UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advanced Engineering Mathematics **Subject Code-MVLSI101**
Year: 1st Year **Semester:First**

Module Number	Topics	Number of Lectures (33)
1	**Complex Variables:**	**8L**
	Review ofcomplex variables	1
	Conformal mapping &transformations, Function ofcomplex variables, Poleandsingularity, Integrationwithrespecttocomplexargument,Residuesandbasictheoremsonresidues	7
2	**NumericalAnalysis**	**8L**
	Introduction, Interpolationformulae, Differenceequation,Rootsofequations, Solutionofsimultaneous linear	3
	Non-linearequations,SolutiontechniquesforODEandPDE,Introductiontostability,MatrixEigenvalueandEigenvectorproblems.	5
3	**OptimizationTechnique**	**9L**
	Calculus ofseveral variables, Implicit function theorem, Nature ofsingular points, Necessary and sufficient conditions for optimization	4
	Elementsof calculusvariation,ConstrainedOptimization	2
	Lagrangemultipliers,Gradientmethod,Dynamic programming	3
4	**Probability and Statistics**:	**8L**
	Definition and postulates of probability, Field of probability, Mutually exclusive events, Byes' Theorem, Independence, Bernoulli trial, Discrete Distributions, Continuous distributions	4
	Probable errors, Linear regression, Introduction to non-linear regression, Correlation, Analysis of variance.	4

UNIVERSITY OF ENGINEERING AND MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: VLSI Devices and Modelling **Subject Code-MVLSI 102**
Year: 1ST Year **Semester: 1ST**

Module Number	Topics	Number of Lectures
1	**Semiconductors, Junctions and MOSFET Overview:**	**3L**
	Introduction, Semiconductors, Conduction, Contact Potentials,	2L
	P-N Junction, Overview of the MOS Transistor.	1L
2	**Two Terminal MOS Structure:**	**4L**
	Flat-band voltage, Potential balance & charge balance,	1L
	Effect of Gate-substrate voltage on surface condition,	2L
	Inversion, Small signal capacitance;	1L
3	**Three Terminal MOS Structure:**	**3L**
	Contacting the inversion layer, Body effect,	2L
	Regions of inversion , Pinch-off voltage	1L
4	**Four Terminal MOS Transistor:**	**3L**
	Transistor regions of operation, general charge sheet models, regions of inversion in terms of terminal voltage,	1L
	strong inversion , weak inversion, moderate inversion, interpolation models,	1L
	effective mobility, temperature effects, break down p-channel MOSFET	1L
5	**CMOS Device Design**	**2L**
	Scaling, Threshold voltage, MOSFET channel length	2L
6	**CMOS Performance Factors**	3L
	Basic CMOS circuit elements; parasitic elements;	1L
	sensitivity of CMOS delay to device parameters	1L
	performance factors of advanced CMOS device	1L
7	**Bipolar Devices, Design & Performance**	2L

Faculty In-Charge HOD, ECE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Digital IC Design **Subject Code-MVLSI 103**
Year: 1STYear **Semester: 1ST**

Module Number	Topics	Number of Lectures
1	**Specification Methods**	**8L**
	1. Language based methods including VHDL	4L
	2. Hierarchical state machine descriptions such as State Charts and Petri net based methods.	3L
	3. Functional languages for formal verification.	1L
2	**Synthesis tools**	**3L**
	1. High level synthesis	1L
	2. Scheduling allocation	1L
	3. communication and control	1L
3	**Module Generation**	**8L**
	1. Finite State machines	2L
	2. State encoding	1L
	3. Parameterized blocks PLA, RAM, ROM generation.	2L
	4. Gate Level Synthesis	1L
	5. Binary Decision Diagrams	1L
	6. Logic minimization, optimization and retargeting.	1L
4	**Layout Synthesis**	**5L**
	1. Placement; simulated annealing	1L
	2. Genetic algorithms, constructive methods	1L
	3. Routing; nets, layers	1L
	4. Lees algorithms	1L
	5. Cost functions, channel routing. Examples of a channel router with placement expansion.	1L
5	**Complex gates**	**12L**
	1. Pseudo NMOS; dynamic logic	2L
	2. Dynamic cascaded logic.	1L
	3. Domino logic; 2 and 4 phase logic	2L
	4. Pass transistor logic	1L
	5. Control and timing	1L
	6. Synchronous and asynchronous	1L
	7. Self-timed systems;	1L
	8. Multi-phase clocks	1L
	9. Register transfer; examples of ALU	1L
	10. Shifters, and registers	1L
6	**Effects of scaling circuit dimensions**	**2L**
	1. Physical limits to develop fabrication	2L
Total Number Of Hours = 38L		

Faculty In-Charge HOD, ECE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT,JAIPUR

Lecture-wise Plan

Subject Name: Microelectronics Technology & IC Fabrication Subject Code-MVLSI 104
Year: 1STYear Semester: 1ST

Sl. No	*Topic(S)*	
1	***Module – 1:*** **Cleanroom technology**	**4L**
	Clean room concept	1L
	Growth of single crystal Si	1L
	Surface contamination	1L
	Cleaning & etching	1L
2	***Module – 2:*** **Oxidation**	**9L**
	Growth mechanism and kinetic oxidation	1L
	Oxidation techniques and systems	1L
	Oxide properties	1L
	Oxide induced defects	1L
	Characterization of oxide films	1L
	Use of thermal oxide and CVD oxide	1L
	Growth and properties of dry and wet oxide	1L
	Dopant distribution	1L
	Oxide quality	1L
3	***Module 3:*** **Solid State Diffusion**	**4L**
	Day 14: Fick's equation	1L
	Day 15: Atomic diffusion mechanisms	1L
	Day 16: Measurement techniques	1L
	Day 17: Diffusion in polysilicon and silicon dioxide diffusion systems	1L
4	***Module -4:*** **Ion implantation**	**4L**
	Day 18: Range theory	1L
	Day 19: Equipments	1L
	Day 20: Annealing, shallow junction	1L
	Day 21: High energy implementation	1L
5	***Module -5:*** **Lithography**	**2L**
	Day 22: Optical lithography	1L
	Day 23: Some advanced lithographic techniques	1L
6	***Module -6:*** **Physical Vapor Deposition**	**3L**
	Day 24: APCVD	1L
	Day 25: Plasma CVD	1L
	Day 26: MOCVD	1L
7	***Module -7:*** **Metallization**	**3L**
	Day 27: Different types of metallization	1L
	Day 28: Uses & desired properties	1L
	Day 29: VLSI Process integration	1L

Faculty In-Charge HOD, ECE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lecture-wise Plan

Subject Name: Advanced Digital Communication Subject Code: **MVLSI-105**
Year: **1st Year** Semester: **1st**

Module Number	Topics	Number of Lectures
1	**Spectral analysis of signals:**	**4L**
	1. Orthogonal & orthonormal signals. Gram-Schmidt procedure to represent a set of arbitrary signals by a set of orthonormal components; - numerical examples.	2
	2. The concept of signal-space coordinate system, representing a signal vector by its orthonormal components, measure of distinguishability of signals.	2
2	**Characteristics of random variables and random processes:**	**8L**
	1. Common probability density functions, - Gaussian, Rayleigh, Poisson, binomial, Rice, Laplacian, lognormal	3
	2. Probability of error in Gaussian Binary symmetric channel.	1
	3. Random processes – time average, ensemble average, covariance, autocorrelation, cross correlation, stationary process, ergodic process, wide sense stationary process. 4. Power spectral density and autocorrelation, power spectral density of a random binary signal.	4
3	**Source coding:**	**10L**
	Sampling theorem, instantaneous/ flat top/ natural sampling, band width of PAM signal, quantization, quantization noise, principle of pulse code modulation, delta modulation & adaptive delta modulation. Parametric coding/ hybrid coding/ sub band coding: APC, LPC, Pitch predictive, ADPCM, voice excited vocoder, vocal synthesizer.	5
	1. UPNRZ, PNRZ, UPRZ, PRZ, AMI, Manchester etc. 2. Calculation of their power spectral densities. 3. Bandwidths and probabilities of error Pe for different line codes. 4. Principle, transmitter, receiver, signal vectors, their distinguish ability and signal band width for BPSK, QPSK, M-ARY PSK, QASK, MSK, BFSK, M-ARY FSK.	5

4	**Spread spectrum modulation:**	**10L**
	1. Principle of DSSS, processing gain, jamming margin, single tone interference, principle of CDMA, MAI and limit of number of simultaneous users.	3
	2. Digital cellular CDMA system: model of forward link, reverse link, error rate performance of decoder using m-sequence chip codes.	3
	3. Properties of m-sequences, their generation by LFSR, their PSDs, limitations of m sequences.	2
	4. Gold sequence, Kasami sequence – generating the sequences, their characteristic mean, cross correlation and variance of cross correlation, their merits and limitations as chip codes in CDMA	2
5	**Multiplexing & multiple access:**	**2L**
	1. TDM/TDMA, FDM/FDMA, Space DMA, Polarization DMA, OFDM, ALOHA, Slotted ALOHA, Reservation ALOHA, CSMA-CD, CSMA-CA – basic techniques and comparative performances e.g. signal bandwidth, delay, probability of error etc.	2
6	**Noise:**	**3L**
	1. Representation of noise in frequency domain. 2. Effect of filtering on the power spectral density of noise – Low pass filter, band pass filter, differentiating filter, integrating filter. 3. Quadrature components of noise, their power spectral densities and probability density functions. Representation of noise in orthogonal components.	3
7	**Characteristics of different types of channels:**	5L
	1. Gaussian, Poisson etc. **Band limited channel:** 2. Characteristics of band limited channel, inter symbol interference (ISI) - it's mathematical expression. 3. Niquist's theorem for signal design for no ISI in ideal band limited channel, Niquist's criteria, raised cosine pulse signals. 4. Signal design for controlled ISI in ideal band limited channel, partial response signals, duobinary & partial duobinary signals - their methods of generation and detection of data.	5L

	5. Concept of maximum likelihood detection, log likely hood ratio. 6. Detection of data with controlled ISI by linear transverse filters. 7. Performance of minimum mean square estimation (MMSE) detection in channels with ISI.	
8	**Base band signal receiver and probabilities of bit error:**	**5L**
	1. Peak signal to RMS noise output ration, probability of error. 2. Optimum filter, its transfer function. 3. Matched filter, its probability of error. 4. Probability of error in PSK, effect of imperfect phase synchronization or imperfect bit synchronization. 5. Probability of error in FSK, QPSK. 6. Signal space vector approach to calculate probability of error in BPSK, BFSK, QPSK. 7. Relation between bit error rate and symbol error rate. 8. Comparison of various digital modulation techniques vis-à-vis band width requirement and probabilities of bit error.	5L
Total Number Of Hours = 45L		

Faculty In-Charge HOD, ECE Dept.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Lab Manual

Title of Course: CAD Tools for VLSI Design
Course Code: MVLSI191
L-T-P scheme: 0-0-3 **Course Credit: 4**

Objectives: The overall course objective is to teach electrical engineering students fundamental concepts of hardware description languages and advanced techniques in digital system design. Specific objectives include the following:

1. Learn VHDL (Very high speed integrated circuit Hardware Description Language).
2. Utilize VHDL to design and analyse digital systems including arithmetic units and state machines.
3. Learn field programmable gate array (FPGA) technologies and utilize associated computer aided design (CAD) tools to synthesize and analyse digital systems.
4. Learn testing strategies and construct test-benches.
5. Conduct laboratory experiments using an FPGA based development board to prototype digital systems and to confirm the analysis done in class.
6. Prepare informative and organized lab reports that describe the methodologies employed, the results obtained, and the conclusions made in a laboratory experiment.

Learning Outcomes: The students will have a detailed knowledge of the concepts of IEEE and ANSI standard HDL. Upon the completion of Operating Systems practical course, the student will be able to:

- **Understand** and implement basic digital logic circuits of VLSI.
- **Model** complex digital systems at several levels of abstractions; behavioural and structural, synthesis and rapid system prototyping.
- **Developand Si**mulate register-level models of hierarchical digital systems.
- **Designand model** complex digital system independently or in a team
- Carry out **implementations** of registers and counter**s.**
- **Simulate and synthesize** all type of digital logic circuits used in VLSI.
- Finally **design** a CPU.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1: Design of basic Gates: AND, OR, NOT.
Exercise No. 2: Design of universal gates
Exercise No. 3: Design of XOR and XNOR gate.
Exercise No. 4: Design of 2:1 MUX.
Exercise No. 5: Design of 2 to 4 Decoder.
Exercise No. 6: Design of Half-Adder and Full Adder.
Exercise No. 7: Design of 8:3 Priority Encoder.
Exercise No. 8: Design of 4 Bit Binary to Grey Code Converter.
Exercise No. 9: Design of all Flip-Flops.
Exercise No. 10: Design of Shift register.
Exercise No. 11: Design of ALU.

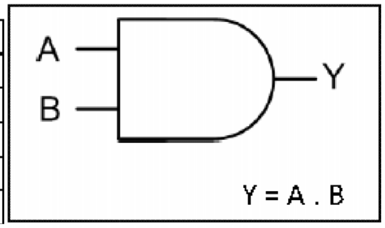
Text Book:
1. J. Bhaskar, A VHDL Primer, 3rd edition, Prentice Hall.

Recommended Systems/Software Requirements:

1. Intel based desktop PC with minimum of 1GHZ or faster processor with at least 1GB RAM and 8 GB free disk space.
2. Xilinx ISE14.2 software in Windows XP or Linux Operating System.

Experiment No: 1Design of basic Gates: AND, OR, NOT.
Aim: Write VHDL code for basic gates: AND, OR, NOT.
Apparatus: Xilinx ISE 14.2 software
AND Gate

2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

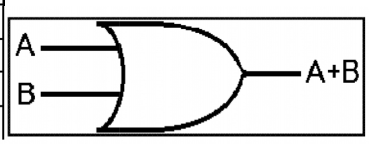


VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
 entity AND1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end AND1;
architecture behavioral of AND1 is
 begin
process (a, b)
begin
if (a= “1”and b=“1”)
then c<=“1”; else c<=“0”;
end if;
end process;
end behavioral;
```

OR Gate:

2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1



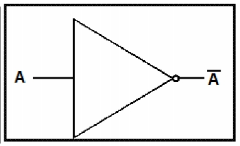
VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity OR1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end OR1;
architecture behavioral of OR1 is
begin
process (a, b)
begin
if (a=“0”and b=“0”) then c<=     “0”; else c<=“1”;
```

```
end process;
end behavioral;
```

NOT Gate:

NOT gate	
A	$\overline{A}$
0	1
1	0



VHDL Codes:

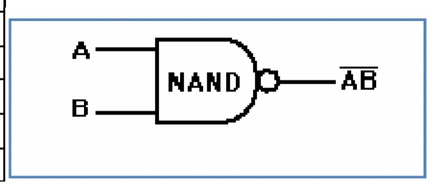
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity NOT1 is
port (a : in STD_LOGIC; c : out STD_LOGIC) ;
end NOT1;
architecture behavioral of NOT1 is
 begin
 process (a)
begin
 if (a=“0”) then c<=“1”;
else c<=“0”;
end if;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.
Output: Student will check the output.
Discussion: Student will conclude here.

Experiment No.-2 : Design Universal gates
Aim: Write VHDL code for universal gates: NAND and NOR gate.
Apparatus: Xilinx ISE14.2 software

NAND gate:

2 Input NAND gate		
A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0



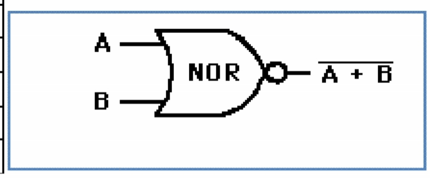
VHDL codes:

```
library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
 use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity NAND1 is
```

end NAND1;
architecture
behavioral of NAND1 is
begin
process (a, b)
begin
if (a= “1”and b=“1”)then c<=   “0”;
else c<=“1”;
end if;
end process;
end behavioral;

<u>NOR gate:</u>

2 Input NOR gate		
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0



VHDL codes:
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity NOR1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end NOR1;
architecture behavioral of NOR1 is
begin
process (a, b)
begin
if (a=“0”and b= “0”) then c<=“0”;
else c<=“0”;
end if;
end process;
end behavioral;
Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.
Output: Student will check the output.
Discussion: Student will conclude here.

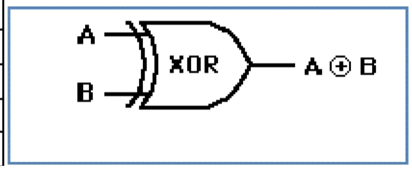
Experiment No.-3 : Design XOR and XNOR gate

Aim: Write VHDL code for XOR and XNOR gate.

Apparatus: Xilinx ISE 14.2 software

<u>XOR gate:</u>

2 Input EXOR gate		
A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0



VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity XOR1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end XOR1;
architecture behavioral of XOR1 is
 begin
process (a, b)
variable (s1, s2, s3, s4:STD_LOGIC)
begin
s1:=NOT a;
 s2:=NOT b;
s3:=s1 AND b;
 s4:=s2 AND a;
 c<=s3 OR s4;
end process;
 end behavioral;
```

XNOR gate:

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	1

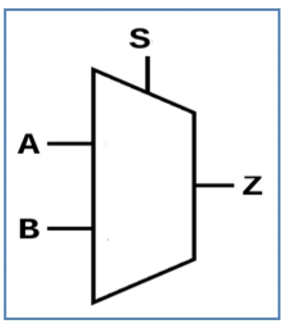
VHDL codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity XNOR1 is
port (a : in STD_LOGIC; b : in STD_LOGIC; c : out STD_LOGIC) ;
end XNOR1;
architecture behavioral of XNOR1 is
```

```
process (a, b)
variable (s1, s2, s3, s4:STD_LOGIC)
begin
s1:=NOT a;
 s2:=NOT b;
 s3:=a AND b;
 s4:=s1 AND s2;
 c<=s3 OR s4;
end process;
 end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.
Output: Student will check the output.
Discussion: Student will conclude here.

Experiment No.-4: Design 2:1 MUX
Aim: Write VHDL code for 2:1 mux using other basic gates.
Apparatus: Xilinx ISE 14.2 software
2:1 MUX:
A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.



S	Z
0	A
1	B

$$Z = A\bar{S} + BS$$

VHDL Codes:

```
library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
 use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity mux_2 to 1 is
 port (a : in STD_LOGIC; b : in STD_LOGIC; s : in STD_LOGIC z : out STD_LOGIC) ;
end mux_2 to 1;
architecture behavioral of mux_2 to 1 is
begin
process (a, b, s)
begin
if (s=“0”)then
z<=a;
else z<=b;
end if;
end process;
```

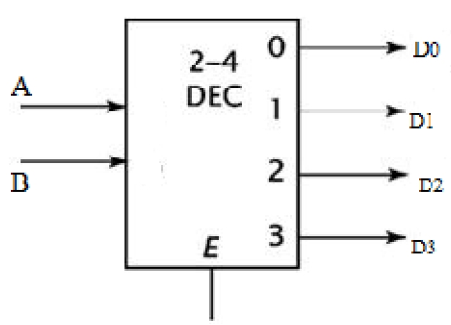
2:1 mux using BASIC gates:

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 use IEEE.STD_LOGIC_UNSIGNED.ALL;
 entity mux_2 to 1 is
 port (a : in STD_LOGIC; b : in STD_LOGIC; s : in STD_LOGIC z : out STD_LOGIC) ;
end mux_2 to 1 ;
architecture behavioral of mux_2 to 1 is
 begin
process (a, b, s)
variable (s1, s2, s3:STD_LOGIC)
begin s1:=NOT s; s2:=s1 AND a; s3:=s AND b; z<=s2 OR s3;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.
Output: Student will check the output.
Discussion: Student will conclude here.

Experiment No.-5 :Design 2:4 Decoder
Aim: Write VHDL code for 2:4 decoder.
Apparatus: Xilinx ISE 14.2 software
2:4 decoder**:** A decoder is a combinational circuit that converts binary information from n inputs line to a maximum of 2^n unique output lines.



E	A	B	D0	D1	D2	D3
0	X	X	0	0	0	0
1	0	0	0	0	0	1
1	0	1	0	0	1	0
1	1	0	0	1	0	0
1	1	1	1	0	0	0

VHDL Codes:

```
library IEEE;
 use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder_2_to_4 is
port (a : in STD_LOGIC_VECTOR; E : in STD_LOGIC; d : out STD_LOGIC_VECTOR (3 downto
0) ;
end decoder_2_to_4 ;
architecture behavioral of decoder_2_to_4 is
```

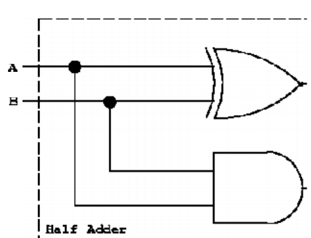
```
 case a is when “00”=> d<=“0001”;
when “01”=> d<=“0010”;
when “10”=> d<=“0100”;
when others=>d<=“1000”;
end case;
end process;
end behavioral;
```

Using DATA_FLOW approach

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
 use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder_2_to_4 is
 port (a : in STD_LOGIC; b : in STD_LOGIC; E : in STD_LOGIC d : out STD_LOGIC_VECTOR (3
downto 0)) ;
end decoder_2_to_4;
architecture dataflow of decoder_2_to_4 is
 signal(abar, bbar: STD_LOGIC)
begin
abar<=NOT a;
 bbar<=NOT b;
 d(0)<=abar AND bbar AND E;
 d(1)<=abar AND b AND E;
 d(2)<=a AND bbar AND E;
d(3)<=a AND b AND E;
end dataflow;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.
Output: Student will check the output.
Discussion: Student will conclude here.

Experiment No.-6: Design Half adder and Full adder
Aim: Write VHDL code for Half-adder, full-adder.
Apparatus: Xilinx ISE 14.2 software
Half-adder:



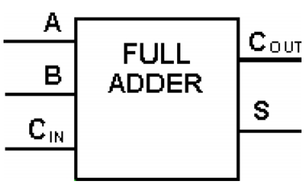
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

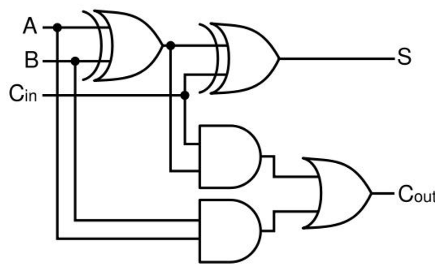
VHDL codes:

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

```
 port (a : in STD_LOGIC; b : in STD_LOGIC; s : out STD_LOGIC; c : out STD_LOGIC);
end half_adder;
architecture behavioral of half_adder is
begin
 process (a,b)
begin
 if (a=“0”and b=“0”) then s<=“0”; c<=“0”;
elsif (a=“1”and b=“1”) s<=“0”; c<=“1”;
else s<=“1”; c<=“0”;
end if;
end process;
end behavioral;
```

Full-Adder:





A	B	C	Sum	Carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

VHDL codes:

```
 library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
 use IEEE.STD_LOGIC_ARITH.ALL;
 use IEEE.STD_LOGIC_UNSIGNED.ALL;
Entity full_adder is
port (a : in STD_LOGIC_VECTOR (0 to 2); s : out STD_LOGIC_VECTOR (0 to 1));
end full_adder;
architecture behavioral of full_adder is
begin
 process (a)
begin
case a is
when “000”=> s<=“00”;
when “001”=> s<=“10”;
when “010”=> s<=“10”;
```

```
when “100”=> s<=“10”;
when “101”=> s<=“01”;
when “110”=> s<=“01”;
when others =>s<=“11”;
end case;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.
Output: Student will check the output.
Discussion: Student will conclude here.

Experiment No-7 : Design 3:8 Decoder
Aim: Write VHDL code for 3:8decoder.
Apparatus: Xilinx ISE 14.2 software
<u>3:8 decoder</u>

Inputs			**outputs**							
A	**B**	**C**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity decoder_3_to_8is
port (a : in STD_LOGIC_VECTOR (2 downto 0);
d : out STD_LOGIC_VECTOR (7 downto 0);
end decoder_3_to_8;
architecture Behavioural of decoder_3_to_8 is
 begin
proess(a)
```

```
case a is
when "000"=> d<="00000001";
 when "001"=> d<="00000010";
when "010"=> d<="00000100";
when "011"=> d<="00001000";
when "100"=> d<="00010000";
 when "101"=> d<="00100000";
 when "110"=> d<="01000000";
when others=>d<="10000000";
 end case;
end process;
end Behavioural;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.
Output: Student will check the output.
Discussion: Student will conclude here.

Experiment No.-8 : Design 8:3 priority encoder
Aim: Write VHDL code for 8:3priority encoder.
Apparatus: Xilinx ISE 8.1 software
8:3 priority encoder:
An encoder is a digital circuit that performs inverse operation of decoder. An encoder has 2^n input lines and n output lines. The output lines generate the binary code corresponding to the input value.
Truth-table for 8:3priority encoder

Inputs								**outputs**		
A7	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**	**D2**	**D1**	**D0**
0	0	0	0	0	0	0	0	X	X	X
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	X	0	0	1
0	0	0	0	0	1	X	X	0	1	0
0	0	0	0	1	X	X	X	0	1	1
0	0	0	1	X	X	X	X	1	0	0
0	0	1	X	X	X	X	X	1	0	1
0	1	X	X	X	X	X	X	1	1	0
1	X	X	X	X	X	X	X	1	1	1

VHDL Codes:

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

entity p_encoder_8_to_3 is
port (a : in STD_LOGIC_VECTOR (7downto 0);
d : out STD_LOGIC_VECTOR (2downto 0));
endp_encoder_8_to_3;
architecture behavioral of p_encoder_8_to_3 is
begin
process (a)
begin
case a is
when “00000001”=> d<=“000”;
 when “0000001X”=> d<=“001”;
when “000001XX”=> d<=“010”;
 when “00001XXX”=> d<=“011”;
 when “0001XXXX”=> d<=“100”;
when “001XXXXX”=>d<=“101”;
when “01XXXXXX”=> d<=“110”;
 when “1XXXXXXX”=> d<= “111”;
 when others=>d<=“XXX”;
end case;
 end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-8: Design Binary to Gray converter

Aim: Design of 4 Bit Binary to Grey code Converter.

Apparatus: Xilinx ISE 14.2 software

<u>Binary top gray converter:</u>

The binary to grey converter is a combinational circuit that takes binary number as input and converts it into grey code. Grey code differs from the preceding and succeeding number by a single bit.

VHDL Codes:

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

entity b2g is
```

```
g : out std_logic_vector (3 downto 0));
end b2g;
architecture behavioral of b2g is
 begin
process (b)
begin
case b is
when “0000” => g<= “0000”;
 when “0001” => g<= “0001”;
when “0010” => g<= “0011”;
when “0011” => g<= “0010”;
 when “0100” => g<= “0110”;
when “0101” => g<= “0111”;
when “0110” => g<= “0101”;
 when “0111” => g<= “0100”;
 when “1000” => g<= “1100”;
when “1001” => g<= “1101”;
 when “1010” => g<= “1111”;
 when “1011” => g<= “1110”;
 when “1100” => g<= “1010”;
 when “1101” => g<= “1011”;
 when “1110” => g<= “1001”;
when others => g<= “1000”;
 end case;
end process;
 end behavioral;
```

Data flow model for binary to grey code converter:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity bin2grey_conv is
            port (b : in std_logic_vector (3 downto0);
             g : out std_logic_vector (3 downto));
end bin2grey_conv;
architecture dataflow of bin2grey_conv is
begin
g(3)<=b(3);
g(2)<=(b(3)) xor (b(2));
 g(1)<=b(2) xor b(1);
 g(0)<=b(1) xor b(0);
end dataflow;
```

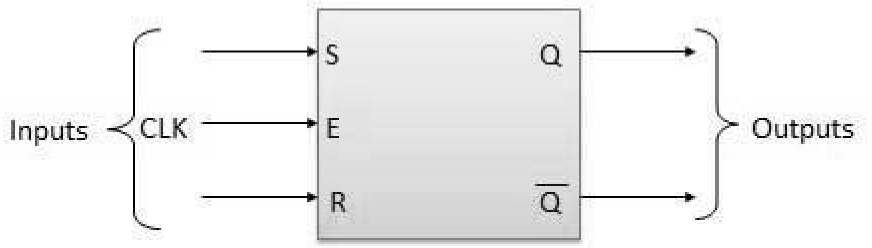
Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-9: Design flip-flops
Aim: Study all Flip-flops using VHDL
Apparatus: Xilinx ISE 14.2 software

(1) S-R flip-flop:



S	R	Qn+1
0	0	Q_n
0	1	0
1	0	1
1	1	

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; entity flipflop_SR is
port (s, r, clk, rst : in std_logic; q : out std_logic);
end flipflop_SR;
architecture behavioral of flipflop_SR is
begin
process (s, r, clk, rst)
 begin
if (clk= “1” and clk‟event) then if (rst= “1”) then
q<= “0”;
elsif (rst= “0”) then
q<= “1”;
elsif (s= “0” and r= “0” and rst= “0”) then
q<=q;
elsif (s= “0” and r= “1” and rst= “0”) then
q<= “0”;
elsif (s= “1” and r= “0” and rst= “0”) then
q<= “1”;
elsif (s= “1” and r= “1” and rst= “0”) then
q<= “U”;
end if;
end if;
end process;
end behavioral;
```

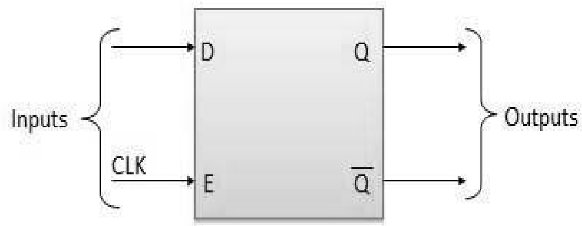
(2) J-K flip-flop:

J	K	Qn+1
0	0	Q_n
0	1	0
1	0	1
1	1	Not Q_n

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; entity flipflop_JK is
port (j, k, clk, rst : in std_logic; q : inoutstd_logic);
end flipflop_JK;
architecture behavioral of flipflop_JK is
begin
process (j, k, clk, rst)
begin
if (clk= “1” and clk‟event) then if (rst= “1”) then
q<= “0”;
elsif (rst= “0”) then
q<= “1”;
elsif (j= “0” and k= “0” and rst= “0”) then
q<=q;
elsif (j= “0” and k= “1” and rst= “0”) then
q<= “0”;
elsif (j= “1” and k= “0” and rst= “0”) then
q<= “1”;
elsif (j= “1” and k= “1” and rst= “0”) then
q<= NOT q;
end if;
end if;
end process;
end behavioral;
```

(3) D flip-flop:



D	Qn+1
0	0
1	1

VHDL Codes:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL; entity flipflop_Dis
port (d,clk, rst : in std_logic; q : inoutstd_logic);
 end flipflop_D;
 architecture behavioral offlipflop_Dis
 begin
 process (d,clk, rst)
 begin
 if (clk= “1” and clk‘‘event) then if (rst= “1”) then
q<= “0”;
else
q<=q;
end if;
end if;
end process;
end behavioral;
```

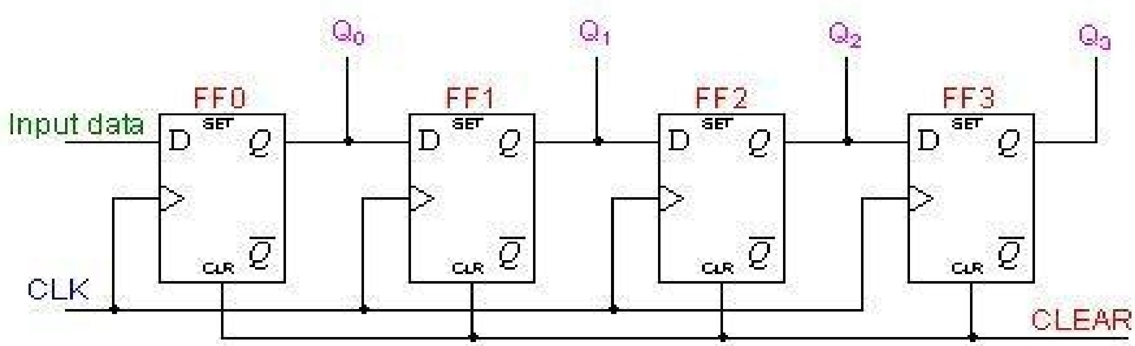
Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No.-10 : Design Shift register
Aim: Design of 8-bit shift register using VHDL.
Apparatus: Xilinx ISE 14.2 software

Shift register



VHDL Codes:

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

```
        port (a : inoutbit_vector (0 to 7);
        r, l, rst, load, clk : in bit;
        q : out bit_vector (0 to 7));
end leftshift;
architecture behavioral of leftshift is
begin
process (load, rst, a, clk)
begin
if (clk= "1" and clk"event) then if (load= "1") then
q<=a;
elsif(load= "0") then if (rst= "1") then q<= "00000000";
else
if (l= "1") then
q<=a slll;
end if;
if (r= "1") then
q<= a srll;
end if;
end if;
end if;
end if;
end process;
end behavioral;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

Experiment No. 11: Design ALU
Aim: Write VHDL program to perform Arithmetic Logic Unit (ALU) operation.
Apparatus: Xilinx ISE 14.2 software
ALU:
An ALU performs arithmetic and logical operation. It receives data from register file and perform operations on it given by control signals generated through control unit.

Sel	Unit	Operation
000	Arithmetic Unit	z <= x
001		z <= x+1
010		z <= y
011		z <= x+y
100	Logic Unit	z <= not x
101		z <= x and y
110		z <= x or y
111		z <= x xor y

VHDL codes:

```
Entity ALU is
Port(x,y : in std_logic_vector(0 to 7);
sel : in std_logic_vector (0 to 2);
z : out std_logic_vector (0 to 7));
```

```
signal arith, logic : std_logic_vector (0 to 7);
 begin
with sel (0 to 1) select
arith <= x when “00”; x+1 when “01”; y when “10”; x+y when others;
with sel (0 to 1) select
logic <= not x when “00”; x and y when “01”; x or y when “10”;
x xor y when others;
with sel (2) select
z <= arith when “0”; logic when others;
end dataflow;
```

Test Bench codes: Student will write/modify test bench codes in Xilinx ISE.

Output: Student will check the output.

Discussion: Student will conclude here.

UNIVERSITY OF ENGINEERING AND MANAGEMENT,JAIPUR
Lab Manual

Title of Course: Embedded System Lab-I
Course Code: MVLSI-192
L-T-P scheme: 0-0-3 **Course Credit: 4**

Objectives:
An embedded system is some combination of computer hardware and software, either fixed in capability or programmable, that is specifically designed for a kind of application device. Industrial machines, automobiles, medical equipment, cameras, household appliances, airplanes, vending machines, and toys (as well as the more obvious cellular phone and PDA) are among the myriad possible hosts of an embedded system. Embedded systems that are programmable are provided with a programming interface, and embedded systems programming is a specialized occupation. Since the embedded system is dedicated to specific tasks, design engineers can optimize it, reducing the size and cost of the product, or increasing the reliability and performance. Some embedded systems are mass-produced, benefiting from economies of scale.

Learning Outcomes: The students will have a detailed knowledge of the concepts ofmicrocontroller and microcontroller based system and students also study the new language like embedded C. Upon the completion of this practical course, the student will be able to:

- **Understand** and implement basic program of embedded C language.
- **Use** the new processor and synchronization libraries in software/ hardware interfaces**.**
- **Study**the benefits to use microcontroller in our real life.
- **Analyze** and simulate the various program.
- **Interface**various hardware interface with 8051 microcontroller.
- **Simulate** the application based program in proteus environment.

Course Contents:
Exercises that must be done in this course are listed below:
Exercise No.1: Write an assembly language program to add, subtract, multiply, divide 16 bit data by Atmel microcontroller.
Exercise No. 2: Write an assembly language program to generate 10 KHz frequency using 8051.
Exercise No. 3: To study the implementation & interfacing of LCD using 8051 microcontroller
Exercise No. 4: To study implementation & interfacing of LED
Exercise No. 5: To study implementation & interfacing of seven segment display
Exercise No. 6: To study implementation & interfacing stepper motor with 8051 microcontroller
Exercise No. 7: To study implementation & interfacing of relay with 8051 microcontroller
Exercise No. 8: To study implementation & interfacing of keypad with 8051 microcontroller
Exercise No. 9: Study of implementation of DC Motor control using PWM method.
Exercise No. 10: Study and observation of Position control of Servo Motor

Text Book:
1. Muhammad Ali Mazidi, J.G. Mazidi, R.D.McKinlay,The 8051 Microcontroller and Embedded Systems, Pearson Prentice Hall.

Recommended Systems/Software Requirements:

Minimum system requirement: -

Processor	:	AMD Athlon ™ 1.67 GH_z
RAM	:	256 MB
Hard Disk	:	40 GB
Mouse	:	Optical Mouse

Hardware requirement: - Microcontroller kit, Interfacing kit, SMPS for microcontroller, Microcontroller burner, Microcontroller AT89C51, etc.

Experiment No: 1

Aim: Write an assembly language program to add, subtract, multiply, divide 16 bit data by Atmel microcontroller.

APPARATUS: M51-02 trainer kit, keyboard and power cord.

PROGRAM:

Addition:

```
ORG 0000H
CLR C               ;make CY=0
MOV A, #0E7H        ; load the low byte now A=E7H
ADD A, #8DH         ; add the low byte now A=74H and CY=1
MOV R6, A           ; save the low byte of the sum in R6
MOV A, #3BH         ; load the high byte
ADDC A, #3BH        ; add with carry (3B+3C+1=78)
MOV R7, A           ; save the high byte of the sum
```

Subtraction:

```
ORG 3000H
CLR C               ; make CY=0
MOV A, #50H         ; load the low byte now A= 50H
MOV R1, #30H        ; load the byte now R1=30H
SUBB A, R1          ; subtract contents of A and R1
JNC Next
CPL A
INC A
Next: MOV R2, A
SJMP 3000H
```

Multiply:

```
ORG 4000H
MOV A, #03H         ; move the first no. into acc
MOV B, #02H         ; move the second no. into B
MUL AB              ; multiply the contents of acc with B
SJMP 4000H
```

Divide:

```
ORG 5000H
MOV A, #25H         ; move the first no. into acc.
MOV B, #5H          ; move the second no. into B
DIV AB              ; divide the contents of acc with B
SJMP 5000H
```

RESULT: Addition, Subtraction, Multiplication, Division of 16-bit data has been performedsuccessfullyon the kit.

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong powersuppliesmay cause damage to your equipment.

Experiment No: 2

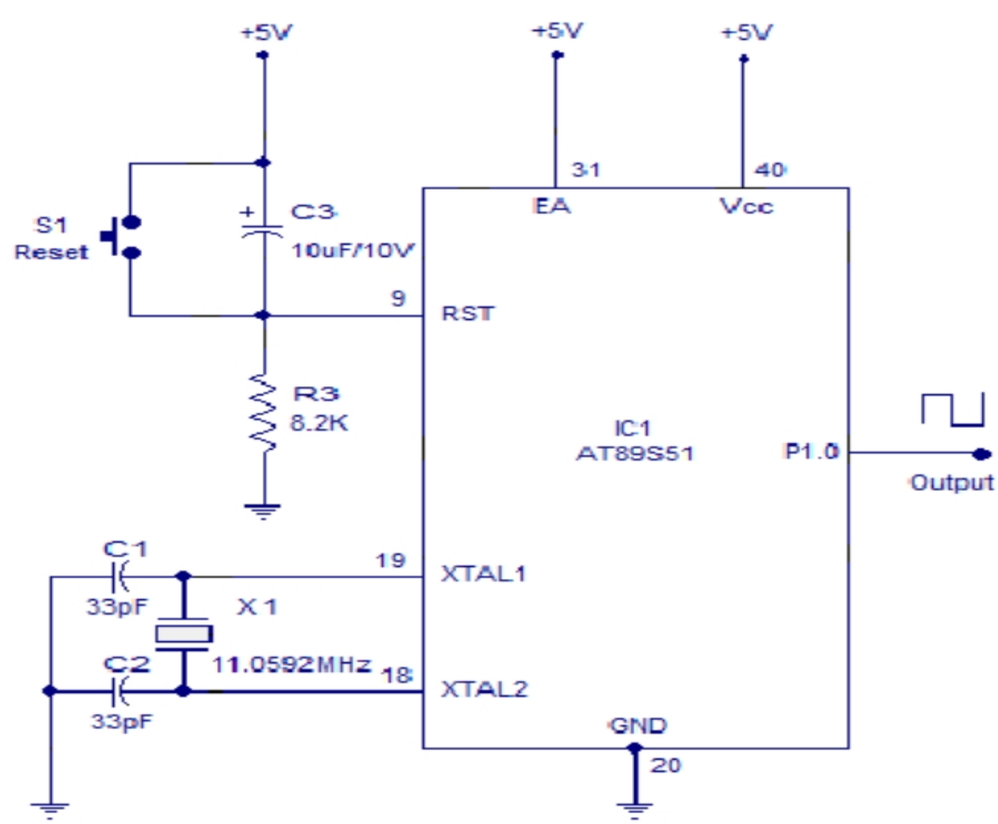
Aim: Write an assembly language program to generate 10 KHz frequency usingAtmel microcontroller.

APPARATUS: E89-01 KIT, power cord, CRO and connecting leads.

Theory: Square waves of any frequency can be generated using the 8051 timer. The technique is verysimple. Write up a delay subroutine with delay equal to half the time of the square wave. Make anyport pin high and call the delay subroutine. After the delay subroutine is finished, make the correspondingport pin low and call the delay subroutine gain. After the subroutine is finished, repeat the cycle again. Theresult will be a square wave of the desired frequency at the selected port pin. The circuit diagram is shown below and it can be used for any square wave, but the program has to be accordingly.

Procedure:
1. Initialize the timer by setting TMOD Register.
2. Load the value in TL1 & TH1 from where Timer starts.
3. Start timer using TR1.
4. Monitor the status of TF1 continuously for an overflow.
5. When overflow occurs stop the Timer.
6. Reset the TF1 flag bit.
7. Go to step2 for next round if required.

Circuit Diagram:



Square wave generation using 8051

Program: -

```
ORG 00H
MOV P0, #01H
AGAIN:
MOV TMOD, #01H
MOV TL0, #0E3H
MOV TH0, #0FFH
SETB TR0
BACK:
JNB TF0, BACK
CLR TR0
CPL P3.4
CPL P3.5
CPL P3.6
CPL P3.7
CLR TF0
JMP AGAIN
END
```

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong powersuppliesmay cause damage to your equipment.

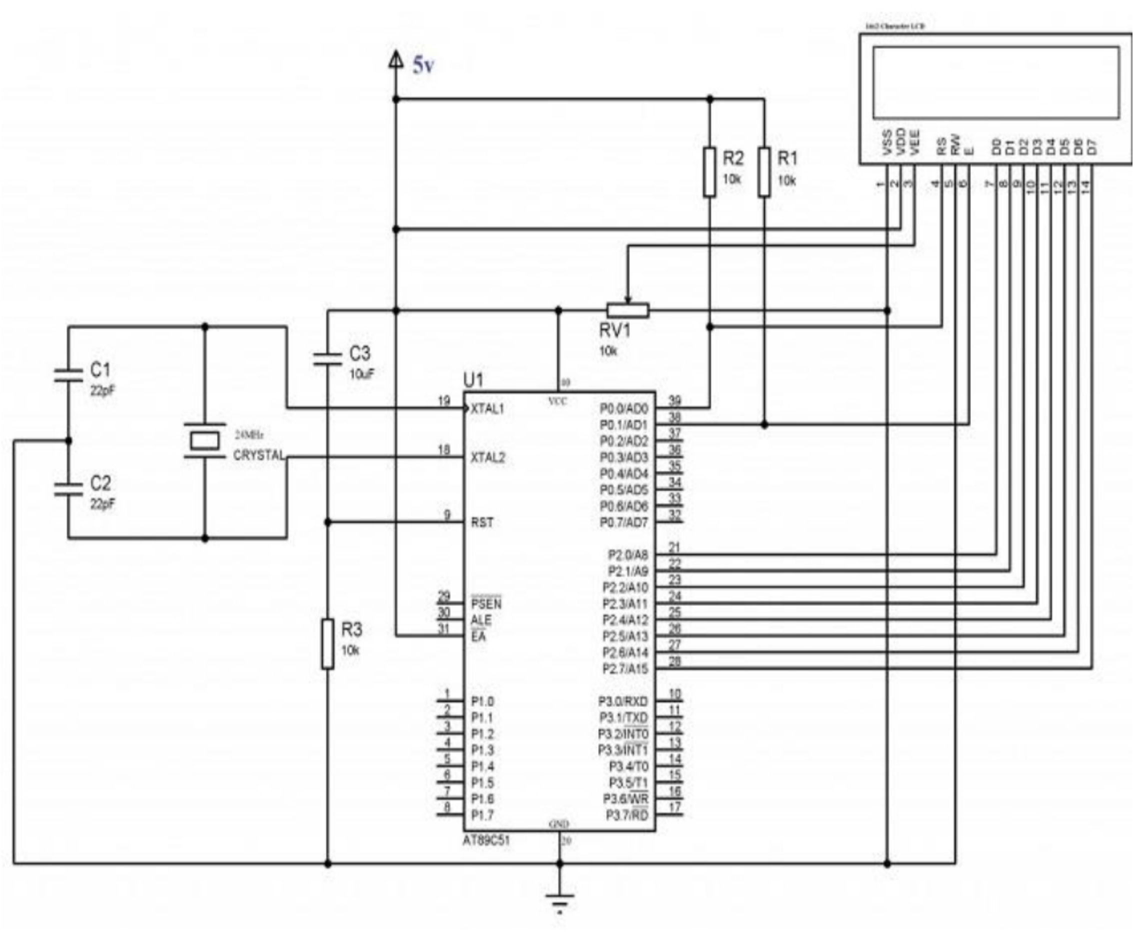
Experiment No: 3

Aim: To study the implementation & interfacing of LCD.

Apparatus Required: Microcontroller kit, Interfacing kit, Keyboard, Monitor, SMPS for Microcontroller.

Theory: Liquid Crystal Display (LCD) is very commonly used electronic display module and having a widerange of applications such as calculators, laptops, mobile phones etc. 16×2 character lcd display is verybasic module which is commonly used in electronics devices and projects. It can display 2 lines of 16characters. Each character is displayed using 5×7 or 5×10-pixel matrix. LCD can be interfaced withmicrocontroller in 4 Bit or 8 Bit mode. These differ in how data is send to LCD. In 8-bit mode to write acharacter, 8 bit ASCII data is send through the data lines D0 – D7 and data strobe is given through E of the LCD. LCD commands which are also 8 bit are written to LCD in similar way. But 4 Bit Mode uses only 4 datalines D4 – D7. In this mode 8-bit character ASCII data and command data are divided into two parts andsend sequentially through data lines. The idea of 4-bit communication is used save pins of microcontroller.4-bit communication is a bit slower than 8-bit communication but this speed difference can be neglectedsince LCDs are slow speed devices.

Circuit Diagram: -



Interfacing of LCD with 8051 microcontroller

Program:

```
ORG 0000H
MOV P0, #00H
LCD_INIT:
MOV A, #01H
ACALL SEND_CMD_LCD
ACALL DELAY
MOV A, #06H
ACALL SEND_CMD_LCD
ACALL DELAY
MOV A,#3CH
ACALL SEND_CMD_LCD
ACALL DELAY
MOV A,#0FH
ACALL SEND_CMD_LCD
ACALL DELAY
MOV A,#82H
ACALL SEND_CMD_LCD
ACALL DELAY
MAIN:
MOV A,#'W'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'E'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'L'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A, #'R'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'C'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'M'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'C'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'E'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'T'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#','
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'B'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'C'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'O'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'M'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'E'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#' '
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'T'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'O'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#0C0H
ACALL SEND_CMD_LCD
ACALL DELAY
MOV A,#'B'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A,#'A'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A, #'H'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A, #'A'
ACALL SEND_DATA_LCD
ACALL DELAY
MOV A, #'L'
ACALL SEND_DATA_LCD
ACALL DELAY
LJMP MAIN
SEND_CMD_LCD:
MOV P0, A
SETB P2.0
CLR P2.1
SETB P2.2
NOP
CLR P2.2
RET
SEND_DATA_LCD:
MOV P0, A
```

```
CLR P2.0
CLR P2.1
SETB P2.2
NOP
CLR P2.2
RET
DELAY:
MOV R0, #255
LOOP: DJNZ R0, LOOP
RET
END
```

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong powersuppliesmay cause damage to your equipment.

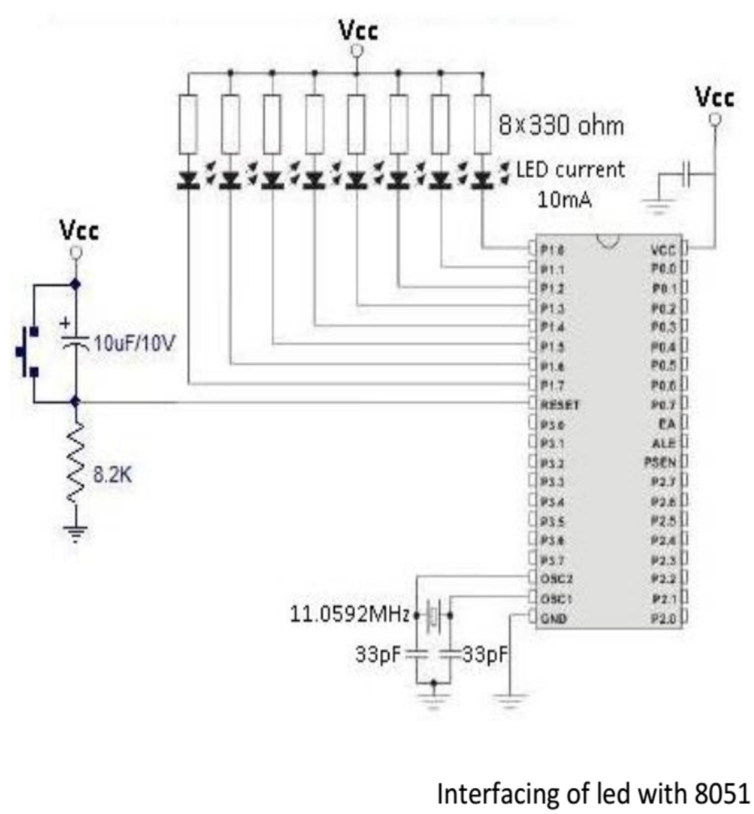
Experiment No: 4

Aim: To study implementation & interfacing of LED

Apparatus Required: Microcontroller kit, Interfacing kit, Keyboard, Monitor, SMPS for Microcontroller.

Theory: Light Emitting Diodes are the semiconductor light sources. Commonly used LEDs will have a cut-offvoltage of 1.7V and current of 10mA. When an LED is applied with its required voltage and current it glows withfull intensity. The Light Emitting Diode is like the normal PN diode but it emits energy in the form of light.The colour of light depends on the band gap of the semiconductor.Thus, LED is directly connected to the AT89C51 microcontroller. The negative terminal of the LED is connectedto the ground through a resistor. Value of this resistor is calculated using the following formula.
R= (V-1.7)/10mA, where V is the input voltage.
Generally, microcontrollers output a maximum voltage of 5V. Thus, the value of resistor calculated for this is 330Ohms. Thus, this can be connected either to the cathode or anode of the LED.

Circuit Diagram:



Interfacing of led with 8051

Program: - org 0000h
mov p0,#00h
main:
mov p0,#55h
acall delay
mov p0,#0aah
acall delay
mov p0,#33h
acall delay
mov p0,#0cch
acall delay
mov p0,#0fh
acall delay
mov p0,#0f0h
acall delay
mov p0,#0ffh
acall delay
sjmp main
org 0000h mov
p0,#00h main:
mov p0,#55h
acall delay
mov p0,#0aah
acall delay
mov p0,#33h
acall delay
mov p0,#0cch
acall delay
mov p0,#0fh
acall delay
mov p0,#0f0h
acall delay
mov p0,#0ffh
acall delay
sjmp main

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong powersuppliesmay cause damage to your equipment.

Experiment No: 5

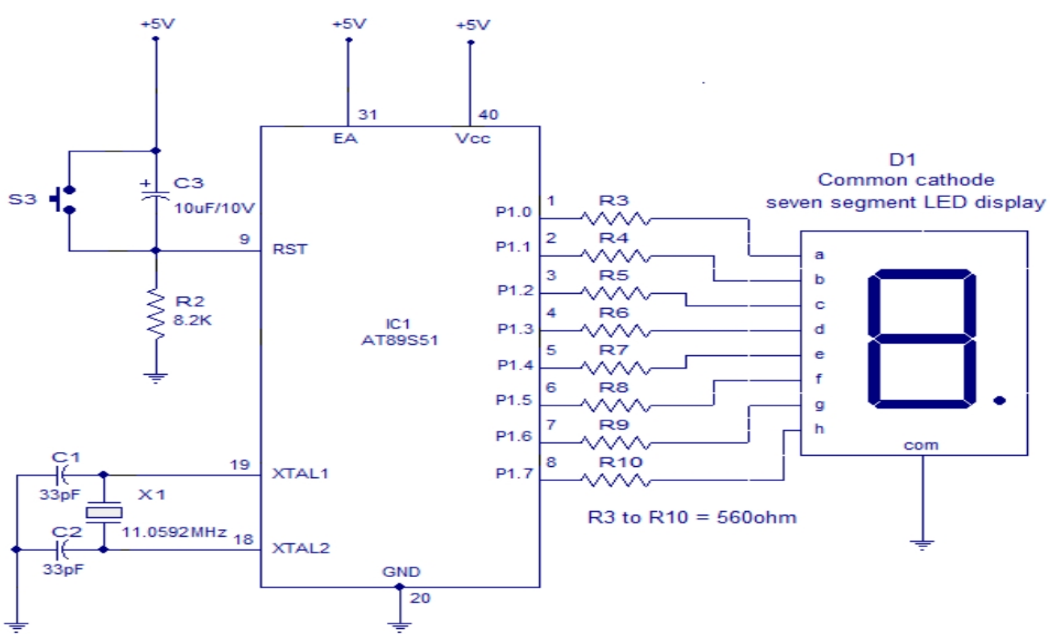
Aim: To study implementation & interfacing of seven segment display

Apparatus Required: Microcontroller kit, Interfacing kit, Keyboard, Monitor, SMPS for Microcontroller.

Theory:
Seven segment displays are used in several systems to display the numeric information. The sevensegment can display one digit at a time. Thus the no. of segments used depends on the no. of digits in thenumber to be displayed. Interfacing seven segment with a controller or MCU is tricky.Digit drive pattern.Digit drive pattern of a seven segment LED display is simply the different logic combinations of itsterminals'a' to 'h' to display different digits and characters. The common digit drive patterns (0 to9) of a seven-segment display are shown in the table below.

Digit	A	b	C	d	E	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

Circuit Diagram:



Interfacing 7 segment display to 8051

Program:

```
ORG 0000H
MOV P2,#00H
MAIN:
MOV P2,#0C0H
ACALL DELAY
MOV P2,#0F9H
ACALL DELAY
MOV P2,#0A4H
ACALL DELAY
MOV P2,#0B0H
ACALL DELAY
MOV P2,#99H
ACALL DELAY
MOV P2,#92H
ACALL DELAY
MOV P2,#82H
ACALL DELAY
MOV P2,#0F8H
ACALL DELAY
MOV P2,#80H
ACALL DELAY
MOV P2,#98H
ACALL DELAY
SJMP MAIN
DELAY:
MOV R7,#10
HERE:MOV R6,#255
HERE1:MOV R5,#255
AGAIN:DJNZ R5,AGAIN
DJNZ R6,HERE1
DJNZ R7,HERE
RET
SJMP MAIN
END
```

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong powersuppliesmay cause damage to your equipment.

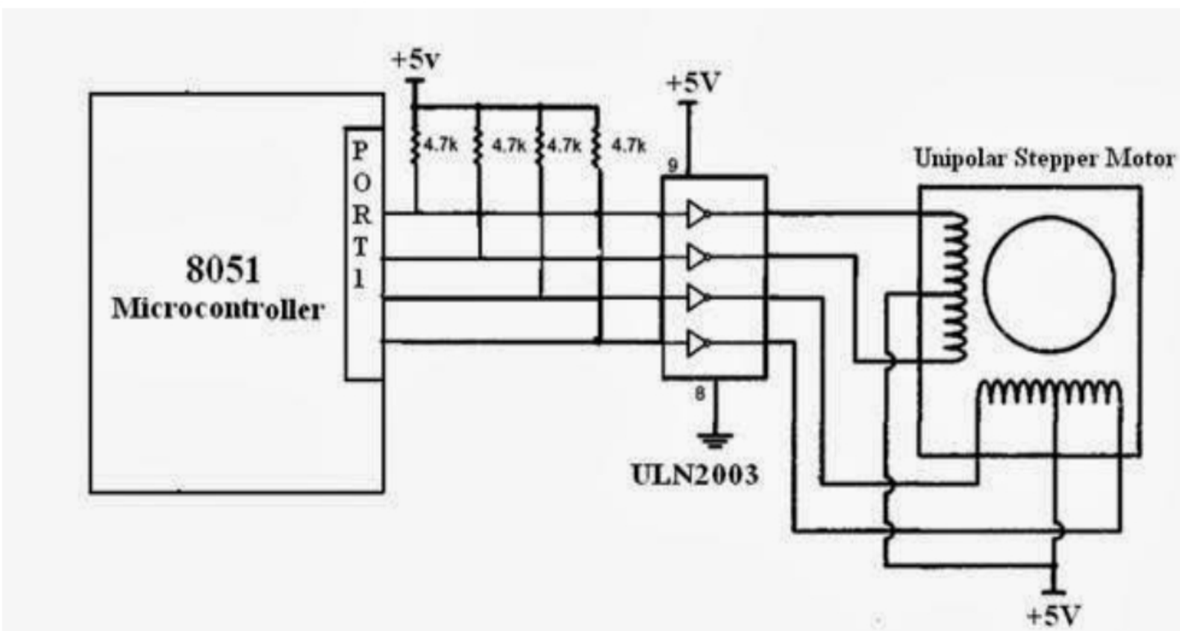
Experiment No: 6

Aim: To study implementation & interfacing stepper motor with 8051 microcontroller.

Apparatus Required: Microcontroller kit, Interfacing kit, Keyboard, Monitor, SMPS for Microcontroller.

Theory: A stepper motor is a brushless and synchronous motor which divides the complete rotation into number of steps. Each stepper motor will have some fixed step angle and motor rotates at this angle. The ULN2003 IC is used to drive the stepper motor as the controller cannot provide current required by the motor. Stepper motor has 6 pins. In these six pins, 2 pins are connected to the supply of 12V and the remaining are connected to the output of the stepper motor. Stepper rotates at a given step angle. Each step-in rotation is a fraction of full cycle. This depends on the mechanical parts and the driving method.

Circuit Diagram:



Program:

```
MOVP0, #01H
ACALL DELAY
MOVP0, #02H
ACALL DELAY
MOVP0, #04H
ACALL DELAY
MOV P0, #08H

ACALL DELAY
LJMP START
DELAY: MOV TMOD, #10H
AGAIN1: MOV R3, #5
AGAIN: MOV TL1, #08H
```

```
SETB TR1
BACK: JNB TF1, BACK
CLR TR1
CLR TF1
```

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong powersuppliesmay cause damage to your equipment.

Experiment No: 7

Aim: To study implementation & interfacing of relay with 8051 microcontroller

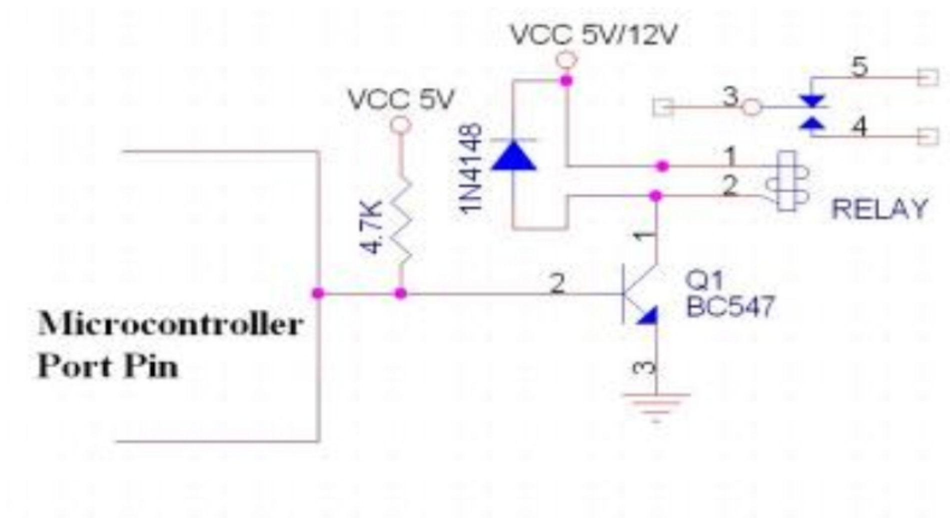
Apparatus Required: Microcontroller kit, Interfacing kit, Keyboard, Monitor, SMPS for Microcontroller.

Theory:

An electromagnetic relay is a switch which is used to switch High Voltage or Current using Low power circuits. It magnetically isolates low power circuits from high power circuits. It is activated by energizing an electromagnet, coil wounded on a soft iron core. A relay should not be directly connected to a microcontroller, it needs a driving circuit due to the following reasons.

- A microcontroller will not able to supply current required for the proper working of a relay. Themaximum current that A89C51 microcontroller can source or sink is 15mA while a relay needsabout 50 – 100mA current.
- A relay is activated by energizing its coil. Microcontroller may stop working by the negative voltagesproduced in the relay due to its back emf.

Circuit Diagram:



Interfacing of relay with 805

Program:

```
ORG 0000H
MAIN:
SETB P3.7
ACALL DELAY
CLR P3.7
ACALL DELAY
SJMP MAIN
DELAY:
MOV R0,#10
HERE: MOV R1,#200
```

```
HERE1: MOV R2,#200
HERE2: DJNZ R2,HERE2
DJNZ R1,HERE1
DJNZ R0,HERE
RET
END
```

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong powersuppliesmay cause damage to your equipment.

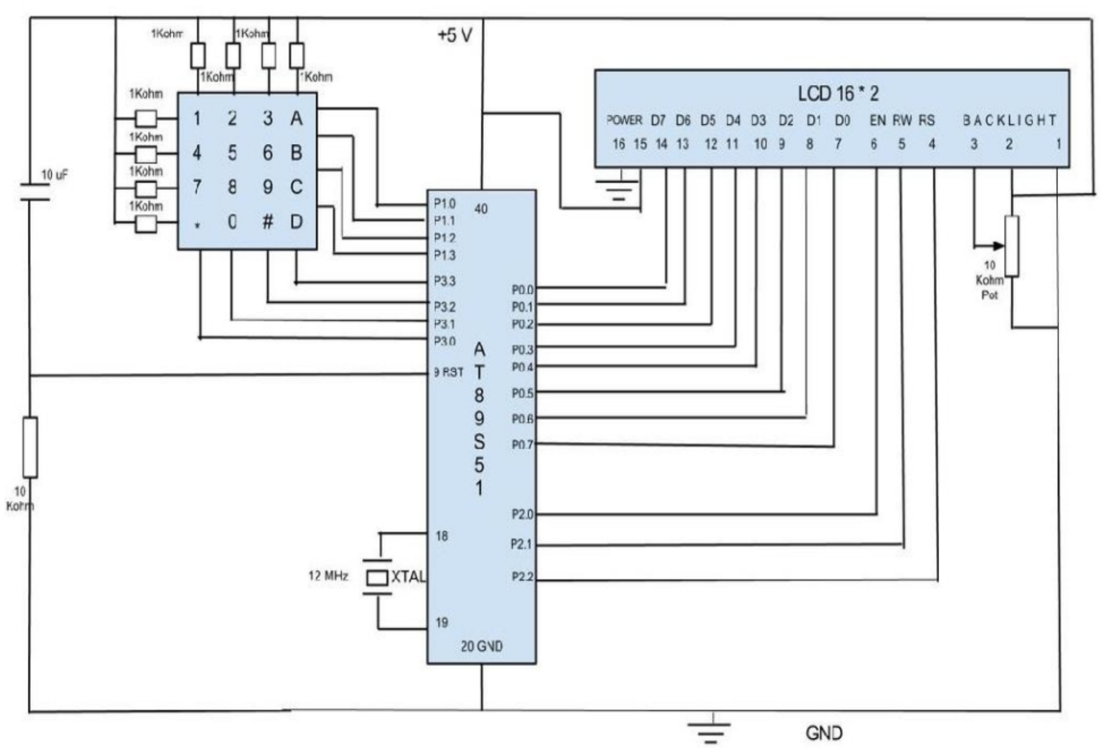
Experiment No: 8

Aim: To study implementation & interfacing of keypad with 8051 microcontroller

Apparatus Required: Microcontroller kit, Interfacing kit, Keyboard, Monitor, SMPS for Microcontroller.

Theory:
Matrix Keypads are commonly used in calculators, telephones etc. where several input switches are required. We know that matrix keypad is made by arranging push button switches in row and columns. In the straight forward way to connect a 4×4 keypad (16 switches) to a microcontroller we need 16 inputs pins. Keypad is a widely-used input device with lots of application in our everyday life. From a simple telephone to keyboard of a computer, ATM, electronic lock, etc., keypad is used to take input from the user for further processing.

Circuit Diagram:



Program:

```
ORG 0000H
MOV P2,#0FFH
LCD_INIT:
mov a,#38h
acall comwrt
acall delay
mov a,#0Ch
acall comwrt
acall delay
mov a,#01h
acall comwrt
acall delay
mov a,#06h
acall comwrt
acall delay
mov a,#80h
acall comwrt
acall delay
START:
MOV A,#'M'
ACALL DATAWRT
MOV A,#'A'
ACALL DATAWRT
MOV A,#'T'
```

```
ACALL DATAWRT
MOV A,#'R'
ACALL DATAWRT
MOV A,#'I'
ACALL DATAWRT
MOV A,#'X'
ACALL DATAWRT
MOV A,#' '
ACALL DATAWRT
MOV A,#'K'
ACALL DATAWRT
MOV A,#'E'
ACALL DATAWRT
MOV A,#'Y'
ACALL DATAWRT
MOV A,#'B'
ACALL DATAWRT
MOV A,#'O'
ACALL DATAWRT
MOV A,#'A'
ACALL DATAWRT
MOV A,#'R'
ACALL DATAWRT
MOV A,#'D'
ACALL DATAWRT
MOV A,#0C0H
ACALL COMWRT
MOV A,#'K'
ACALL DATAWRT
MOV A,#'E'
ACALL DATAWRT
MOV A,#'Y'
ACALL DATAWRT
MOV A,#' '
ACALL DATAWRT
MOV A,#'P'
ACALL DATAWRT
MOV A,#'R'
ACALL DATAWRT
MOV A,#'E'
ACALL DATAWRT
MOV A,#'S'
ACALL DATAWRT
MOV A,#'S'
ACALL DATAWRT
MOV A,#'E'
ACALL DATAWRT
MOV A,#'D'
ACALL DATAWRT
MOV A,#':'
ACALL DATAWRT
```

```
PREVIOUS_KEY_RELEASED:
MOV P3,#00H
MOV A,P2
ANL A,#00001111B
CJNE A,#00001111B,PREVIOUS_KEY_RELEASED
NEXT_KEY_SCAN:
ACALL DEBOUNCE_TIME
MOV A,P2
ANL A,#00001111B
CJNE A,#00001111B,KEY_SCAN_AGAIN
SJMP NEXT_KEY_SCAN
KEY_SCAN_AGAIN:
ACALL DEBOUNCE_TIME
MOV A,P2
ANL A,#00001111B
CJNE A,#00001111B,IDENTIFY_KEY_COL
SJMP NEXT_KEY_SCAN
IDENTIFY_KEY_COL:
MOV P3,#11111110B
MOV A,P2
ANL A,#00001111B
CJNE A,#00001111B,ROW_0
MOV P3,#11111101B
MOV A,P2
ANL A,#00001111B
CJNE A,#00001111B,ROW_1
MOV P3,#11111011B
MOV A,P2
ANL A,#00001111B
CJNE A,#00001111B,ROW_2
MOV P3,#11110111B
MOV A,P2
ANL A,#00001111B
CJNE A,#00001111B,ROW_3
ROW_0:
MOV DPTR,#ROW_0_ELEMENTS
SJMP FIND_KEY
ROW_1:
MOV DPTR,#ROW_1_ELEMENTS
SJMP FIND_KEY
ROW_2:
MOV DPTR,#ROW_2_ELEMENTS
SJMP FIND_KEY
ROW_3:
MOV DPTR,#ROW_3_ELEMENTS
SJMP FIND_KEY
FIND_KEY:
RRC A
JNC MATCH_KEY
INC DPTR
SJMP FIND_KEY
```

```
MATCH_KEY:
CLR A
MOVC A,@A+DPTR
MOV P0,A
LJMP PREVIOUS_KEY_RELEASED
comwrt:
mov p1,a
clr p3.4
clr P3.5
setb p3.6
acall delay
clr p3.6
ret
datawrt:
mov p1,a
setb p3.4
clr P3.5
setb p3.6
acall delay
clr p3.6
ret
delay:
MOV R1,#255
here:
DJNZ R1,here
ret
DEBOUNCE_TIME:
MOV TMOD,10H
START1:
MOV TL1 ,#0FFH
MOV TH1 ,#0B7H
SETB TR1
AGAIN:
JNB TF1,AGAIN
CLR TR1
CLR TF1
RET
ORG 0500H
ROW_0_ELEMENTS: DB '0','1','2','3'
ROW_1_ELEMENTS: DB '4','5','6','7'
ROW_2_ELEMENTS: DB '8','9','A','B'
ROW_3_ELEMENTS: DB 'C','D','E','F'
END
```

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong powersuppliesmay cause damage to your equipment.

Experiment No: 9

Aim: To Study of implementation of Motor control using PWM method.

Apparatus Required: Microcontroller kit, Interfacing kit, Keyboard, Monitor, SMPS for Microcontroller.

PROCEDURE:
1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch 'On' the power supply.
4. Program PWM interface.hex file (Via CD - NV5001/ \Modules programs\MC05 Drive module \DC motor interface module\PWM Interface program) in AT89C52 Microcontroller viaprogrammer.
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIFsocket
6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to DC motor /PWM interface block socket (MC05) to Port P2 inNV5001 Trainer.
9. Connect 2 mm patch cord between +12V DC block socket (in NV5001) to +12V DC socket inDC motor /PWM interface block (in MC05).
10. Switch 'On' the power supply.
11. Check the status of port pins on tp7 to tp11
12. Observe the status of PWM switch at tp11.
13. Observe the rotation speed of DC Motor.
14. Press PWM switch and repeat steps 11 to 13 one time and observe the speed change of DCmotor.

PROGRAM: To monitor the PWM status and control the speed of DC motor in 100% and 25%duty cycle pulse.

```
PWM_SW EQU P2.4
INPUT2 EQU P2.2
INPUT1 EQU P2.1
PWM_INPUT EQU P2.0
------------------------------------------------------------------------------------------------------------
-----
ORG 0000H
JMP START
------------------------------------------------------------------------------------------------------------
-----
ORG 0200H
START : MOV A, #00H
CLR C
SETB PWM_SW
SETB PWM_INPUT
SETB INPUT1
CLR INPUT2
```

```
SJMP START
-------------------------------------------------------------------------------------------------------------
------
FIR_ROU : SETB PWM_SW
CLR PWM_INPUT
LCALL DELAY_1S_2
LCALL DELAY_1S_2
LCALL DELAY_1S_2
LCALL DELAY_1S_2
SETB PWM_INPUT
-------------------------------------------------------------------------------------------------------------
--------
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
LCALL DELAY_1S
SJMP FIR_ROU
-------------------------------------------------------------------------------------------------------------
-----
DELAY_1S_2 : MOV R2, #50
DHERE1_1_1 : MOV R3, #100
DHERE1_2 : NOP
DJNZ R3, DHERE1_2
DJNZ R2, DHERE1_1_1
RET
-------------------------------------------------------------------------------------------------------------
-----
DELAY_1S : MOV R2,#100
DO3_3 : DEC R2
DJNZ R2, DO3_3
RET
-------------------------------------------------------------------------------------------------------------
-------
END
```

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong powersuppliesmay cause damage to your equipment.

UNIVERSITY OF ENGINEERING AND MANAGEMENT,JAIPUR
Lab Manual

Experiment No: 10

Aim: To Study and observation of Position control of Servo Motor.

Apparatus Required: Microcontroller kit, Interfacing kit, Keyboard, Monitor, SMPS for Microcontroller.

PROCEDURE:
1. Insert AT89C52 Microcontroller in Programmer unit (in NV5001).
2. Connect serial cable between computer serial port and programmer unit serial port female connector (in NV5001).
3. Switch 'On' the programmer switch in programmer unit (in NV5001) and switch 'On' the power supply.
4. Program servo motor module.hex file (Via CD - NV5001/ \Modules programs\MC05 Drive module \DC motor interface module\Servo motor module) in AT89C52 Microcontroller via programmer.
5. Switch 'Off' the power supply and remove the programmed controller from programmer ZIF socket
6. Switch 'Off' the programmer switch in Programmer unit (in NV5001).
7. Insert programmed Microcontroller to microcontroller unit ZIF socket.
8. Connect 20 Pin FRC cable to servo motor interface block socket (MC05) to Port P2 in NV5001 Trainer.
9. Switch 'On' the power supply.
10. Check the status of port pins on tp12 to tp13.
11. Observe servo motor rotates and stop in the centre position or in 90-degree angle.
12. Press position control switch and repeat steps 10.
13. Observe servo motor rotates and stop in l80 degree angle or in a left side position.

PROGRAM: To monitor the status of position control switch and control the angle of servo motor.

```
SERVO_PIN EQU P2.0
SW_PIN EQU P2.1
-----------------------------------------------------------------------------------------------------
-------
ORG 0000H
JMP START
-----------------------------------------------------------------------------------------------------
------
ORG 0200H
START : CLR SERVO_PIN
SETB SW_PIN
LOOP_S : LCALL DELAY
SETB SW_PIN
SETB SERVO_PIN
LCALL DELAY_15MS_P
CLR SERVO_PIN
LCALL DELAY_16MS
JNB SW_PIN, SW_1_1
LCALL DELAY
SJMP LOOP_S
```

```
-----
SW_1_1 : LCALL DELAY
SETB SW_PIN
SETB SERVO_PIN
LCALL DELAY_25MS_P
CLR SERVO_PIN
LCALL DELAY_16MS
LCALL DELAY
SJMP SW_1_1
------------------------------------------------------------------------------------------------------------
-----
DELAY_16MS : MOV R2, #150
DHERE1_16 : MOV R3, #32
DAGAIN_16 : NOP
DJNZ R3, DAGAIN_16
DJNZ R2, DHERE1_16
RET
------------------------------------------------------------------------------------------------------------
-----
DELAY_25MS_P : MOV R2, #20
DHERE1_25_P : MOV R3, #37
DAGAIN_25_P : NOP
DJNZ R3, DAGAIN_25_P
DJNZ R2, DHERE1_25_P
RET
------------------------------------------------------------------------------------------------------------
-----
DELAY_15MS_P : MOV R2, #20
DHERE1_15_P : MOV R3, #20
DAGAIN_15_P : NOP
DJNZ R3, DAGAIN_15_P
DJNZ R2, DHERE1_15_P
RET
DELAY : MOV R5, #250
DHERE1 : MOV R4, #220
DAGAIN : NOP
NOP
DJNZ R4, DAGAIN
DJNZ R5, DHERE1
RET
------------------------------------------------------------------------------------------------------------
------
END
```

PRECAUTIONS: Make sure correct power supply is given to the kit/Equipment. Wrong powersuppliesmay cause damage to your equipment.

UNIVERSITY OF ENGINEERING AND MANAGEMENT, JAIPUR

Course Description

Title of Course: Seminar
Course Code: MVLSI181
L-T-P scheme: 0-2-0 **Course Credit: 1**

The overall aim of the seminar series is to help develop an emerging field at the intersection of multi-disciplinary understandings of culture and education. It will build on the existing body of work on education and culture, but its aim is explore and develop new perspectives in this area.
The objectives of the six exploratory seminars are:

- **to explore new research from a range of academic disciplines which sheds light on the questions outlined above**
- **to showcase cutting edge research on education and culture from outstanding academic researchers from the UK and internationally**
- **to bring together seminar participants from different disciplines such as Sociology, Philosophy, Psychology, Human Geography, Media Studies as well as Education and Cultural Studies**
- **to encourage and financially support the participation of PhD students**
- **to actively involve practitioners and users from each venue**
- **to engage a core group of policy makers**
- **to use the seminars to develop links between academics and stakeholders in the arts, library, media, community and educational sectors**