OR x="010" OR x="100" OR x="111") THEN
           sum<='1';ELSE sum<='0';END IF;
        IF(x="011"OR x="101" OR x="110" OR x="111") THEN
           cout<='1';ELSE cout<='0';END IF;
        END PROCESS;
END fulladder_BH;
```

<u>Verilog</u>

```
module fulladder_bh (a, b, cin, sum, cout)
input a, b, cin;
output sum, cout;
reg sum, cout;
wire [2:0] x;
assign x = {a, b, cin};
always @(x)
begin
        if(x==3'b001|x==3'b010|x==3'b100|x==3'b111)
        sum = 1'b1; else sum = 1'b0;
        if(x==3'b011|x==3'b101|x==3'b110|x==3'b111)
        cout = 1'b1; else cout = 1'b0;
end
endmodule
```

3. STRUCTURAL

VHDL

```
ENTITY fulladder IS
        PORT(a, b, cin:IN std_logic;
                sum, cout:OUT std_logic);
END fulladdder;
ARCHITECTURE fulladder_ST OF fulladder IS
COMPONENT ha is
Port(x,y:in std_logic;
        S,c: out std_logic);
End component;
COMPONENT or2 is
Port(x,y:in std_logic;
        z: out std_logic);
End component;
Signal s1,s2,s3: std_logic;
BEGIN
U1: ha port map(a,b,s1,s2);
U2: ha port map(s1,cin,sum,s3);
U3: or2 port map(s2,s3,cout);
End fulladder_ST;
```

Note: Components to be declared

Verilog

```
module fulladder_bh (a, b, cin, sum, cout);
input a, b, cin;
```

```
output sum, cout;
wire s1,s2,s3;
xor x1(sum,a,b,cin);
and a1(s1,a,b);
and a2(s2,a,cin);
and a3(s3,b,cin);
or o1(cout,s1,s2,s3);
endmodule
```

Test Bench

Result:

Applications:

Remarks:

Signature of Staff Incharge with date

Experiment No:3a Date: ____________

DECODER (2:4)

Aim: Write a HDL code to design 2:4 DECODER

Verilog

```
module decoder(I,D);
input [1:0] I;
output [3:0] D;
reg [3:0] D;
always @(I)
begin
        if (I==2'B00) D=4'B0001;
        else if (I==2'B01) D=4'B0010;
        else if (I==2'B10) D=4'B0100;
        else if (I==2'B11) D=4'B1000;
        else D=4'BZZZZ;
end
endmodule
```

Test Bench

Results:

Applications:

Remarks:

Signature of Staff Incharge with date:

Experiment No:3b Date: ____________

ENCODER (8:3)

Aim: Write a HDL code to design 8:3 ENCODER with and without priority.

Verilog

```
module encode (I,D);
input [7:0] I;
output [2:0] D;
reg [2:0] D;
always @ (I)
begin
        if (I[7]==1'B1) D=3'B111;
        else if (I[6]==1'B1) D=3'B110;
        else if (I[5]==1'B1) D=3'B101;
        else if (I[4]==1'B1) D=3'B100;
        else if (I[3]==1'B1) D=3'B011;
        else if (I[2]==1'B1) D=3'B010;
        else if (I[1]==1'B1) D=3'B001;
        else if (I[0]==1'B1) D=3'B000;
        else D=3'BZZZ;
end
endmodule
```

TestBench

Result:

1. **Without priority**

Verilog

```
module encoder_wop (I,D);
input [7:0] I;
output [2:0] D;
reg [2:0] D;
always @(I)
begin
  case (I)
  8'd1 : D=3'd0;
  8'd2 : D=3'd1;
  8'd4 : D=3'd2;
```

```
   8'd8 : D=3'd3;
   8'd16 : D=3'd4;
   8'd32 : D=3'd5;
   8'd64 : D=3'd6;
   8'd128 : D=3'd7;
default: D=3'BZZZ;
endcase
end
endmodule
```

TestBench

Result:

Inputs								Outputs		
I(7)	I(6)	I(5)	I(4)	I(3)	I(2)	I(1)	I(0)	D(2)	D(1)	D(0)
1	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0

Applications:

Remarks :

Signature of Staff Incharge with date:

Experiment No: 3c Date: ____________

MULTIPLEXER (8:1)

Aim: Write a HDL code to design 8:1 MULTEPLEXER

Verilog

```
module mux8_1 (I, S, D);
input [7:0] I;
input [2:0] S;
output D;
reg D;
always @ (I, S)
begin
   case(S)
   3'd0: D=I[0];
   3'd1: D=I[1];
   3'd2: D=I[2];
   3'd3: D=I[3];
   3'd4: D=I[4];
   3'd5: D=I[5];
   3'd6: D=I[6];
   3'd7: D=I[7];
default: D=1'BZ;
endcase
end
endmodule
```

TestBench

Result:

INPUTS			OUTPUTS
S(2)	S(1)	S(0)	D
0	0	0	I(0)
0	0	1	I(1)
0	1	0	I(2)
0	1	1	I(3)
1	0	0	I(4)
1	0	1	I(5)
1	1	0	I(6)
1	1	1	I(7)

Applications:

Remarks :

Signature of Staff Incharge with date:

Experiment No: 3d Date: ____________

DEMULTIPLEXER

Aim: Write a HDL code to design 1:8 DEMULTEPLEXER

Verilog

```
module demux1_8(I,S,D);
input I;
input [2:0] S;
output  [7:0]D;
reg [7:0]D;
always @(I,S)
begin
D=8’d0;
   CASE (S)
   3'd0 : D[0]=I;
   3'd1 : D[1]=I;
   3'd2 : D[2]=I;
   3'd3 : D[3]=I;
   3'd4 : D[4]=I;
   3'd5 : D[5]=I;
   3'd6 : D[6]=I;
   3'd7 : D[7]=I;
default: D=8'B00000000;
endcase
end
endmodule
```

TestBench

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 2e Date: ____________

BINARY TO GRAY

Aim: Write a HDL code to convert 4-bit BINARY TO GRAY

Verilog

```
module bin_to_gray(B,G);
input [3:0] B;
output [3:0] G;
assign G[0] = B[1]^B[0];
assign G[1] = B[2]^B[1];
assign G[2] = B[3]^B[2];
assign G[3] = B[3];
endmodule
```

Test Bench

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 2f Date: ____________

COMPARATOR

Aim: Write a HDL code to design 4-bit comparator.

Verilog

```
module comparator (a, b, alb, aeb, agb)
input[3:0] a, b;
output alb, aeb, agb;
reg alb, aeb, agb;
always@(a,b)
begin
if(a<b)
        begin
        alb=1’b1; aeb=1’b0; agb=1’b0;
        end
else if (a>b)
        begin
        alb=1’b0; aeb=1’b0; agb=1’b1;
        end
else
        begin
        alb=1’b0; aeb=1’b1; agb=1’b0;
        end
end
endmodule
```

TestBench

Result :

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 4a Date: ____________

SR FLIP-FLOP

Aim: Write a HDL code for SR-FF.

Verilog

```
module dff(sr, clk, q);
input clk;
input [1:0]sr;
output q;
reg q;
always @(posedge clk)
begin
        case (sr)
        2'B00:q=q;
        2'B01:q=1'B0;
        2'B10:q=1'B1;
        default:q=1'BZ;
        endcase
end
endmodule
```

TestBench

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 4b Date: ____________

D-FLIP FLOP

Aim: Write a HDL code for D-FF.

Verilog

```
module dff(d, rst, clk, q, qb);
input d, rst, clk;
output q, qb;
reg q, qb;
always @(posedge clk)
begin
        if (rst == 1'B1)
        begin
                q = 1'B0; qb = 1'B1;
        end
        else
        begin
                q = d; qb = ~q;
        end
end
endmodule
```

Test Bench

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 4c Date: ____________

JK-FLIP FLOP

Aim: Write a HDL code for JK-FF.

Verilog

```
module jkff(jk, clk, q, qb);
input clk;
input [1:0]jk;
output q,qb;
reg q,qb;
always @(POSEDGE clk)
begin
        case(jk)
        2'B00:q=q;
        2'B01:q=1'B0;
        2'B10:q=1'B1;
        2’B11:q= ~q;
        default:q=1'BZ;
        endcase
        qb=~q;
end
endmodule
```

Verilog

JK FLIPFLOP(For Hardware Implementation)

```
module jkff(jk, clk, q, qb);
input clk;
input [1:0]jk;
```

```
output q,qb;
reg q,qb;
reg clk1;
reg[22:0] div
always @(posedge clk)
begin
div<=div+1’b1;
clk1<=div[22];
end
always @(posedge clk1)
begin
        case(jk)
        2'b00:q=q;
        2'b01:q=1'b0;
        2'b10:q=1'b1;
        2’b11:q= ~q;
        default:q=1'bz;
        endcase
        qb=~q;
end
endmodule
```

Test Bench

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 4d   Date: ____________

T-FLIP FLOP

Aim: Write a HDL code for T-FF.

Verilog

```
module tff(t, clk, q, qb)
input t, clk;
output q, qb;
reg q, qb;
always @(posedge clk)
begin
	if (t==1'B0)
	begin
		q = q; qb = qb;
	end
	else
	begin
		q = ~q; qb = ~qb;
	end
end
endmodule
```

TestBench

Result:

NOTE: Include clock divider to the above program to implement T flip-flop in hardware(both VHDL and Verilog).

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 5 Date: ____________

4-bit ALU

Aim: Write a HDL code to design 4-bit ALU.

Verilog

```
module alu (a, b, code, en, aluout)
input [3:0] a, b;
input [2:0] code;
input en;
output [7:0] aluout;
reg [7:0] aluout ;
wire[7:0] x, y;

assign x = {4'B0000, a};
assign y = {4'B0000, b};
always @ (x, y, code, en, a,b)
begin
        If (en == 1'B1)
                case (code)
                3'd0: aluout = x + y;
                3'd1: aluout = x - y;
                3'd2: aluout = ~x;
                3'd3: aluout = a * b;
                3'd4: aluout = x & y;
                3'd5: aluout = x | y;
                3'd6: aluout = ~(x & y);
                3'd7: aluout = ~(x | y);
                default: aluout = 8'B00000000;
                endcase
                else
                aluout = 8'BZZZZZZZZ;
                end
                endmodule
```

TestBench

Result

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 6a Date: ____________

SYNCHRONOUS BINARY COUNTER

Aim: Write a HDL code to design 4-bit synchronous binary counter.

Verilog

```
module syncbinary (clk, rst, q);
input clk, rst;
output [3:0]q;
reg [3:0]q;
initial q = 4'B0000;
always @ (posedge clk)
begin
if (rst == 1'B1)
                q = 4'B0000;
else
                q = q + 1;
end
endmodule
```

Test Bench

4-bit synchronous Binary counter (For Hardware Implementation)

Verilog

```
module syncbinary (clk, rst, q);
input clk, rst;
output [3:0]q;
reg [3:0]q;
reg clkdiv;
reg[22:0] div;
initial q = 4'B0000;
always @ (posedge clk)
begin
div=div+1'B1;
clkdiv=div[22];
end
always@(posedge clkdiv)
begin
if (rst == 1'B1)
                q = 4'B0000;
else
                q = q + 1;
end
endmodule
```

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 6b Date: ____________

SYNCHRONOUS BCD COUNTER

Aim: Write a HDL code to design 4-bit synchronous BCD counter.

Verilog

```
module syncbcd (clk, rst, q);
input clk, rst;
output [3:0]q;
reg [3:0]q;
initial q = 4’b0000;
always @ (posedge clk)
begin
        if (rst == 1’b1|q==4’b1001)
                q = 4’b0000;
        else
                q=q+1;
end
endmodule
```

TestBench

Result:

Note: Write a program to implement Synchronous BCD counter using both VHDL and Verilog in hardware.

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 6c Date: ____________

ASYNCHRONOUS BINARY COUNTER

Aim: write a HDL code to design 4-bit Asynchronous binary counter.

Verilog

```
module asyncbinary (clk, rst, q);
input clk, rst;
output [3:0]q;
reg [3:0]q;
initial q = 4'b0000;
always @ (posedge clk or posedge rst)
begin
if (rst == 1'b1)
                q = 4'b0000;
else
q=q+1;
end
endmodule
```

4-bit Asynchronous Binary counter (For Hardware Implementation)

Verilog

```
    module bsync (clk, reset, count);
    input clk;
    input reset;
    output [3:0] count;
    reg [3:0] count;
    reg[22:0] div;
    reg clkdiv;
    always @ (posedge clk)
    begin
    div=div+1’b1;
```

```
clkdiv=div[22];
end
always @ (posedge clkdiv or posedge RESET)
begin
if (reset==1)
count=4'b0000;
else
count=count+1'b1;
end
endmodule
```

TestBench

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 6d Date: ____________

ASYNCHRONOUS BCD COUNTER

Aim: write a HDL code to design 4-bit Asynchronous BCD counter.

Verilog

```
Module asyncbcd (clk, rst, q);
Input clk, rst;
Output [3:0]q;
Reg [3:0]q;
Initial q = 4'b0000;
Always @ (posedge clk or posedge rst)
Begin
        If (rst == 1'b1)q = 4'b0000;
        Elsif (q== 4'b1001) q = 4'b0000;
        Else
                q=q+1;
End
Endmodule
```

TestBench

Result:

<u>Note:</u> Write a program to implement Synchronous BCD counter using both VHDL and Verilog in hardware.

Applications:

Signature of Staff Incharge with date

Experiment No: 7a Date: ____________

Mealy state machine

Aim: Design Mealy and Moore state machine for a given sequence.

```
Module  state_machine_moore(clk,  reset,  in, out);
parameter  zero=0,  one1=1,  two1s=2;
output  out; input  clk,  reset,  in;
reg  out; reg  [1:0]  state, next_state;
//  Implement  the  state  register
always  @(posedge clk or posedge reset) begin
  if (reset)
    state <= zero;
  else
    state <= next_state;
  end
always  @(state or in) begin
  case  (state)
   zero:  begin  //last  input  was  a  zero out  =  0;
   if  (in)
      next_state=one1;
   else
      next_state=zero;
   end
   one1:  begin  //we've  seen  one  1 out  =  0;
   if  (in)
     next_state=two1s;
   else
     next_state=zero;
   end
   two1s:  begin  //we've  seen  at  least  2  ones out  =  1;
   if  (in)
```

```
     next_state=two1s;
   else
     next_state=zero;
   end
   default:  //in  case  we  reach  a  bad  state out  =  0;
    next_state=zero;
  endcase
 end
 // output logic
 always  @(state) begin
  case (state)
    zero: out <= 0;
     one1: out <= 0;
     two1s: out <= 1;
   default : out <= 0;
   endcase
 end
endmodule
```

TestBench:

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 7b   Date: ____________

Moore state machine

Aim:Design Moore state machine for a given sequence.

```
module  state_machine_moore(clk,  reset,  in, out);
parameter  zero=0,  one1=1,  two1s=2;
output  out; input  clk,  reset,  in;
reg  out; reg  [1:0]  state, next_state;
//  Implement  the  state  register
always  @(posedge clk or posedge reset) begin
  if (reset)
    state <= zero;
  else
    state <= next_state;
  end

always  @(state or in) begin
  case  (state)
   zero:  begin  //last  input  was  a  zero out  =  0;
   if  (in)
      next_state=one1;
   else
      next_state=zero;
   end
   one1:  begin  //we've  seen  one  1 out  =  0;
   if  (in)
     next_state=two1s;
   else
     next_state=zero;
   end
   two1s:  begin  //we've  seen  at  least  2  ones out  =  1;
   if  (in)
     next_state=two1s;
   else
     next_state=zero;
   end
   default:  //in  case  we  reach  a  bad  state out  =  0;
    next_state=zero;
  endcase
 end
 // output logic
 always  @(state) begin
  case (state)
    zero: out <= 0;
     one1: out <= 0;
     two1s: out <= 1;
   default : out <= 0;
   endcase
 end
endmodule
```

Test Bench:

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 8 Date: ____________

Serial adder

Aim: Write a HDL code to design serial adder

```
module serialadder(clk,rst,pload,adata,bdata,enable,pout);
  input clk,rst,pload,enable;
  input [7:0] adata, bdata;
  output [7:0] pout;

  reg shiftrega_lsb,  shiftregb_lsb;
  reg [7:0] shiftrega, shiftregb, shiftregc;

  wire sum,cout;
  reg holdc, holda;

// instantiated the full adder
full_adder_1bit bit_adder_inst(shiftrega[0],shiftregb[0],holdc,sum,cout);



assign pout=shiftregc;

always@(posedge clk or rst)
 begin
   if (rst) begin
     shiftrega=8'd0;
     shiftregb=8'd0;
     shiftregc=8'd0;
   end else
   if(pload)   begin
     shiftrega=adata;
     shiftregb=bdata;
```

```
    shiftregc=8'b0;
   end else if(enable) begin
    shiftrega_lsb=shiftrega[0];
    shiftrega=shiftrega>>1;
    shiftrega[7]=shiftrega_lsb;

    shiftregb_lsb=shiftregb[0];
    shiftregb=shiftregb>>1;
    shiftregb[7]=shiftregb_lsb;

    shiftregc =shiftregc>>1;
    shiftregc[7]=sum;
  end
end


always@(posedge clk,rst)
 begin
   if(rst) begin
    holdc=1'b0;
    holda =1'b0;
   end else if(enable)
    holdc=cout;
   else
    holdc=holda;
end

endmodule
```

Test Bench:

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

INTERFACING PROGRAMS

Experiment No: 9a Date: ____________

DC MOTOR

Aim: Write HDL code to control speed and direction of DC Motor.

Theory:

The DC motor interface is implemented by L293D IC. The feedback is made up of opto-interrupter MOC70P2. When a fan blade of DC motor obstructs the IR rays of the MOC70P2, you get logic signal 1 at OPA or OPB. The sequence of outputs of two opto-interrupters OPA and OPB determines the direction and speed. The controls and feedback signals are available of 10 pin box header.

BH1 – BH2

E	OPA	NC	NC	GND
IN1	IN2	OPB	NC	NC

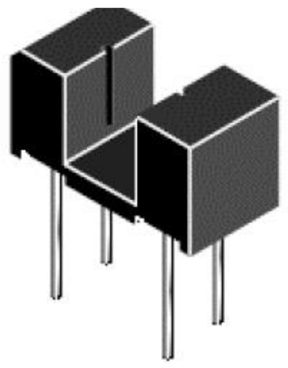
Signals E, IN1 and IN2 are inputs to control DC motor. Signals OPA and OPB are outputs of opto-interrupters.

DC MOTOR control

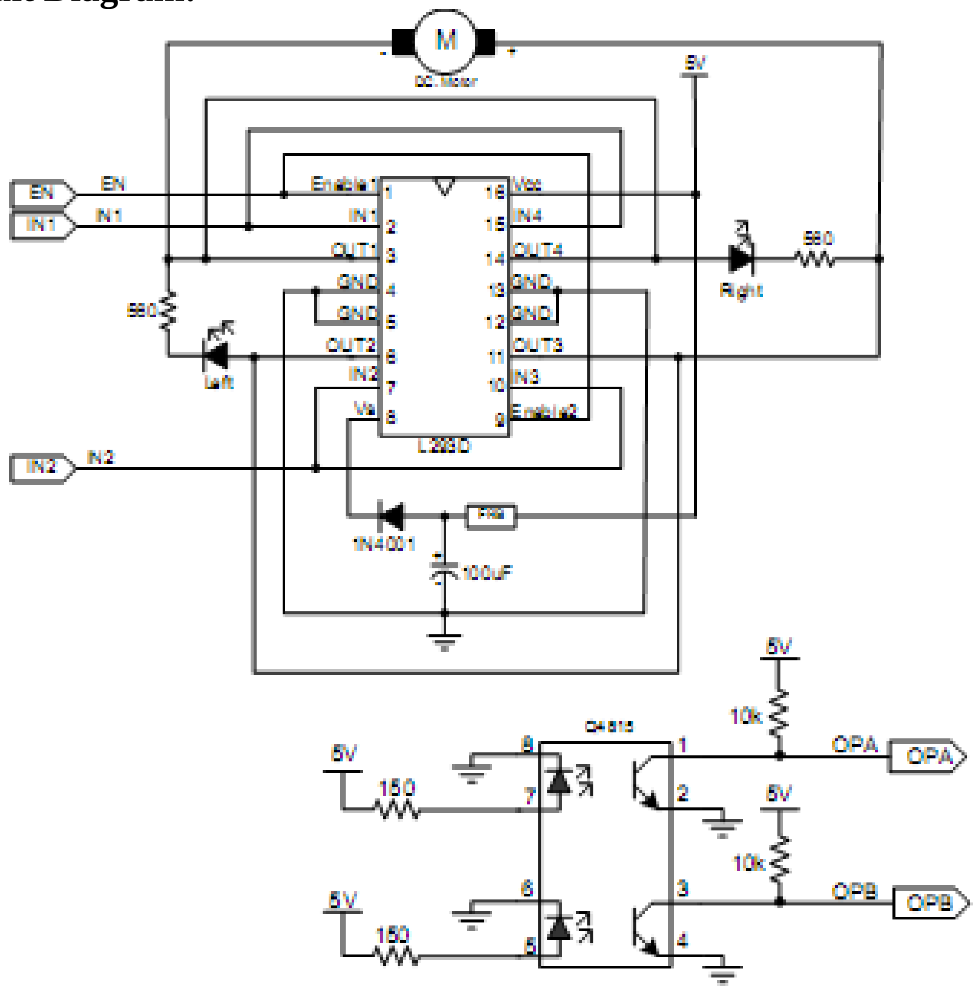
Control	E	IN1	IN2
STOP	0	X	X
RIGHT	1	1	0
LEFT	1	0	1

DESCRIPTION

The MOC70PX consists of an infrared light emitting diode coupled to an NPN silicon phototransistor packaged into an injection molded housing. The housing is designed for wide gap, non contact sensing.



Circuit Diagram:



VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity dcm is
port  ( pclk100k:in std_logic;
        counter:out std_logic_vector(17 downto 0);
        dcm_out:out std_logic_vector(2 downto 0));
end dcm;
```

```
architecture behavioral of dcm is
signal count18:std_logic_vector(17 downto 0);
signal dirsel:std_logic_vector(1 downto 0);
signal dcm_right,dcm_left,dcm_en:std_logic;
begin
process(pclk100k)                  --clock divider
begin
        if(rising_edge(pclk100k)) then
                dirsel<=count18(17 downto 16);
                count18<=count18+1;
        end if;
end process;
process(dirsel)
begin
case dirsel is
when "00"=> dcm_left<='0';
           dcm_right<='1';
           dcm_en<='1';
when "10"=> dcm_left<='1';
           dcm_right<='0';
           dcm_en<='1';
when others => dcm_left<='0';
           dcm_right<='0';
           dcm_en<='0';
end case;
dcm_out<=dcm_left & dcm_en & dcm_right;
end process;
counter<=count18(17 downto 0);
end behavioral;
```

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 9b   Date: ____________

STEPPER MOTOR

Aim: Write HDL code to control speed and direction of Stepper Motor.

Theory:

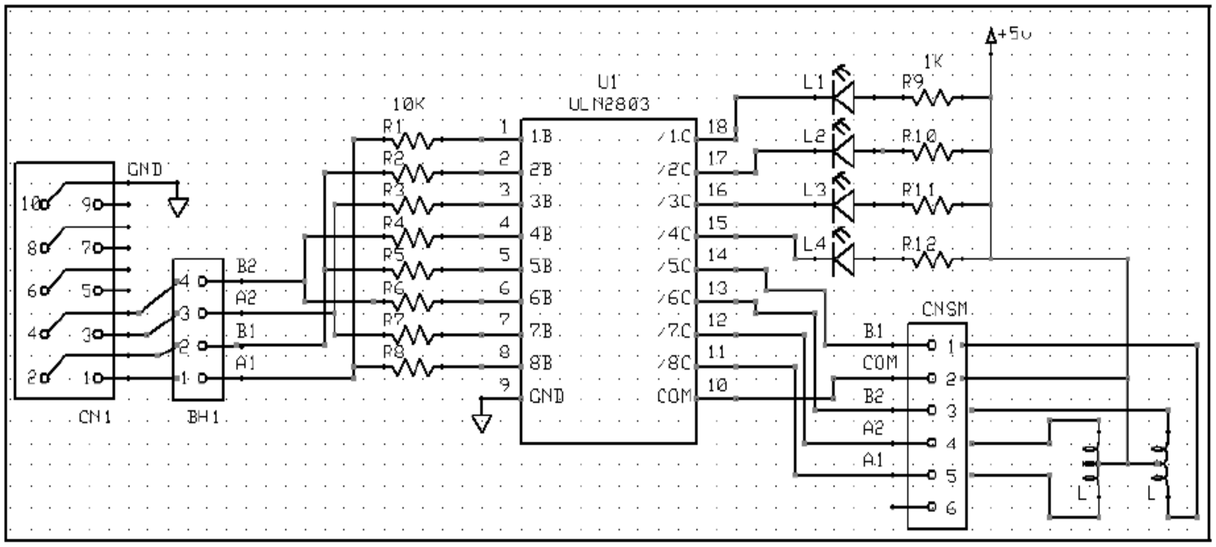
Stepper motors are used in many applications for precise positioning. Disk drives, Printers and Robots are few of the applications. They are called steppers because they rotate in steps of a few degrees for each pulse of voltage applied to one or more of their terminals. The motor used in our kit is PM20S, 18^0/step, 4-phase unipolar motor.

A simple interface to PM20S motor is provided with 10 pin box header.

CN1

B1	B2	NC	NC	GND
A1	A2	NC	NC	NC

Interface Circuit:



Normal Torque Half Step

Sequence	A1	B1	A2	B2
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1
1	1	0	0	0

High Torque Full Step

Sequence	A1	B1	A2	B2
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1
1	1	1	0	0

Full Step:

VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity smotor is
   Port ( dir : in  STD_LOGIC;
          rst : in  STD_LOGIC;
          clk : in  STD_LOGIC;
          a1b1a2b2 : out  STD_LOGIC_VECTOR (3 downto 0));
```

```
end smotor;
architecture Behavioral of smotor is
signal clkdiv: std_logic_vector(15 downto 0);
signal count: std_logic_vector(1 downto 0);
signal sig: std_logic_vector (3 downto 0);
signal clkdiv2:std_logic;
begin
process(clk)
begin
if rising_edge(clk) then
clkdiv<=clkdiv +1;
end if;
  clkdiv2<=clkdiv(15);
end process;
process(clkdiv2,rst,dir)
begin
if(rst='1')then
count<="00";
elsif(rising_edge(clkdiv2))then
if(dir='1')then
count<=count +1;
else
count<=count-1;
end if;
end if;
end process;
process(count)
begin
if(count="00")then
sig<="0011";
elsif count="01"then
```

```
sig<="0110";
elsif count="10"then
sig<="1100";
elsif count="11"then
sig<="1001";
end if;
end process;
a1b1a2b2<=sig;
end Behavioral;
```

UCF(User Constraint File):

```
NET "a1b1a2b2[0]" LOC = P40;
NET "a1b1a2b2[1]" LOC = P41;
NET "a1b1a2b2[2]" LOC = P43;
NET "a1b1a2b2[3]" LOC = P44;
NET "dir" LOC = P2;
NET "clk" LOC = P15;
NET "rst" LOC = P5;
```

Half step:

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity stephsp is
   Port ( dir : in  STD_LOGIC;
          rst : in  STD_LOGIC;
          clk : in  STD_LOGIC;
          a1b1a2b2 : out  STD_LOGIC_VECTOR (3 downto 0));
end stephsp;
```

```
architecture Behavioral of stephsp is
signal clkdiv: std_logic_vector(15 downto 0);
signal count: std_logic_vector(2 downto 0);
signal sig: std_logic_vector (3 downto 0);
signal clkdiv2: std_logic;
begin
process(clk)
begin
if rising_edge(clk) then
clkdiv<=clkdiv +1;
end if;
  clkdiv2<=clkdiv(15);
end process;
process(clkdiv2,rst,dir)
begin
if(rst='1')then
count<="000";
elsif(rising_edge(clkdiv2))then
if(dir='1')then
count<=count +1;
else
count<=count-1;
end if;
end if;
end process;
process(count)
begin
if(count="000")then
sig<="1000";
elsif count="001"then
sig<="1100";
```

```
elsif count="010"then
sig<="0100";
elsif count="011"then
sig<="0110";
elsif count="100"then
sig<="0010";
elsif count="101"then
sig<="0011";
elsif count="110"then
sig<="0001";
elsif count="111"then
sig<="1001";
end if;
end process;
a1b1a2b2<=sig;
end Behavioral;
```

UCF(User Constraint File):

```
NET "a1b1a2b2[0]" LOC = P40;
NET "a1b1a2b2[1]" LOC = P41;
NET "a1b1a2b2[2]" LOC = P43;
NET "dir" LOC = P2;
NET "clk" LOC = P15;
NET "rst" LOC = P5;
NET "a1b1a2b2[3]" LOC = P44;
```

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

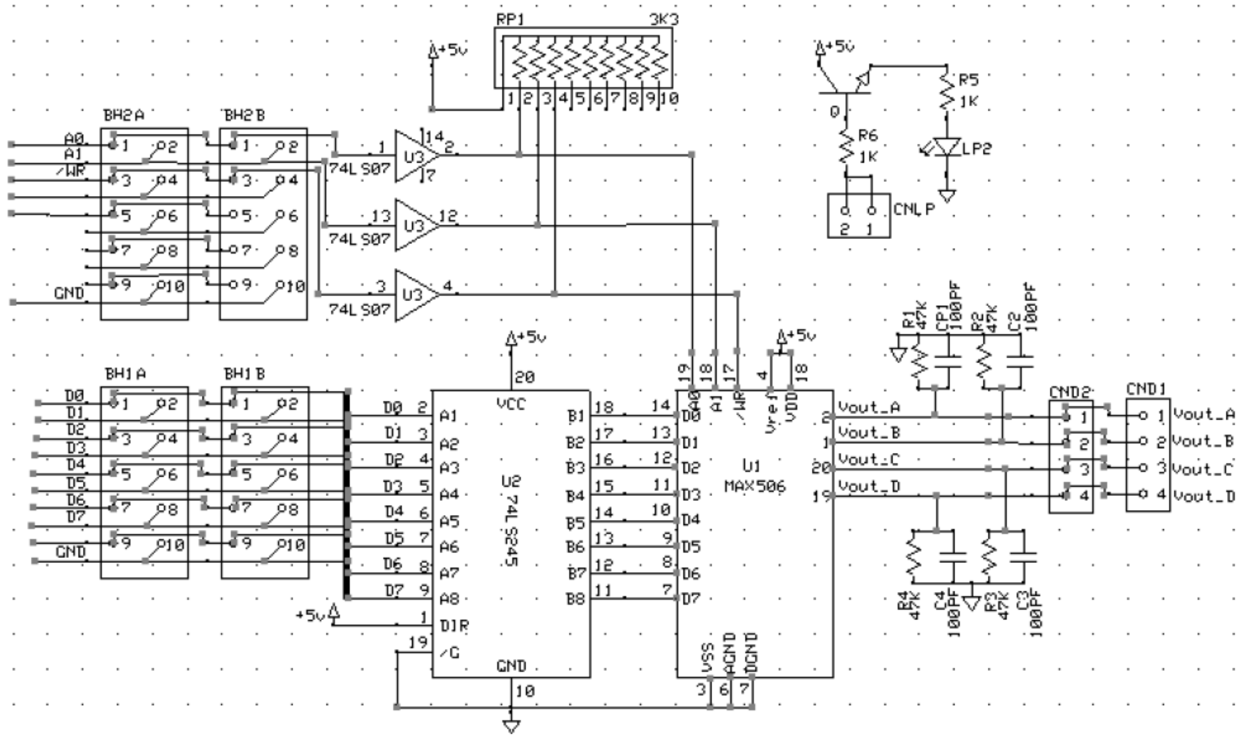
Experiment No: 10a   Date: ____________

SQUARE WAVE

Aim: To Generate Square Wave using DAC (Digital to Analog Converter).

Theory:

Interface Circuit:



VHDL Code

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.all;
entity dac is
   Port ( clk : in  STD_LOGIC;
header1 : out  STD_LOGIC_VECTOR (7 downto 0);
header2 : out  STD_LOGIC_VECTOR (7 downto 0));
end dac;
architecture Behavioral of dac is
signal div: std_logic_vector(5 downto 0);
signal clk100k: std_logic;
signal count8: std_logic_vector(7 downto 0);
signal square: std_logic_vector(7 downto 0);
signal a0,a1,wr: std_logic;
begin
process(clk)
begin
if (rising_edge (clk)) then
div<=div+1;
end if;
end process;
clk100k<=div(3);
process(clk100k)
begin
if(rising_edge (clk100k)) then
count8<=count8 + 1;
if(count8(2)='1') then
square<=X"FF";
else
square<=X"00";
end if;
end if;
end process;
wr<='0' when (clk100k ='1') else '1';
a0<=count8(0);
a1<=count8(1);
header1<=square;
```

```
header2<=X"0"&"0"&wr&a1&a0;
end Behavioral;
```

Result:

Applications:

Remarks :

Signature of Staff Incharge with date:

Experiment No: 10b   Date: ____________

TRIANGULAR WAVE

Aim: To Generate Triangular Wave using DAC.

VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity tri2 is
   Port ( clk: in  STD_LOGIC;
          rst : in  STD_LOGIC;
          h1 : out  STD_LOGIC_vector(7 downto 0);
          h2 : out  STD_LOGIC_vector(7 downto 0));
end tri2;
architecture Behavioral of tri2 is
signal div: std_logic_vector(24 downto 0);
signal count: std_logic_vector(7 downto 0);
signal a0,a1,wr: std_logic;
signal ud : std_logic:='0';
begin
process(clk, rst)
begin
if(rst='1') then
count<=(others=>'0');
elsif(clk'event and clk='1') then
if ud='0' then
count<= count + 1;
elsif ud='1' then
```

```
count<=count - 1;
end if;
end if;
end process;
process(clk)
begin
if clk'event and clk='1' then
if count = "11111111" then ud<='1';
elsif count="00000000" then ud<='0';
end if;
end if;
end process;
h1<= count;
wr<='0' when (clk ='1') else '1';
h2<= X"0"&"0"&wr&a1&a0;
end Behavioral;
```

UCF(User Constraint File):

```
NET "h1[0]" LOC = P2;
NET "h1[1]" LOC = P5;
NET "h1[2]" LOC = P6;
NET "h1[3]" LOC = P7;
NET "h1[4]" LOC = P8;
NET "h1[5]" LOC = P9;
NET "h1[6]" LOC = P10;
NET "h1[7]" LOC = P11;
NET "h2[0]" LOC = P40;
NET "h2[1]" LOC = P41;
NET "h2[2]" LOC = P43;
NET "h2[3]" LOC = P44;
NET "h2[4]" LOC = P45;
NET "h2[5]" LOC = P46;
 NET "h2[6]" LOC = P50;
NET "h2[7]" LOC = P51;
NET "clk" LOC = P15;
NET "rst" LOC = P23;
```

Result:

Applications:

Remarks :

Signature of Staff Incharge with date

Experiment No: 10c Date: ___________

RAMP WAVE

Aim: To Generate Ramp Wave using DAC.

VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ramp is
   Port ( clk : in  STD_LOGIC;
header1 : out  STD_LOGIC_VECTOR (7 downto 0);
header2 : out  STD_LOGIC_VECTOR (7 downto 0));
end ramp;
architecture Behavioral of ramp is
signal div: std_logic_vector(5 downto 0);
signal clk100K: std_logic;
signal count8 : std_logic_vector(7 downto 0);
signal a0,a1,wr: std_logic;
begin
process(clk)
begin
if rising_edge(clk) then
div<=div+1;
end if;
end process;
clk100K<=div(5);
p1: process(clk100k)
begin
if (rising_edge(clk100k)) then
```

```
count8<=count8+1;
end if;
end process P1;
wr<= '0' when (clk100k='1') else '1';
header1<=count8;
header2<=X"0" & '0' &wr& a1 & a0;
end Behavioral;
```

Result:

Applications:

Remarks :

Signature of Staff Incharge with date:

Experiment No: 11 Date: ___________

RELAY

Aim: To write a HDL code to control the external lights using relays.

VHDL Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity relay is
port (   p: in std_logic;
         m: out std_logic);
end relay;

architecture behav of relay is
begin
process(p)
begin
         if(p='1') then
         m<='1';
         else
         m<='0';
         end if;
end process;
end behav;
```

Result:

Applications:

Remarks :

Signature of Staff Incharge with date:

HDL LABORATORY (17EC5DLHDL)

PROBABLE/SUGGESTED QUESTION BANK

1. Expand VHDL.What is the difference between VHDL and Verilog?
2. What is the difference between (i) signal and variable (ii) generic & Parameter (iii) function & procedure (iv) task & function (v) always & initial (vi) register & variable (vii) signal & wire.
3. What are the different styles of models in VHDL and Verilog?
4. What are the operators in VHDL & Verilog?
5. Which operator is having most priority?
6. What is meant by sensitivity list?
7. Give the Following syntax in HDL (i) if, for, function, procedure (ii) while, case.
8. What is the operating frequency of your FPGA?
9. Expand FPGA & ASIC.
10. What are the data types in VHDL?
11. What are the data types in Verilog? What is delta time?
12. What is the difference between (0 to 3) & (3 downto 0)?
13. Write the truth table & Excitation table for D filp flop, SR , T, JK
14. What are the file operations in Verilog?
15. What is meant by synthesis?
16. Write the flow chart for synthesis process?
17. What is the difference between combination circuit & sequential circuit?
18. What is the difference between latch & Flip flop?Write a Verilog code to swap contents of two registers with and without a temporary register?
19. In a pure combinational circuit is it necessary to mention all the inputs in sensitivity list? If yes, why?What is the difference between wire and reg?
20. Give only two xor gates one must function as buffer and another as not gate?
21. Build a 4:1 mux using only 2:1 mux? What are shift operators in HDL?
22. What are the logical operators in VHDL & Verilog?
23. What is the gate density of your FPGA?

24. What is data flow model, structural model, behavioral model? How you invoke from VHDL to Verilog and vice versa?
25. What is the difference between SR flip flop & JK flip flop?
26. What is the difference between synchronous reset & Asynchronous reset?
27. What is the difference between stepper motor & DC motor?
28. What is the step size of stepper motor?
29. What is mux and demux?
30. What is encoder and decoder?
31. What is the difference between encoder & priority encoder?
32. What is binding?
33. What is the difference between "bit" and "std_logic"?
34. What are the std_logic values?
35. What are the different types of buffers are in Verilog HDL?
36. What is the difference between dc motor and stepper motor?
37. What are the applications of dc motor and stepper motor?
38. Write the syntax for casex and casez. What is screen time?
39. What are the VHDL file processing?
40. Draw the simulation waveform for D-latch using signal assignment and variable assignment statements inside the process.
41. Give the syntax for arrays in VHDL and Verilog.

References:

1. http://ece.niu.edu.tw/~chu/download/fpga/verilog.pdf
2. http://www.ics.uci.edu/~alexv/154/VHDL-Cookbook.pdf
3. http://access.ee.ntu.edu.tw/course/dsd_99second/2011_lecture/W2_HDL_Fundamentals_2011-03-02.pdf
4. Stephen Brown and Zvonko Vranesic, "Fundamentals of Digital Logic Design with VHDL", Second Edition, The McGraw-Hill, 2009.
5. Nazeih M.Botros, "HDL Programming (VHDL and Verilog)", John Wiley India Pvt. Ltd. 2008.

6. Samir Palnitkar, “Verilog HDL-A guide to digital design and synthesis”, Sunsoft Press, 1996
7. Charles H Roth Jr., “Digital System Design using VHDL”, PWS Publishing Company.