CS 0413 – VLSI AND EMBEDDED SYSTEM DESIGN

LABARATORY MANUAL

SEMESTER VII

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

SRM UNIVERSITY
(Under Section 3 of the UGC Act, 1956)

S.R.M NAGAR, KATTANKULATHUR – 603203.

KANCHEEPURAM DISTRICT

YEAR: 2015-2016

Department of Electronics and Communication Engineering

CS0413

VLSI and Embedded System Design Lab

Laboratory Manual

Course Team

Mr. S. Nivash
Mr. B.Srinath
Mrs. P.Niraimathi
Mrs. J.Subhashini
Mrs. P.Radhika
Mrs. V.Sarada
Mrs. E.Chitra
Mrs. V.K.Daliya
Mrs .K.Suganthi
Mrs. A.Ruhanbevi

June 2015

REVISION: 05

LIST OF EXPERIMENTS

SI.No	Title of the Experiments
1	Design of Logic gates
2	Design of Binary Adders
3	Design of Multiplexers and De-multiplexers
4	Design of Encoders and Decoders
5	Flip Flops
6	Counters
7	Toggle a Port bit in 8051
8	Bitwise Operators Using 8051
9	Arithmetic Operations using 8051
10	Delay Operators in 8051

Course Handout

SRM University
Faculty of Engineering and Technology
Department of Electronics and Communication Engineering

CS0413 VLSI & EMBEDDED SYSTEM DESIGN LAB

Seventh Semester, 2015-16 (odd semester)

Course (Catalog) description

The course explores the design aspects of an introduction to the characteristics of digital logic, design, construction, testing and debugging of simple digital circuits using Verilog HDL. and also provide an introduction to the development of application using microcontrollers.

Compulsory/Elective course: Compulsory for CSE students.

Credit hours: 2 credits.

Laboratory

DSP Lab- TP9L3, Embedded System Lab - TP 11L1,
VLSI Simulation Lab- TP11L4, VLSI Design Lab- TP12L4

Course coordinator(s):

Mr.S.Nivash, Assistant. Professor, Department of ECE

Instructor(s)

Class / Lab schedule: one 150 minutes lab session per week, for 14-15 weeks

Name of the instructor	Class	Venue	Class hours	Email (domain: @ktr.srmuniv.ac.in)
Mr.S.Nivash	X1	TP11L1	Day 1-7,8 Day 2-3,4	nivash.s@ktr.srmuniv.ac.in
Mr.B.Srinath	X2	TP11L1	Day 1 - 5,6 Day 3 - 7,8	srinath.b@ktr.srmuniv.ac.in
Mrs.P.Niraimathi	X3	TP11L1	Day 4 -7,8 Day 5 -7,8	niraimathi.p@ktr.srmuniv.ac.in

Name of the instructor	Class	Venue	Class hours	Email (domain: @ktr.srmuniv.ac.in)
Mrs.J.Subashini	X4	TP11L1	Day 2 - 5,6 Day 5 - 3,4	subashini.j@ktr.srmuniv.ac.in
Mrs.P.Radhika	X5	TP11L4	Day 1 - 5,6 Day 2 - 5,6	radhika.p@ktr.srmuniv.ac.in
Mrs.V.Sarada	Y1	TP11L1	Day 1 - 3,4 Day 2 - 7,8	sarada.v@ktr.srmuniv.ac.in
Mrs.E.Chitra	Y2	TP11L4	Day 2 - 7,8 Day 3 - 3,4	chitra.e@ktr.srmuniv.ac.in
Mrs.V.K.Daliya	Y3	TP9L3	Day 1 - 5,6 Day 2 - 5,6	daliya.vk@ktr.srmuniv.ac.in
Mrs.K.Suganthi	Y4	TP11L4	Day 1 - 3,4 Day 4 – 3,4	suganthi.k@ktr.srmuniv.ac.in
Dr.A.Ruhan Bevi	Y5	TP10L1	Day 1- 5,6 Day 2 - 5,6	ruhanbevi.a@ktr.srmuniv.ac.in

Relationship to other courses

Pre-requisites: Digital Computer Fundamentals

Assumed knowledge: Digital Computer Fundamentals, Verilog and Programming in Keil C

Following courses: Nil

Required Text Books:

1. Samir Palnitkar, "Verilog HDL: A guide to Digital Design and Synthesis", 2nd edition, Pearson Education.
2. Jan M. Rabaey, Anantha Chandrakasan and Borivoje Nikolic, "Digital Integrated Circuits", Second Edition, Prentice-Hall.
3. Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. McKinlay, "The 8051 Microcontroller and Embedded systems", Person Education, 2004.
4. Rajkamal, "Embedded Systems: Architecture, Programming and Design", Tata McGraw-Hill Education, 2008.
5. Lab manual; additional materials posted on SRM web.

Web Resources:

1. www.asic-world.com
2. www.keil.com/c51/

Computer usage

Modelsim is used to simulate the design and to verify the functionality.

1. Modelsim 5.7
2. Xilinx ISE 7.1
3. Keil C

Professional component

General	-	0%
Basic Sciences	-	0%
Engineering sciences & Technical arts	-	0%
Professional subject	-	100%

Broad area: VLSI | Embedded

Hardware Laboratory Usage

Each laboratory station is equipped with computer with the simulation software loaded. Students work individually and maintain individual laboratory notebooks and submit individual reports.

Course objectives

The objectives of this course is to	Correlates to Program Objective
To gain expertise in design and development and simulation of digital circuits with Verilog.	(3), (4)

Course Learning Outcome

This course provides the design of various digital circuits using different VLSI simulation software tools like Modelsim, Keil C. The outcome of this course to learn Verilog language, Keil C and also learn the usage of different tools.	Correlates to program outcome		
	H	**M**	**L**
1. To design and simulate list of combinational and sequential digital circuits using Modelsim & Xilinx – Verilog language	d,f	c	b,k
2. To design and simulate the operations of systems like verilog using Modelsim & Toggle, Bitwise, Delay and any Control Logic Design in 8051.	d,f		
3. To design Toggle, Bitwise, Arithmetic, Delay using Keil C.	f		

H: high correlation, M: medium correlation, L: low correlation

Course Topics

SI.No.	Lab Experiments
*I.Design and simulate using Modelsim/Xilinx-**Verilog Language***	
1	Design of Logic gates
2	Design of Binary Adders
3	Design of Multiplexers and De-multiplexers
4	Design of Encoders and Decoders
5	Flip Flops
6	Counters
II. Design and simulate using Keil µversion - 8051	
7	Toggle a Port bit in 8051
8	Bitwise Operators Using 8051
9	Arithmetic Operations using 8051
10	Delay Operators in 8051

Evaluation methods

Attendance	-	5%
Pre-lab questions	-	10%
In-lab experiment	-	15%
Post-lab questions	-	10%
Report	-	15%
Model exam	-	20%
Final exam	-	25%

Laboratory Policies and Report Format

Reports are due at the beginning of the lab period. The reports are intended to be a complete documentation of the work done in preparation for and during the lab. The report should be complete so that someone else familiar with digital communication could use it to verify your work. The prelab and postlab report format is as follows:

1. A neat thorough prelab must be presented to your Staff Incharge at the beginning of your scheduled lab period. **Lab reports should be submitted on A4 paper.** Your report is a professional presentation of your work in the lab. Neatness, organization, and completeness will be rewarded. Points will be deducted for any part that is not clear.

2. In this laboratory students will work in teams of three. However, the lab reports will be written individually. Please use the following format for your lab reports.

a. **Cover Page:** Include your name, Subject Code, Section No., Experiment No. and Date.

b.**Objectives:** Enumerate 3 or 4 of the topics that you think the lab will teach you. DO NOT REPEAT the wording in the lab manual procedures. There should be one or two sentences per objective. Remember, you should write about what you will learn, not what you will do.

c. **Design and simulation**: This part contains all the steps required to arrive at your final stage. This should include all input and output waveforms, explanations, etc.

d.This section should also include a clear and error free program description of your design process. Simply including a circuit schematic is not sufficient.

e. **Questions**: Specific questions(Prelab and Postlab) asked in the lab should be answered here. Retype the questions presented in the lab and then formally answer them.

3. Your work must be original and prepared independently. However, if you need any guidance or have any questions or problems, please do not hesitate to approach your staff incharge during office hours. Copying any prelab/postlab will result in a grade of 0. The incident will be formally reported to the University and the students should follow the dress code in the Lab session.

4. Each laboratory exercise (simulation results) must be completed and demonstrated to your Staff In-charge in order to receive working design model credit. This is the procedure to follow:

a. **Design model works**: If the design model works during the lab (2 periods), call your staff in-charge and he/she will sign and date it. This is the end of this lab, and you will get a complete grade for this portion of the lab.

b. **Design model does not work**: If the Design model does not work, you must make use of the open times for the lab room to complete your experiment. When your design model is ready, contact your staff in-charge to set up a time when the two of you can meet to check your simulation.

5. Attendance at your regularly scheduled lab period is required. An unexpected absence will result in loss of credit for your lab. If for valid reason a student misses a lab, or makes a reasonable request in advance of the class meeting, it is permissible for the student to do the lab in

a different section later in the week if approved by the staff in-charge of both the sections. Habitually late students (i.e., students late more than 15 minutes more than once) will receive 10 point reductions in their grades for each occurrence following the first.

6.Final grade in this course will be based on laboratory assignments. All labs have an equal weight in the final grade. Grading will be based on pre-lab work, laboratory reports, post-lab and in-lab performance (i.e., completing lab, answering laboratory related questions, etc.,).The Staff In-charge will ask pertinent questions to individual members of a team at random. Labs will be graded as per the following grading policy:

Attendance	-	5%
Pre-lab questions	-	10%
In-lab experiment	-	15%
Post-lab questions	-	10%
Report	-	15%
Model exam	-	20%
Final exam	-	25%

7. **Reports Due Dates**: Reports are due one week after completion of the corresponding lab. A late lab report will have 10% of the points deducted for being one day late.

8. **Systems of Tests:** Regular laboratory class work over the full semester will carry a weightage of 75%. The remaining 25% weightage will be given by conducting an end semester practical examination for every individual student if possible or by conducting a 1 to 1 ½ hours duration common written test for all students, based on all the experiment carried out in the semester.

.....

Prepared by: S.Nivash, Assistant Professor, Department of ECE

Dated: 23.06 .2015 **Revision No.:** 05 **Date of revision:** NA

.....

Program Educational Objectives

(i) To prepare students to compete for a successful career in Electronics and Communication Engineering profession through global education standards.

(ii) To enable the students to aptly apply their acquired knowledge in basic sciences and mathematics in solving Electronics and Communication Engineering problems.

(iii) To produce skillful graduates to analyze, design and develop a system/component/ process for the required needs under the realistic constraints.

(iv) To train the students to approach ethically any multidisciplinary engineering challenges with economic, environmental and social contexts

(v) To create awareness among the students about the need for life long learning to succeed in their professional career as Electronics and Communication Engineers.

Program Outcomes

a. an ability to apply knowledge of mathematics, science, and engineering
b. an ability to design and conduct experiments, as well as to analyze and interpret data
c. an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability
d. an ability to function on multidisciplinary teams
e. an ability to identify, formulate, and solve engineering problems
f. an understanding of professional and ethical responsibility
g. an ability to communicate effectively
h. the broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context
i. a recognition of the need for, and an ability to engage in life-long learning
j. a knowledge of contemporary issues
k. an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.

Name of the Staff	Group	Signature
Mr.S.Nivash	I	
Mr.B.Srinath		
Mrs.P.Niraimathi		
Mrs.J.Subashini		
Mrs.P.Radhika		
Mrs.V.Sarada	II	
Mrs.E.Chitra		
Mrs.V.K.Daliya		
Mrs.K.Suganthi		
Dr.A.Ruhan Bevi		

Course Co-ordinator

(Mr.S.Nivash)

Professor In-charge

(Mr.V.Natrajan)

Syllabus of VLSI & Embedded System Design Lab

		L	T	P	C
CS0413	**VLSI & Embedded System Design Lab**	**0**	**0**	**3**	**2**
	Total Contact Hours – 45				
	Prerequisite: Nil				
PURPOSE					
The purpose of the lab is to train the students to design to know and understand Verilog and design circuits using it.					
INSTRUCTIONAL OBJECTIVES					
1.	To gain expertise in design and development and simulation of digital circuits with Verilog.				

LIST OF EXPERIMENTS

1. Design of Logic gates
2. Design of Binary Adders
3. Design of Multiplexers and De-multiplexers
4. Design of Encoders and Decoders
5. Flip Flops
6. Counters
7. Toggle a Port bit in 8051
8. Bitwise Operators Using 8051
9. Arithmetic Operations using 8051
10. Delay Operators in 8051

REFERENCES

6. "LAB MANUAL", Department of ECE, SRM University
7. Samir Palnitkar, "Verilog HDL: A guide to Digital Design and Synthesis", 2nd edition, Pearson Education.
8. Jan M. Rabaey, Anantha Chandrakasan and Borivoje Nikolic, "Digital Integrated Circuits", Second Edition, Prentice-Hall.
9. Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. McKinlay, "The 8051 Microcontroller and Embedded systems", Person Education, 2004.
10. Rajkamal, "Embedded Systems: Architecture, Programming and Design", Tata McGraw-Hill Education, 2008.

<table>
<tr><th colspan="13">VLSI & Embedded System Design Lab</th></tr>
<tr><td colspan="2">Course designed by</td><td colspan="11">Department of Electronics and Communication Engineering</td></tr>
<tr><td rowspan="2">1</td><td rowspan="2">Student outcome</td><td>A</td><td>b</td><td>c</td><td>d</td><td>e</td><td>f</td><td>g</td><td>h</td><td>i</td><td>j</td><td>k</td></tr>
<tr><td></td><td>X</td><td>X</td><td>X</td><td></td><td>X</td><td></td><td></td><td></td><td></td><td>X</td></tr>
<tr><td>2</td><td>Mapping of Instructional Objectives with student outcome</td><td></td><td>1</td><td>1</td><td>1</td><td></td><td>1</td><td></td><td></td><td></td><td></td><td>1</td></tr>
<tr><td rowspan="2">3</td><td rowspan="2">Category</td><td colspan="2">General (G)</td><td colspan="2">Basic Sciences (B)</td><td colspan="4">Engineering Sciences & Technical Arts (E)</td><td colspan="3">Professional Subjects (P)</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td colspan="4"></td><td colspan="3">X</td></tr>
<tr><td rowspan="2">4</td><td rowspan="2">Broad area</td><td colspan="2">Communication</td><td colspan="2">Signal Processing</td><td colspan="3">Electronics</td><td>VLSI</td><td colspan="3">Embedded</td></tr>
<tr><td colspan="2"></td><td colspan="2"></td><td colspan="3">X</td><td>X</td><td colspan="3">X</td></tr>
<tr><td>5</td><td>Approval</td><td colspan="11"></td></tr>
</table>

S.R.M University

Faculty of Engineering and Technology

Department of Electronics and Communication Engineering

Sub Code : CS 0413 **Semester : VII**

Sub Title : VLSI & EMBEDDED SYSTEM DESIGN LAB **Course Time : Jun - Nov'15**

Pre Requisite : CS0102 Digital Computer Fundamentals

Course Requisite : CS0405 VLSI Design & Embedded System

Program Outcome

b. Graduate will demonstrate the ability to identify, formulate and solve engineering problems.

c. Graduate will demonstrate the ability to design and conduct experiments, analyze and interpret data.

Experiments in VLSI Devices and Design will satisfy the program outcome b and c.

d. Graduate will demonstrate the ability to design a system, component or process as per needs and specification

Experiment 7, 8 and 9: To understand the operations of systems like Toggle, Bitwise, Delay and any Control Logic Design in 8051.

f. Graduate will demonstrate the skills to use modern engineering tools, software's and equipment to analyze problems.

Experiments in VLSI Devices and Design will satisfy the program outcome f.

k. Graduate will show the ability to participate and try to succeed in competitive examinations

To participate in placement exams of most of the software companies.

S.R.M University

Faculty of Engineering and Technology

Department of Electronics and Communication Engineering

Sub Code : CS 0413 **Semester : VII**

Sub Title : VLSI & EMBEDDED SYSTEM DESIGN LAB **Course Time : Jun - Nov'15**

Pre Requisite : CS0102 Digital Computer Fundamentals

Course Requisite : CS0405 VLSI Design & Embedded System

Instructional Objective and Program Outcome

S.No.	Instructional Objective	Program Outcome	Experiment Details
	To gain expertise in design and development and simulation of digital circuits with Verilog.	b. Graduate will demonstrate the ability to identify, formulate and solve engineering problems. **c.** Graduate will demonstrate the ability to design and conduct experiments, analyze and interpret data. **d**. Graduate will demonstrate the ability to design a system, component or process as per needs and specification **f.** Graduate will demonstrate the skills to use modern engineering tools, software's and equipment to analyze problems. **k.** Graduate will show the ability to participate and try to succeed in competitive examinations	Experiments in VLSI Devices and Design will satisfy the program outcome b and c. xperiment 7, 8 and 9: To understand the operations of systems like Toggle, Bitwise, Delay and any Control Logic Design in 8051. Experiments in VLSI Devices and Design will satisfy the program outcome f. To participate in placement exams of most of the software companies.

S.R.M University

Faculty of Engineering and Technology

Department of Electronics and Communication Engineering

Sub Code : CS 0413 **Semester : VII**

Sub Title : VLSI & EMBEDDED SYSTEM DESIGN LAB **Course Time : Jun - Nov'15**

Pre Requisite : CS0102 Digital Computer Fundamentals

Course Requisite : CS0405 VLSI Design & Embedded System

EXPERIMENTS DETAILS

SI.No.	Lab Experiments
*I.Design and simulate using Modelsim/Xilinx-**Verilog Language***	
1	Design of Logic gates
2	Design of Binary Adders
3	Design of Multiplexers and De-multiplexers
4	Design of Encoders and Decoders
5	Flip Flops
6	Counters
II. Design and simulate using Keil µversion - 8051	
7	Toggle a Port bit in 8051
8	Bitwise Operators Using 8051
9	Arithmetic Operations using 8051
10	Delay Operators in 8051

Specifications:

- HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk
- Softwares: Modelsim - 5.7c, Xilinx - 6.1i. QestaSim, Keil µversion – 8051.

Laboratory Report Cover Sheet

SRM University
Faculty of Engineering and Technology
Department of Electronics and Communication Engineering

CS 0413

VLSI Design & Embedded System Lab

Seventh Semester, 2015 (Odd semester)

Name :

Register No. :

Venue : **DSP Lab- TP9L3, Embedded Systems Lab-TP11L1, VLSI Simulation Lab- TP11L4, VLSI Design Lab- TP12L4**

Title of Experiment :

Date of Conduction :

Date of Submission :

Particulars	Max. Marks	Marks Obtained
Pre-lab	10	
Post-lab	10	
In Lab Performance	15	
Lab Report	15	
Total	**50**	

REPORT VERIFICATION

Date :

Staff Name :

Signature :

CS0413 VLSI AND EMBEDDED SYSTEM DESIGN LAB

Contents

List of Experiments:

SI. No	Experiments	Page. No
1	Design of Logic gates	1
2	Design of Binary Adders	8
3	Design of Multiplexers and De-multiplexers	15
4	Design of Encoders and Decoders	26
5	Flip Flops	33
6	Counters	42
7	Toggle a Port bit in 8051	46
8	Bitwise Operators Using 8051	50
9	Arithmetic Operations using 8051	53
10	Delay Operators in 8051	56

Introduction to Combinational Circuit Design

EXP:1 Design of Logic gates

1.1 Introduction

The purpose of this experiment is to simulate the behavior of several of the basic logic gates and you will connect several logic gates together to create simple digital model.

1.2 Software tools Requirement

Equipments:

Computer with Modelsim Software

Specifications:

HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Modelsim - 5.7c, Xilinx - 6.1i.

Algorithm

STEP 1: Open ModelSim XE II / Starter 5.7C

STEP 2: File -> Change directory -> D:\<register number>

STEP 3: File -> New Library -> ok

STEP 4: File -> New Source -> Verilog

STEP 5: Type the program

STEP 6: File -> Save -> <filename.v>

STEP 7: Compile the program

STEP 8: Simulate -> expand work -> select file -> ok

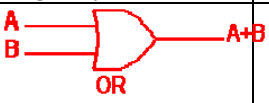
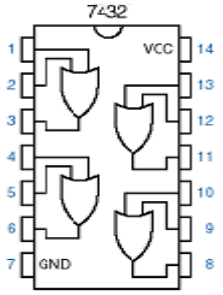
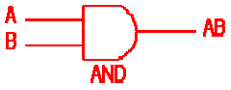
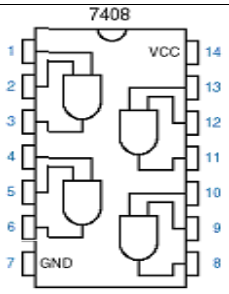
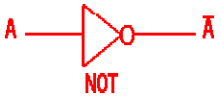
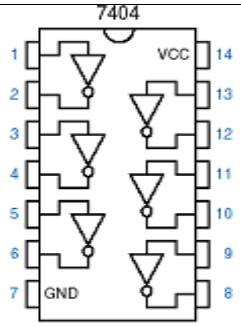
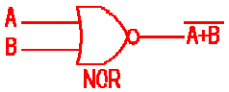
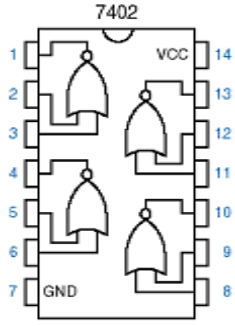
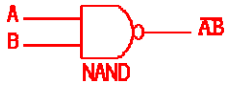
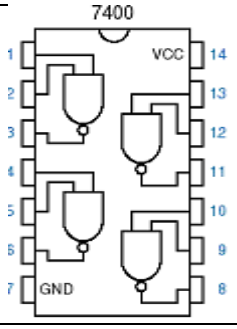
STEP 9: View -> Signals

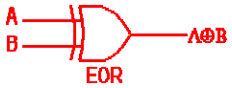
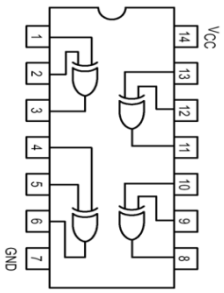
STEP 10: Select values -> Edit -> Force -> input values

STEP 11: Add -> Wave -> Selected signals -> Run

STEP 12: Change input values and run again

1.3 Logic Gates and their Properties

Gate	Description	Truth Table			Logic Symbol	Pin Diagram
OR	The output is active high if any one of the input is in active high state, Mathematically, Q = A+B	A 0 0 1 1	B 0 1 0 1	Output Q 0 1 1 1		
AND	The output is active high only if both the inputs are in active high state, Mathematically, Q = A.B	A 0 0 1 1	B 0 1 0 1	Output Q 0 0 0 1		
NOT	In this gate the output is opposite to the input state, Mathematically, Q = A	A 0 1		Output Q 1 0		
NOR	The output is active high only if both the inputs are in active low state, Mathematically, Q = (A+B)’	A 0 0 1 1	B 0 1 0 1	Output Q 1 0 0 0		
NAND	The output is active high only if any one of the input is in active low state, Mathematically, Q = (A.B)’	A 0 0 1 1	B 0 1 0 1	Output Q 1 1 1 0		

XOR	The output is active high only if any one of the input is in active high state, Mathematically, Q = A.B' + B.A'	A 0 0 1 1	B 0 1 0 1	Output Q 0 1 1 0		7486 

1.4 Pre lab Questions

1. What is truth table?

2. Which gates are called universal gates?

3. A basic 2-input logic circuit has a HIGH on one input and a LOW on the other input, and the output is HIGH. What type of logic circuit is it?

4. A logic circuit requires HIGH on all its inputs to make the output HIGH. What type of logic circuit is it?

5. Develop the truth table for a 3-input AND gate and also determine the total number of possible combinations for a 4-input AND gate.

VERILOG Program

a) AND Gate

Structural Model	**Data Flow Model**	**BehaviouralModel**

moduleandstr(x,y,z);
 	
inputx,y;
 	
output z;
 	
and g1(z,x,y);
 	
endmodule
	moduleanddf(x,y,z);
	
	inputx,y;
	
	output z;
	
	assign z=(x&y);
	
	endmodule
		module andbeh(x,y,z);
		
		input x,y;
		
		output z;
		
		reg z;
		
		always @(x,y)
		
		z=x&y;
		
		endmodule

b) NAND Gate

Structural Model	Data Flow Model	BehaviouralModel

modulenandstr(x,y,z);
 	
inputx,y;
 	
output z;
 	
nand g1(z,x,y);
 	
endmodule
	modulenanddf(x,y,z);
	
	inputx,y;
	
	output z;
	
	assign z= !(x&y);
	
	endmodule
		module nandbeh(x,y,z);
		
		input x,y;
		
		output z;
		
		reg z;
		
		always @(x,y)
		
		z=!(x&y);
		
		endmodule

c) OR Gate

Structural Model	Data Flow Model	BehaviouralModel

module orstr(x,y,z);
 	
inputx,y;
 	
output z;
 	
or g1(z,x,y);
 	
endmodule
	module ordf(x,y,z);
	
	inputx,y;
	
	output z;
	
	assign z=(x|y);
	
	endmodule
		module orbeh(x,y,z);
		
		input x,y;
		
		output z;
		
		reg z;
		
		always @(x,y)
		
		z=x|y;
		
		endmodule

d) NOR Gate

Structural Model	Data Flow Model	BehaviouralModel

modulenorstr(x,y,z);
 	
inputx,y;
 	
output z;
 	
nor g1(z,x,y);
 	
endmodule
	modulenordf(x,y,z);
	
	inputx,y;
	
	output z;
	
	assign z= !(x|y);
	
	endmodule
		Modulenorbeh(x,y,z);
		
		input x,y;
		
		output z;
		
		reg z;
		
		always @(x,y)
		
		z=!(x|y);
		
		endmodule

e) XOR Gate

Structural Model	Data Flow Model	BehaviouralModel

module xorstr(x,y,z);
 	
inputx,y;
 	
output z;
 	
xor g1(z,x,y);
 	
endmodule
	module xordf(x,y,z);
	
	inputx,y;
	
	output z;
	
	assign z=(x^y);
	
	endmodule
		module xorbeh(x,y,z);
		
		input x,y;
		
		output z;
		
		reg z;
		
		always @(x,y)
		
		z=x^y;
		
		endmodule

f) XNOR Gate

Structural Model	Data Flow Model	BehaviouralModel

modulexnorstr(x,y,z);
 	
inputx,y;
 	
output z;
 	
xnor g1(z,x,y);
 	
endmodule
	modulexnordf(x,y,z);
	
	inputx,y;
	
	output z;
	
	assign z= !(x^y);
	
	endmodule
		module xnorbeh(x,y,z);
		
		input x,y;
		
		output z;
		
		reg z;
		
		always @(x,y)
		
		z=!(x^y);
		
		endmodule

g) NOT Gate

Structural Model	Data Flow Model	BehaviouralModel

module notstr(x,z);
 	
input x;
 	
output z;
 	
not g1(z,x);
 	
endmodule
	module notdf(x,z);
	
	input x;
	
	output z;
	
	assign z= !x;
	
	endmodule
		module notbeh(x,z);
		
		input x;
		
		output z;
		
		reg z;
		
		always @(x)
		
		z=!x;
		
		endmodule

Out put waveforms

AND Gate:

OR Gate:

NOT Gate:

NOR Gate:

NAND Gate:

XOR Gate:

1.5 Post lab Questions

1. What is meant by ports?

2. Write the different types of port modes.

3. What are different types of operators?

4. What is difference b/w <= and := operators?

5. What is meant by simulation?

1.6 Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab

Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

Lab cover sheet with staff verification sign.

Answer the pre-lab questions

Complete VERILOG code design for all logic gates and output signal waveforms

Answer the post-lab questions

1.7 Grading

Pre-lab Work	20 points
Lab Performance	30 points
Post-lab Work	20 points
Lab report	30 points

For the lab performance - at a minimum, demonstrate the operation of all the logic gates to your staff in-charge:

The lab report will be graded as follows (for the 30 points):

VERILOG code for each logic gates	15 points
Output signal waveform for all logic gates and its truth table	15 points

EXP:2 Design of Binary Adders

2.1 Introduction

The purpose of this experiment is to introduce the design of simple combinational circuits, in this case half adders, half subtractors, full adders and full subtractors.

2.2 Software tools Requirement

Equipments:

Computer with Modelsim Software

Specifications:

HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Modelsim - 5.7c, Xilinx - 6.1i.

Algorithm

STEP 1: Open ModelSim XE II / Starter 5.7C

STEP 2: File -> Change directory -> D:\<register number>

STEP 3: File -> New Library -> ok

STEP 4: File -> New Source -> Verilog

STEP 5: Type the program

STEP 6: File -> Save -> <filename.v>

STEP 7: Compile the program

STEP 8: Simulate -> expand work -> select file -> ok

STEP 9: View -> Signals

STEP 10: Select values -> Edit -> Force -> input values

STEP 11: Add -> Wave -> Selected signals -> Run

STEP 12: Change input values and run again

2.3 Logic Diagram

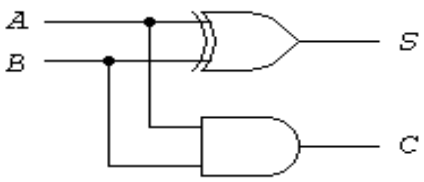


Figure 2.3.1Half adder

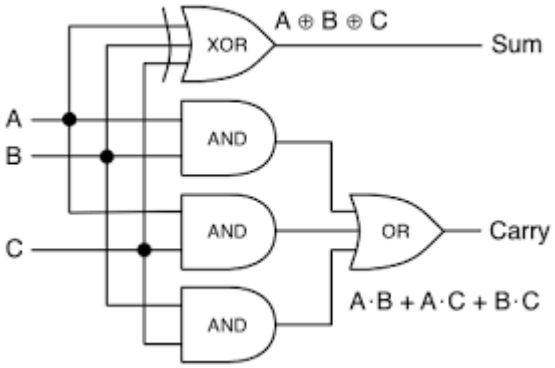


Figure 2.3.2 Full adder

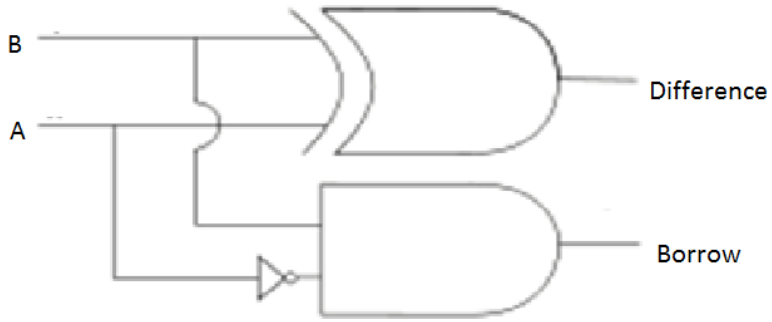


Figure 2.3.3Halfsubtractor

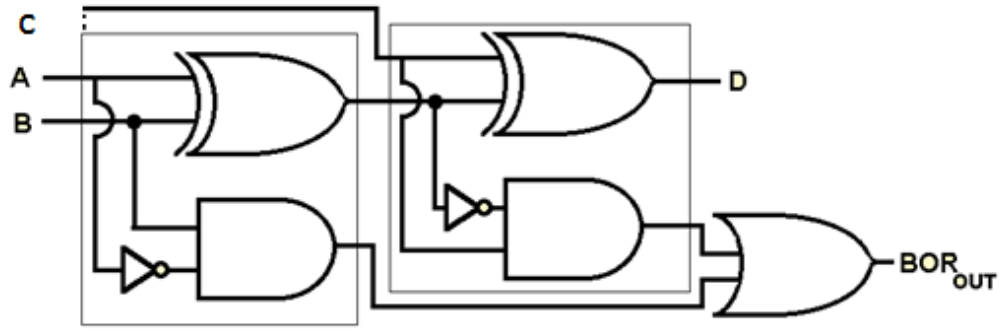


Figure 2.3.4 Full subtractor

2.4 Pre lab Questions

1. What is meant by combinational circuits?

2. Write the sum and carry expression for half and full adder.

3. Write the difference and borrow expression for half and full subtractor.

4. What is signal? How it is declared?

5. Design a one bit adder.

VERILOG Program

HALF ADDER:

Structural model	**Dataflow model**	**Behaviouralmodel**

modulehalfaddstr(sum,carry,a,b);
outputsum,carry;
inputa,b;
xor(sum,a,b);
and(carry,a,b);
endmodule
	modulehalfadddf(sum,carry,a,b);
	outputsum,carry;
	inputa,b;
	assign sum = a ^ b;
	assign carry=a&b;
	endmodule
		modulehalfaddbeh(sum,carry,a,b);
		outputsum,carry;
		inputa,b;
		regsum,carry;
		always @(a,b);
		sum = a ^ b;
		carry=a&b;
		endmodule

FULL ADDER:

Structural model	Dataflow model	Behaviouralmodel

module fulladdstr(sum,carry,a,b,c);
 	
outputsum,carry;
 	
inputa,b,c;
 	
xor g1(sum,a,b,c);
 	
and g2(x,a,b);
 	
and g3(y,b,c);
 	
and g4(z,c,a);
 	
or g5(carry,x,z,y);
 	
endmodule
	modulefulladddf(sum,carry,a,b,c);
	
	outputsum,carry;
	
	inputa,b,c;
	
	assign sum = a ^ b^c;
	
	assign carry=(a&b) | (b&c) | (c&a);
	
	endmodule
		modulefulladdbeh(sum,carry,a,b,c);
		
		outputsum,carry;
		
		inputa,b,c;
		
		regsum,carry;
		
		always @ (a,b,c)
		
		sum = a ^ b^c;
		
		carry=(a&b) | (b&c) | (c&a);
		
		endmodule

HALF SUBTRACTOR:

Structural model	Dataflow Model	BehaviouralModel

modulehalfsubtstr(diff,borrow,a,b);
 	
outputdiff,borrow;
 	
inputa,b;
 	
xor(diff,a,b);
 	
and(borrow,~a,b);
 	
endmodule
	modulehalfsubtdf(diff,borrow,a,b);
	
	outputdiff,borrow;
	
	inputa,b;
	
	assign diff = a ^ b;
	
	assign borrow=(~a&b);
	
	endmodule
		modulehalfsubtbeh(diff,borrow,a,b);
		
		outputdiff,borrow;
		
		inputa,b;
		
		regdiff,borrow;
		
		always @(a,b)
		
		diff = a ^ b;
		
		borrow=(~a&b);
		
		endmodule

FULL SUBTRACTOR:

Structural model	Dataflow Model	BehaviouralModel

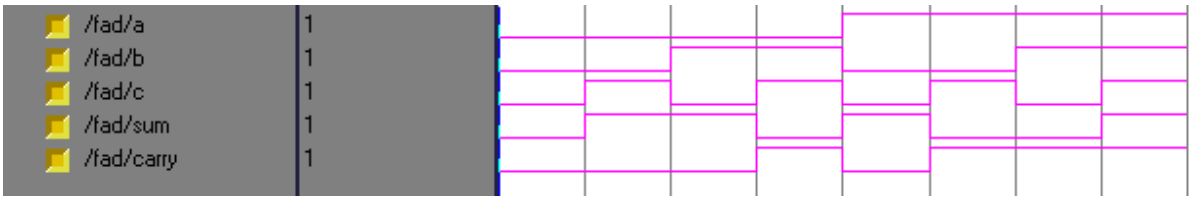
module fullsubtstr(diff,borrow,a,b,c);
 	
outputdiff,borrow;
 	
inputa,b,c;
 	
wire a0,q,r,s,t;
 	
not(a0,a);
 	
xor(x,a,b);
 	
xor(diff,x,c);
 	
and(y,a0,b);
 	
and(z,~x,c);
 	
or(borrow,y,z);
 	
endmodule
	modulefullsubtdf(diff,borrow,a,b,c);
	
	outputdiff,borrow;
	
	inputa,b,c;
	
	assign diff = a^b^c;
	
	assign borrow=(~a&b)|(~(a^b)&c);
	
	endmodule
		modulefullsubtbeh(diff,borrow,a,b,c);
		
		outputdiff,borrow;
		
		inputa,b,c;
		
		outputdiff,borrow;
		
		always@(a,b,)
		
		diff = a^b^c;
		
		borrow=(~a&b)|(~(a^b)&c);
		
		endmodule

Output waveforms:

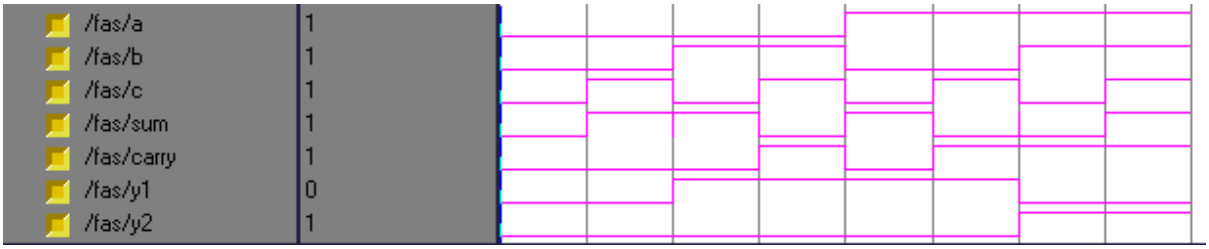
Half Adder:

Half subtractor:

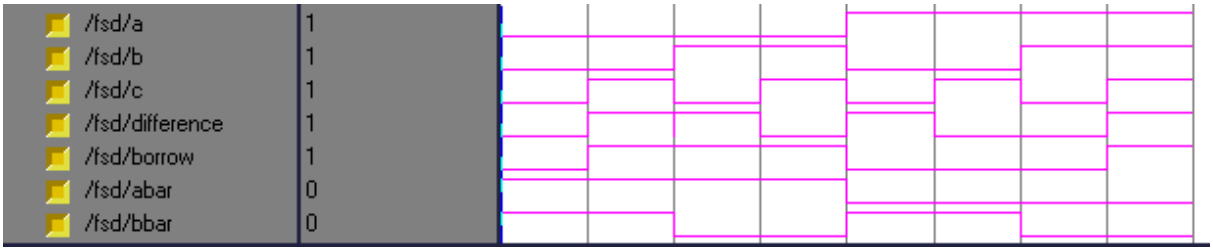
Full adder Dataflow modeling:



Full adder structural modeling:



Full Subtractor Dataflow modeling:



2.5 Post lab Questions

1. What are the signal assignment statements?

2. What are the concurrent statements?

3. Write short notes on: a) Process statement b) Block statement

4. Write about sequential statements.

5. What is the difference b/w high impedance state of the signal (Z) and unknown state of the signal(X).

2.6 Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab

Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

Lab cover sheet with staff verification sign.

Answer the pre-lab questions

Complete VERILOG code design for all logic gates and output signal waveforms

Answer the post-lab questions

2.7 Grading

Pre-lab Work	20 points
Lab Performance	30 points
Post-lab Work	20 points
Lab report	30 points

For the lab performance - at a minimum, demonstrate the operation of all the logic gates to your staff in-charge

The lab report will be graded as follows (for the 30 points):

VERILOG code for each experiments	15 points
Output signal waveform for all experiments and its truth table	15 points

EXP:3 Design of Multiplexers and Demultiplexers

3.1 Introduction

The purpose of this experiment is to write and simulate a VERILOG program for Multiplexers and Demultiplexers.

3.2 Software tools Requirement:

Equipments:

Computer with Modelsim Software

Specifications:

HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Modelsim - 5.7c, Xilinx - 6.1i.

Algorithm

STEP 1: Open ModelSim XE II / Starter 5.7C

STEP 2: File -> Change directory -> D:\<register number>

STEP 3: File -> New Library -> ok

STEP 4: File -> New Source -> Verilog

STEP 5: Type the program

STEP 6: File -> Save -> <filename.v>

STEP 7: Compile the program

STEP 8: Simulate -> expand work -> select file -> ok

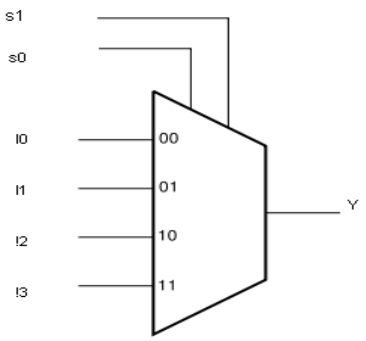
STEP 9: View -> Signals

STEP 10: Select values -> Edit -> Force -> input values

STEP 11: Add -> Wave -> Selected signals -> Run

STEP 12: Change input values and run again

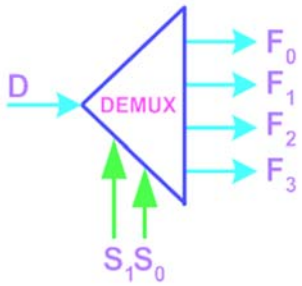
3.3 Logic Diagram



s1	s0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Function Table

Figure 3.3.1 4:1 Multiplexer Block diagram



S1	S0	F0	F1	F2	F3
0	0	D	0	0	0
0	1	0	D	0	0
1	0	0	0	D	0
1	1	0	0	0	D

Figure 3.3.2 1:4 Demux Symbol **Function Table**

Logic Diagram

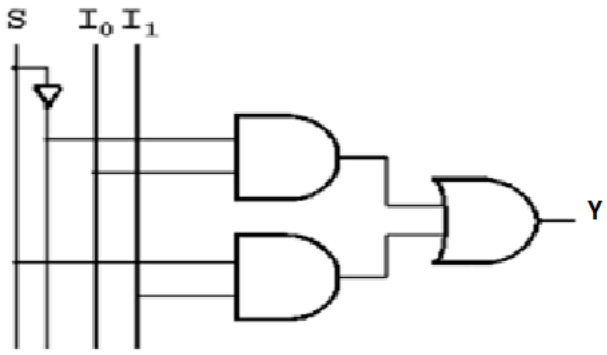


Figure 3.3.3 2:1 Multiplexer

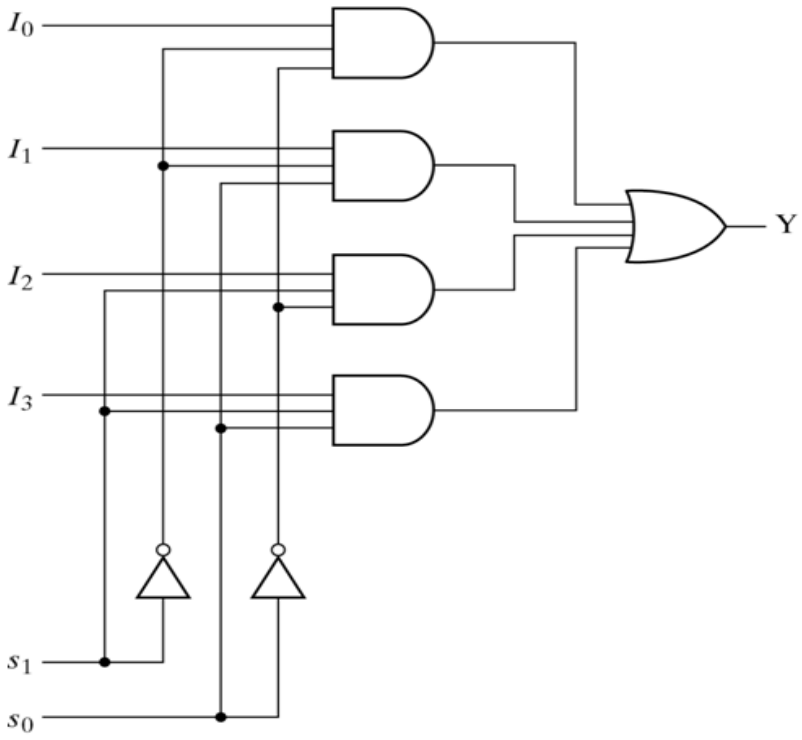


Figure 3.3.4 4:1 Multiplexer

3.4 Pre lab Questions

1. Define mux and demux.
2. Write their applications.
3. What is the relationship b/w input lines and select lines.
4. Design 4:1 mux and 1:4 demux.
5. Write brief notes on case statement.

VERILOG Program

Multiplexers 2:1 MUX

Structural Model	Dataflow Model	BehaviouralModel

module mux21str(i0,i1,s,y);
 	
input i0,i1,s;
 	
output y;
 	
wire net1,net2,net3;
 	
not g1(net1,s);
 	
and g2(net2,i1,s);
 	
and g3(net3,i0,net1);
 	
or g4(y,net3,net2);
 	
endmodule
	module mux21df(i0,i1,s,y);
	
	input i0,i1,s;
	
	output y;
	
	assign y =(i0&(~s))|(i1&s);
	
	endmodule
		module mux21beh(i0,i1,s,y);
		
		input i0,i1,s;
		
		output y;
		
		reg y;
		
		always@(i0,i1)
		
		begin
		
		if(s==0) y=i1;
		
		if(s==1)y=i0;
		
		end
		
		endmodule

4:1 MUX

Structural Model	Dataflow Model	BehaviouralModel

module
mux41str(i0,i1,i2,i3,s0,s1,y);
input i0,i1,i2,i3,s0,s1;
 	
wire a,b,c,d;
 	
output y;
 	
and g1(a,i0,s0,s1);
 	
and g2(b,i1,(~s0),s1);
 	
and g3(c,i2,s0,(~s1));
 	
and g4(d,i3,(~s0),(~s1));
 	
or(y,a,b,c,d);
 	
endmodule
	module
	mux41df(i0,i1,i2,i3,s0,s1,y);
	
	input i0,i1,i2,i3,s0,s1;
	
	output y;
	
	assign
	y=((i0&(~(s0))&(~(s1)))|
	
	(i1&(~(s0))&s1)|
	
	|(i2&s0&(~(s1)))|
	
	(i3&s0&s1);
	
	endmodule
		module mux41beh(in,s,y);
		output y ;
		input [3:0] in ;
		input [1:0] s ;
		reg y;
		always @ (in,s)
		begin
		if (s[0]==0&s[1]==0)
		y = in[3];
		else if (s[0]==0&s[1]==1)
		y = in[2];
		else if (s[0]l==1&s[1]==0)
		y = in[1];
		else
		y = in[0];
		end
		endmodule

Logic Diagram

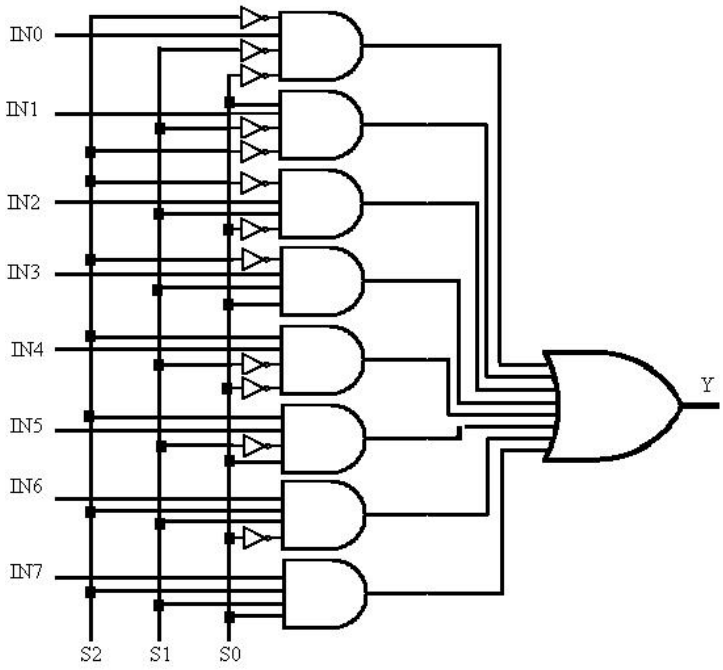


Figure 3.3.5 8:1 Multiplexer

VERILOG Program

8:1 MUX

Structural Model	Dataflow Model	BehaviouralModel

modulemux81str(i0,i1,i2,i3,i4,i5,i6,i7,s0,s1,s2,y);
 	
input i0,i1,i2,i3,i4,i5,i6,i7,s0,s1,s2;
 	
wire a,b,c,d,e,f,g,h;
 	
output y;
 	
and g1(a,i7,s0,s1,s2);
 	
and g2(b,i6,(~s0),s1,s2);
 	
and g3(c,i5,s0,(~s1),s2);
 	
and g4(d,i4,(~s0),(~s1),s2);
 	
and g5(e,i3,s0,s1,(~s2));
 	
and g6(f,i2,(~s0),s1,(~s2));
 	
and g7(g,i1,s0,(~s1),(s2));
 	
and g8(h,i0,(~s0),(~s1),(~s2));
 	
or(y,a,b,c,d,e,f,g,h);
 	
endmodule
	modulemux81df(y,i,s);
	
	output y;
	
	input [7:0] i;
	
	input [2:0] s;
	
	wire se1;
	
	assign se1=(s[2]*4)|(s[1]*2)|(s[0]);
	
	assign y=i[se1];
	
	endmodule
		modulemux81beh(s,i0,i1,i2,i3,i4,i5,i6,i7,y);
		input [2:0] s;
		input i0,i1,i2,i3,i4,i5,i6,i7;
		
		regy;
		always@(i0,i1,i2,i3,i4,i5,i6,i7,s) begin
		case(s) begin
		3'd0:MUX_OUT=i0;
		3'd1:MUX_OUT=i1;
		3'd2:MUX_OUT=i2;
		3'd3:MUX_OUT=i3;
		3'd4:MUX_OUT=i4;
		3'd5:MUX_OUT=i5;
		3'd6:MUX_OUT=i6;
		3'd7:MUX_OUT=i7;
		endcase
		end
		endmodule

Logic Diagram

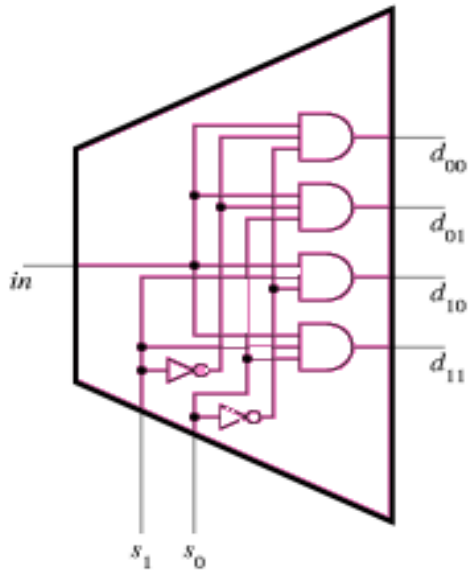


Figure 3.3.6 1:4 Demultiplexer

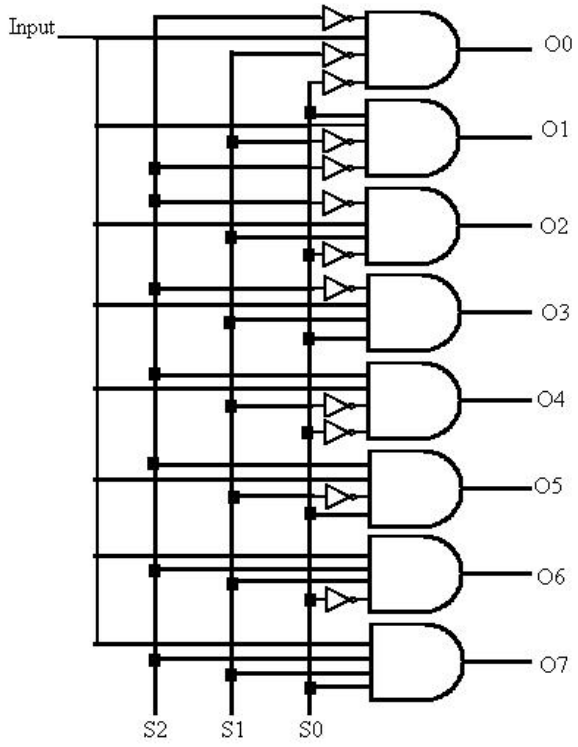


Figure 3.3.7 1:8 Demultiplexer

VERILOG Program

1:4 DEMUX

Structural Model	Dataflow Model	BehaviouralModel

module demux14str(in,d0,d1,d2,d3,s0,s1);
 	
output d0,d1,d2,d3;
 	
input in,s0,s1;
 	
and g1(d0,in,s0,s1);
 	
and g2(d1,in,(~s0),s1);
 	
and g3(d2,in,s0,(~s1));
 	
and g4(d3,in,(~s0),(~s1));
 	
endmodule
	module demux14df(
	in,d0,d1,d2,d3,s0,s1);
	output d0,d1,d2,3;
	input in,s0,s1;
	assign s0 = in & (~s0) & (~s1);
	assign d1= in & (~s0) & s1;
	assign d2= in & s0 & (~s1);
	assign d3= in & s0 & s1;
	endmodule
		module demux14beh(
		din,sel,dout);
		output [3:0] dout ;
		reg [3:0] dout ;
		input din ;
		wire din ;
		input [1:0] sel ;
		wire [1:0] sel ;
		always @ (din or sel) begin
		case (sel)
		0 : dout = {din,3'b000};
		1 : dout = {1'b0,din,2'b00};
		2 : dout = {2'b00,din,1'b0};
		default : dout = {3'b000,din};
		endcase
		
		end
		
		endmodule

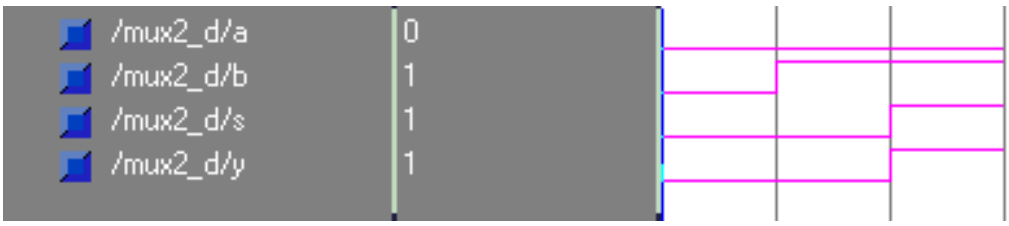
1:8 DEMUX

Structural Model	Dataflow Model	BehaviouralModel

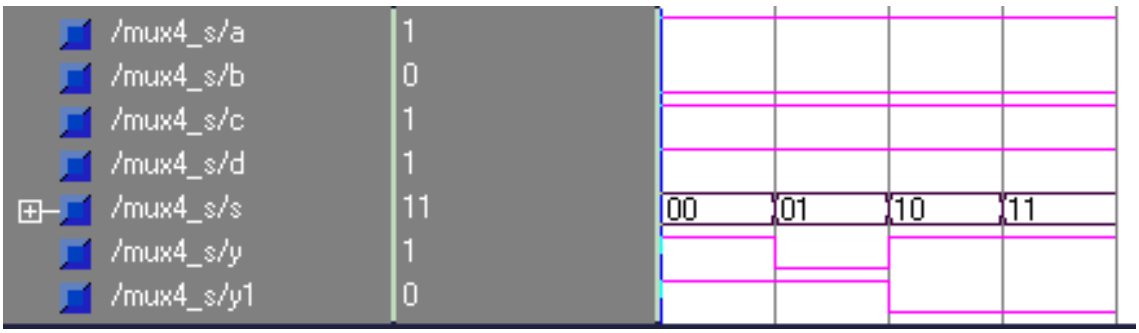
module demux18str(in,s0,s1,s2,d0,d1,d2,d3,d4,d5,d6,d7);
 	
input in,s0,s1,s2;
 	
output d0,d1,d2,d3,d4,d5,d6,d7;
 	
and g1(d0,in,s0,s1,s2);
 	
and g2(d1,in,(~s0),s1,s2);
 	
and g3(d2,in,s0,(~s1),s2);
 	
and g4(d3,in,(~s0),(~s1),s2);
 	
and g5(d4,in,s0,s1,(~s2));
 	
and g6(d5,in,(~s0),s1,(~s2));
 	
and g7(d6,in,s0,(~s1),(~s2));
 	
and g8(d7,in,(~s0),(~s1),(~s2));
 	
endmodule
	module demux18df(in,s0,s1,s2,i0,d1,d2,d3,d4,d5,d6,d7);
	
	input in,s0,s1,s2;
	
	output d0,d1,d2,d3,d4,d5,d6,d7;
	
	assign d0 = in & s0 & s1 & s2;
	
	assign d1 = in & (~s0) & s1 & s2;
	
	assign d2 = in & s0 & (~s1) & s2;
	
	assign d3 = in & (~s0) &(~s1) & s2;
	
	assign d4 = in & s0 & s1 & (~s2);
	
	assign d5 = in & (~s0) & s1 & (~s2);
	
	assign d6 = in & s0 & (~s1) & (~s2);
	
	assign d7 = in & (~s0) & (~s1) & (~s2);
	
	endmodule
		module demux18beh(i, sel, y);
		
		input i;
		
		input [2:0] sel;
		
		output [7 :0] y ;
		
		reg [7:0] y;
		
		always@(i,sel)
		
		begin
		
		y=8'd0;
		
		case(sel)
		
		3'd0:y[0]=i;
		
		3'd1:y[1]=i;
		
		3'd2:y[2]=i;
		
		3'd3:y[3]=i;
		
		3'd4:y[4]=i;
		
		3'd5:y[5]=i;
		
		3'd6:y[6]=i;
		
		default:y[7]=i;
		
		endcase
		
		end
		
		endmodule

Output Wave forms:

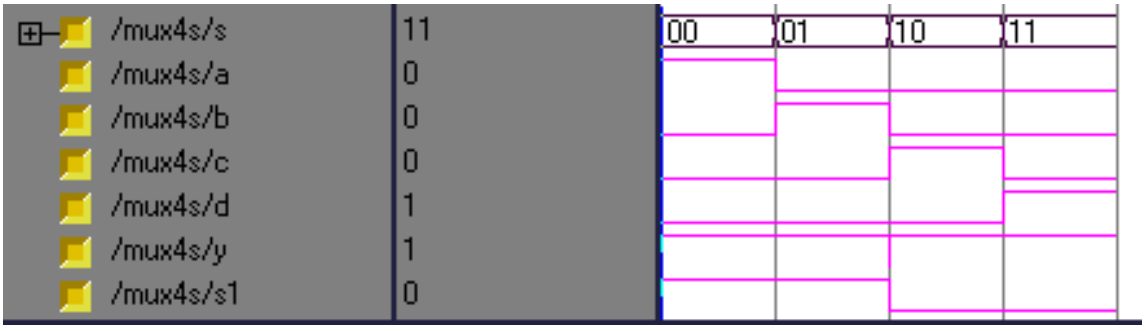
2 X 1 MUX:



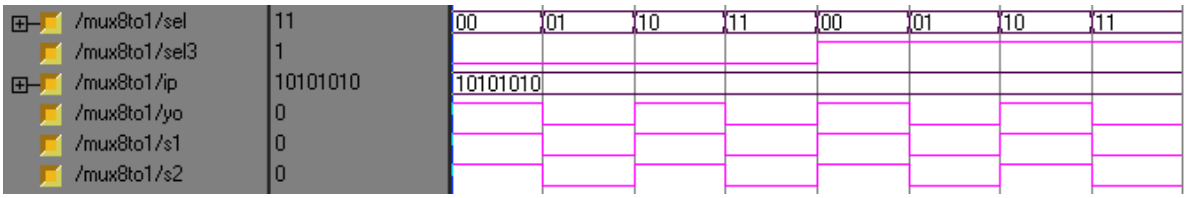
4 X 1 MUX **DATAFLOW MODELING**



4 X 1 MUX **STRUCTURAL MODELING**



MUX 8:1 Using 4:1 & 2:1



1 TO 4 DEMUX BEHAVIOURAL MODELING

3.5 Post Lab questions

1. Implement the function f(A,B,C)=Σm(0,1,3,5,7) by using Mux.

2. Write the VERILOG code for the above design

3. Write the VERILOG code for full subtractor using Demux.

3.6 Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab

Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

Lab cover sheet with staff verification sign.

Answer the pre-lab questions

Complete VERILOG code design for all logic gates and output signal waveforms

Answer the post-lab questions

3.7 Grading

Pre-lab Work	20 points
Lab Performance	30 points
Post-lab Work	20 points
Lab report	30 points

For the lab performance - at a minimum, demonstrate the operation of all the logic gates to your staff in-charge

The lab report will be graded as follows (for the 30 points):

VERILOG code for each experiments	15 points
Output signal waveform for all experiments and its truth table	15 points

EXP:4 Design of Encoders and Decoders

4.1 Introduction

The purpose of this experiment is to introduce you to the basics of Encoders and Decoders. In this lab, you have to implement Priority Encoder and the Boolean function using Decoders.

4.2 Software tools Requirement

Equipments:

Computer with Modelsim Software

Specifications:

HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Modelsim - 5.7c, Xilinx - 6.1i.

Algorithm

STEP 1: Open ModelSim XE II / Starter 5.7C

STEP 2: File -> Change directory -> D:\<register number>

STEP 3: File -> New Library -> ok

STEP 4: File -> New Source -> Verilog

STEP 5: Type the program

STEP 6: File -> Save -> <filename.v>

STEP 7: Compile the program

STEP 8: Simulate -> expand work -> select file -> ok

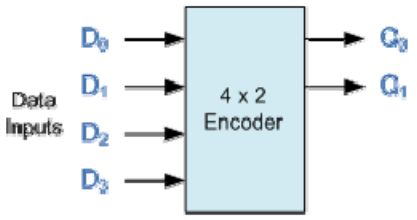
STEP 9: View -> Signals

STEP 10: Select values -> Edit -> Force -> input values

STEP 11: Add -> Wave -> Selected signals -> Run

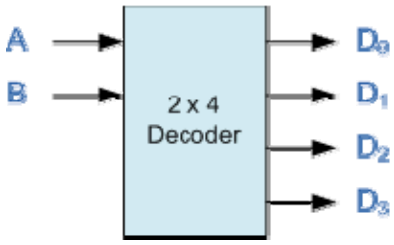
STEP 12: Change input values and run again

4.3 Logic Diagram



Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

Figure 4.3.1 **4-to-2 bit Encoder**



A	B	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Figure 4.3.2 **2-to-4 Binary Decoders**

4.4 Pre lab Questions

1. What is difference b/w encoder and data selector?
2. What is the difference b/w decoder and data distributor?
3. Give the applications of encoder and decoder.
4. Write short notes on “with – select” statement.
5. What are the different logic state systems in Verilog?

Logic Diagram

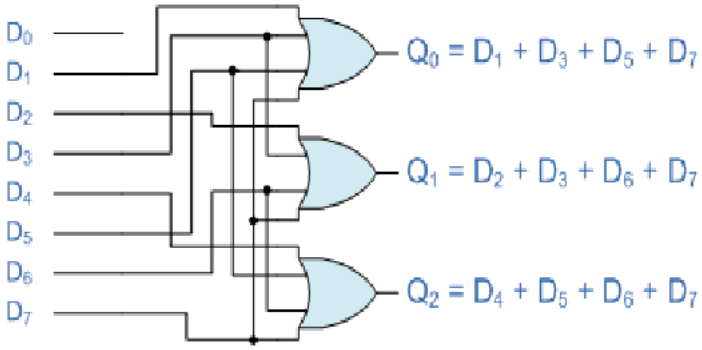


Figure 4.3.3 **8:3 Encoder**

VERILOG Program

8:3 Encoder

Structural Model	Data Flow Model	BehaviouralModel
Module enc83str(d0,d1,d2,d3,d4,d5,d6,d7,q0,q1,q2); Input d0,d1,d2,d3,d4,d5,d6,d7; Output q0,q1,q2; Or g1(q0,d1,d3,d5,d7); Or g2(q1,d2,d3,d6,d7); Or g3(q2,d4,d5,d6,d7); Endmodule	Module enc83df(d0,d1,d2,d3,d4,d5,d6,d7,q0,q1,q2); Input d0,d1,d2,d3,d4,d5,d6,d7; Output q0,q1,q2; Assign q0=d1|d3|d5|d7; Assign q1=d2|d3|d6|d7; Assign q2=d4|d5|d6|d7; Endmodule	module enc83beh (din,a,b,c); input [0:7]din; outputa,b,c; rega,b,c; always@(din) case(din) 8'b10000000:begin a=1'b0;b=1'b0,c=1'b0;end 8'b01000000:begin a=1'b0;b=1'b0;c=1'b1;end 8'b00100000:begin a=1'b0;b=1'b1;c=1'b0;end 8'b00010000:begin a=1'b0;b=1'b1;c=1'b1;end 8'b10001000:begin a=1'b1;b=1'b0,c=1'b0;end 8'b10000100:begin a=1'b1;b=1'b0,c=1'b1;end 8'b10000010:begin a=1'b1;b=1'b1,c=1'b0;end 8'b10000001:begin a=1'b1;b=1'b1,c=1'b1;end endcase endmodule

Logic Diagram

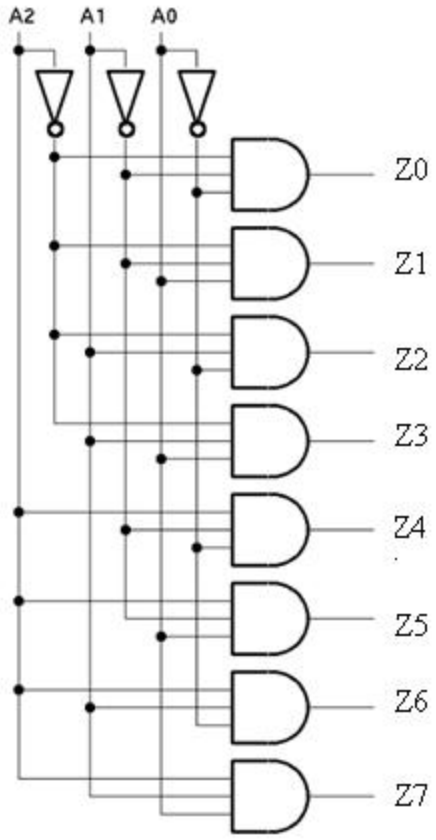


Figure 4.3.4 3:8 Decoder

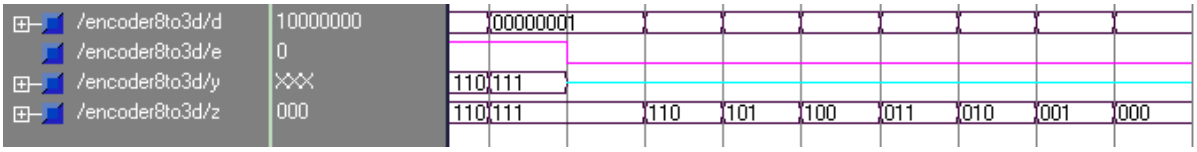
VERILOG Program

3:8 Decoder

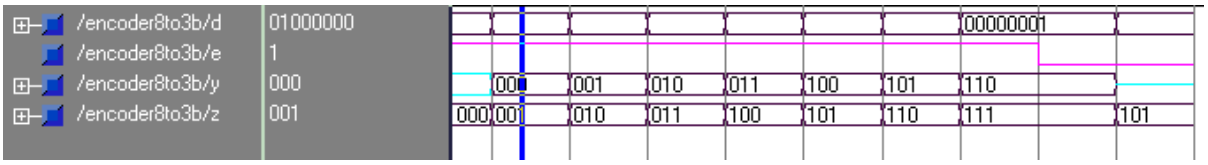
Structural Model	Data Flow Model	BehaviouralModel

module decoder38str(z0,z1,z2,z3,z4,z5,z6,z7,a0,a1,a2);
 	
output z0,z1,z2,z3,z4,z5,z6,z7;
 	
input a0,a1,a2;
 	
not (s0,a0);
 	
not (s1,a1);
 	
not (s2,a2);
 	
and (z0,s0,s1,s2);
 	
and (z1,a0,s1,s2);
 	
and (z2,s0,a1,s2);
 	
and z3,a0,a1,s2);
 	
and (z4,s0,s1,a2);
 	
and (z5,a0,s1,a2);
 	
and (z6,s0,a1,a2);
 	
and (z7,a0,a1,a2);
 	
endmodule
	module decoder38df(z,a0,a1,a2);
	
	output [7:0] z;
	
	input a0,a1,a2;
	
	assign z[0] = ~a0 & ~a1 & ~a2;
	
	assign z[1] = ~a0& ~a1& a2;
	
	assign z[2] = ~a0& a1& ~a2;
	
	assign z[3] = ~a0& a1& a2;
	
	assign z[4] = a0& ~a1& ~a2;
	
	assign z[5] = a0& ~a1& a2;
	
	assign z[6] = a0& a1& ~a2;
	
	assign z[7] = a0& a1& a2;
	
	endmodule
		module decoder38beh(sel,out1);
		
		input [2:0] sel;
		outputreg [7:0] out1;
		
		always @(sel,out1)
		case (sel)
		3'b000 : out1 = 8'b00000001;
		3'b001 : out1 = 8'b00000010;
		3'b010 : out1 = 8'b00000100;
		3'b011 : out1 = 8'b00001000;
		3'b100 : out1 = 8'b00010000;
		3'b101 : out1 = 8'b00100000;
		3'b110 : out1 = 8'b01000000;
		default : out1 = 8'b10000000;
		endcase
		
		endmodule

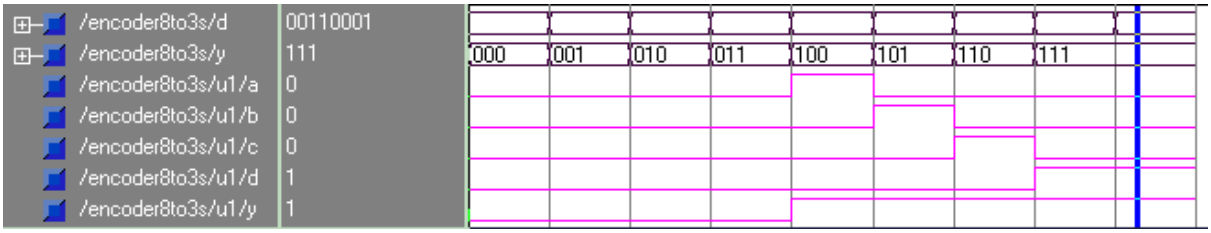
Output Wave forms:



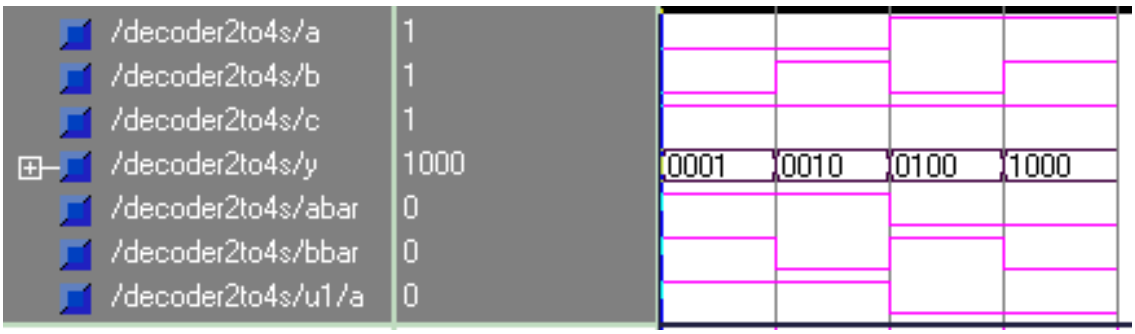
ENCODER 8:3 Data Flow Modeling



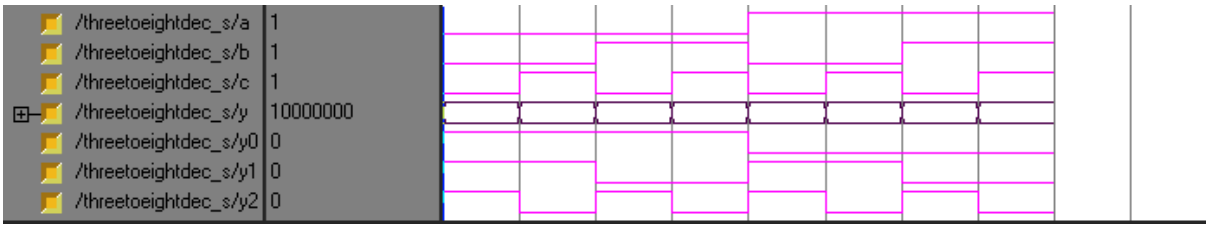
ENCODER 8:3 Behavioral Modeling



ENCODER 8:3 Structural Modeling



DECODER 2:4 Structural Modeling



DECODER 3:8 Structural Modeling

4.5 Post Lab questions

1. Implement full adder by using suitable decoder.

2. Write the VERILOG code for the above design

3. Write the VERILOG code for 3 bit Gray to binary code converter.

4. Write short notes on “test bench” with examples.

4.6 Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab

Report Requirements"document available on the class web page. Be sure to include the following items in your lab report:

Lab cover sheet with staff verification sign.

Answer the pre-lab questions

Complete VERILOG code design for all logic gates and output signal waveforms

Answer the post-lab questions

4.7 Grading

Pre-lab Work	20 points
Lab Performance	30 points
Post-lab Work	20 points
Lab report	30 points

For the lab performance - at a minimum, demonstrate the operation of all the logic gates to your staff in-charge

The lab report will be graded as follows (for the 30 points):

VERILOG code for each experiments	15 points
Output signal waveform for all experiments and its truth table	15 point

EXP 5: Flip Flops

5.1 Introduction

The purpose of this experiment is to introduce you to the basics of flip-flops. In this lab, you will test the behavior of several flip-flops and you will connect several logic gates together to create simple sequential circuits.

5.2 Software tools Requirement

Equipments:

Computer with Modelsim Software

Specifications:

HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Modelsim - 5.7c, Xilinx - 6.1i.

Algorithm

STEP 1: Open ModelSim XE II / Starter 5.7C

STEP 2: File -> Change directory -> D:\<register number>

STEP 3: File -> New Library -> ok

STEP 4: File -> New Source -> Verilog

STEP 5: Type the program

STEP 6: File -> Save -> <filename.v>

STEP 7: Compile the program

STEP 8: Simulate -> expand work -> select file -> ok

STEP 9: View -> Signals

STEP 10: Select values -> Edit -> Force -> input values

STEP 11: Add -> Wave -> Selected signals -> Run

STEP 12: Change input values and run again

5.3 Flip-Flops Logic diagram and their properties

Flip-flops are synchronous bitable devices. The term synchronous means the output changes state only when the clock input is triggered. That is, changes in the output occur in synchronization with the clock.

A flip-flop circuit has two outputs, one for the normal value and one for the complement value of the stored bit. Since memory elements in sequential circuits are usually flip-flops, it is worth summarizing the behavior of various flip-flop types before proceeding further.

All flip-flops can be divided into four basic types: SR, JK, D and T. They differ in the number of inputs and in the response invoked by different value of input signals. The four types of flip-flops are defined in the Table 5.1. Each of these flip-flops can be uniquely described by its graphical symbol, its characteristic table, its characteristic equation or excitation table. All flip-flops have output signals Q and Q'.

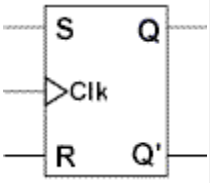
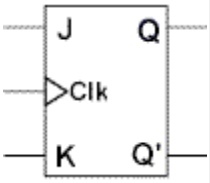
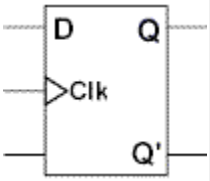
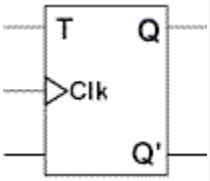
Flip-Flop Name	Flip-Flop Symbol	Characteristic Table	Characteristic Equation	Excitation Table
SR		S R Q(next) 0 0 Q 0 1 0 1 0 1 1 1 ?	Q(next) = S + R'Q SR = 0	Q Q(next) S R 0 0 0 X 0 1 1 0 1 0 0 1 1 1 X 0
JK		J K Q(next) 0 0 Q 0 1 0 1 0 1 1 1 Q'	Q(next) = JQ' + K'Q	Q Q(next) J K 0 0 0 X 0 1 1 X 1 0 X 1 1 1 X 0
D		D Q(next) 0 0 1 1	Q(next) = D	Q Q(next) D 0 0 0 0 1 1 1 0 0 1 1 1
T		T Q(next) 0 Q 1 Q'	Q(next) = TQ' + T'Q	Q Q(next) T 0 0 0 0 1 1 1 0 1 1 1 0

Table 5.3 Flip-flops and their properties

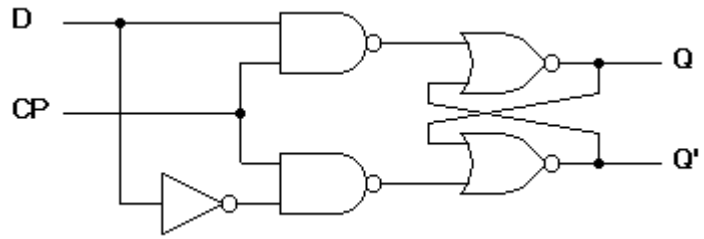


Figure 5.3.1 D- Flip Flop

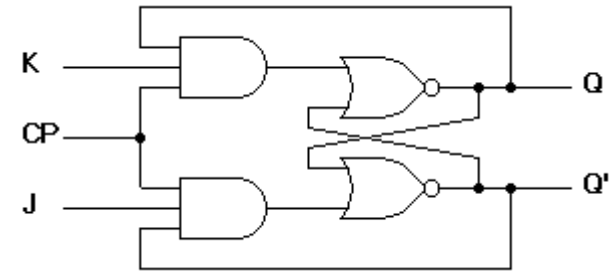


Figure 5.3 .2 JK Flip Flop

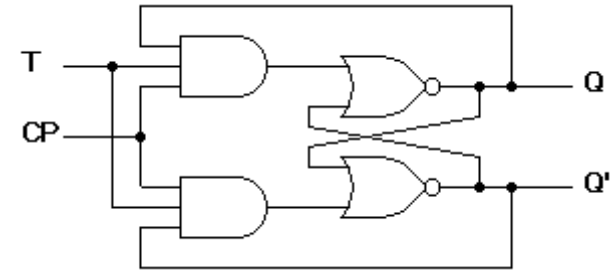


Figure 5.3.3 T Flip Flop

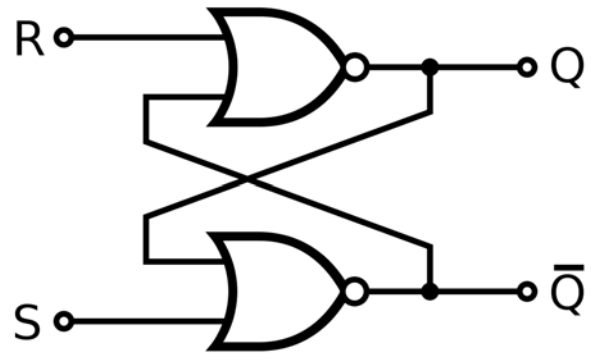


Figure 5.3.4 T Flip Flop

5. 4 Pre-lab Questions

1. Describe the main difference between a gated S-R latch and an edge-triggered S-R flip-flop.

2. How does a JK flip-flop differ from an SR flip-flop in its basic operation?

3. Describe the basic difference between pulse-triggered and edge-triggered flip-flops.

4. What is use of characteristic and excitation table?

5. What are synchronous and asynchronous circuits?

6. How many flip flops due you require storing the data 1101?

S-R Flip Flop

Dataflow Modelling Modelling

```
modulesr_df (s, r, q, q_n);
input s, r;
output q, q_n;
assignq_n = ~(s | q);
assign q = ~(r | q_n);
endmodule
```

Structural Modelling

```
module sr_st(s,r,q,q_n);
input s, r;
output q, q_n;
or g1(q_n,~s,~q);
or g2(q,~r,~q_n);
endmodule
```

Behavioral

```
module sr_beh(s,r,q,q_n);
input s, r;
output q, q_n;
regq, q_n;
always@(s,r)
begin
q,n = ~(s|q);
assign q = ~(r | q_n);
endmodule
```

T Flip Flop

Behavioral Modelling

```
module t_beh(q,q1,t,c);
output q,q1;
inputt,c;
reg q,q1;
initial
begin
q=1'b1;
q1=1'b0;
end
always @ (c)
begin
        if(c)
        begin
        if (t==1'b0) begin q=q; q1=q1; end
        else begin q=~q; q1=~q1; end
        end
end
endmodule
```

Structural Modelling

```
module t_st(q,q1,t,c);
output q,q1;
input t,c;
wire w1,w2;
assign w1=t&c&q;
assign w2=t&c&q1;
assign q=~(w1|q1);
assign q1=~(w2|q);
endmodule
```

Dataflow Modelling

```
module t_df(q,q,1,t,c);
output q,q1;
input t,c;
and g1(w1,t,c,q);
and g2(w2,t,c,q1);
nor g3(q,w1,q1);
nor g4(q1,w2,q);
endmodule
```

D Flip Flop

Behavioral Modelling	*Dataflow Modelling*	*Structural Modelling*

Module dff_async_reset(data, clk, reset ,q);
 	
input data, clk, reset ;
output q;
reg q;
always @ (posedgeclk or negedge reset)
 	
if (~reset) begin
 	
q <= 1'b0;
 	
end
 	
else begin
 	
q <= data;
 	
end
 	
endmodule
	module dff_df(d,c,q,q1);
	
	input d,c;
	
	output q,q1;
	
	assign w1=d&c;
	
	assign w2=~d&c;
	
	q=~(w1|q1);
	
	q1=~(w2|q);
	
	endmodule
		module dff_df(d,c,q,q1);
		
		input d,c;
		
		output q,q1;
		
		and g1(w1,d,c);
		
		and g2(w2,~d,c);
		
		nor g3(q,w1,q1);
		
		nor g4(q1,w2,q);
		
		endmodule

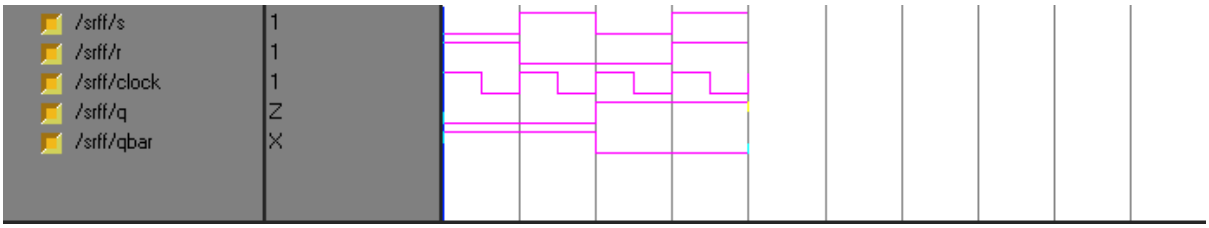
JK Flip Flop

Behavioral Modelling	*Dataflow Modelling*	*Structural Modelling*

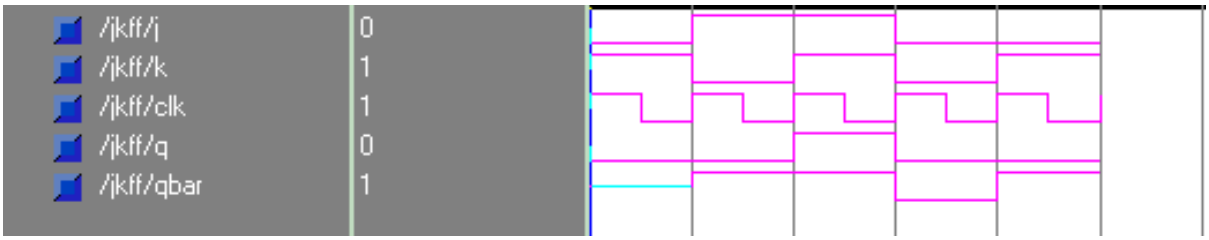
module jk(q,q1,j,k,c);
output q,q1;
input j,k,c;
reg q,q1;
initial begin q=1'b0; q1=1'b1; end
always @ (posedge c)
begin
case({j,k})
{1'b0,1'b0}:begin q=q; q1=q1; end
{1'b0,1'b1}: begin q=1'b0; q1=1'b1; end
{1'b1,1'b0}:begin q=1'b1; q1=1'b0; end
{1'b1,1'b1}: begin q=~q; q1=~q1; end
endcase
end
endmodule
	module jkflip_df (j,k,q,qn);
	input j,k,q;
	output qn;
	wire w1,w2;
	assign w1=~q;
	assign w2=~k;
	assign qn=(j & w1 | w2 & q);
	endmodule
		module jkflip_st(j,k,q,qn);
		input j,k,q;
		output qn;
		and g1(w1,j,~q);
		and g2(w2,~k,q);
		or g3(qn,w1,w2);
		endmodule

Output Waveforms :

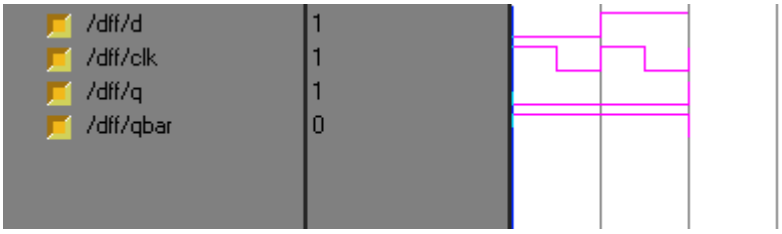
SR flip flop



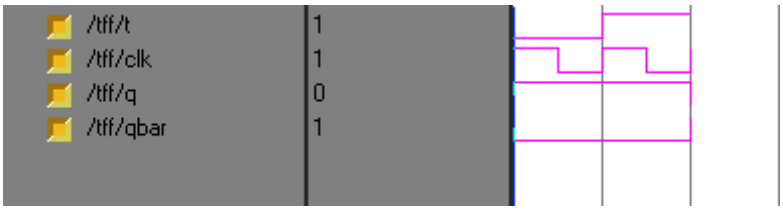
JK flip flop



D flip flop



T flip flop



5.5 Post lab

1. Discuss the application of flip-flops in data storage.

2. Draw the logic diagram of Master Slave JK flip-flop.

3. A flip-flop is presently in the RESET state and must go to the SET state on the next clock pulse. What must J and K be?

4. What do you know about clk and clk event in VERILOG?

5. Convert the following.

    a. JK to T f/f

    b. SR to D

6. Write the VERILOG code for question no 5.

5.6 Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab

Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

Lab cover sheet with staff verification for circuit diagram

Answer the pre-lab questions

Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.

Answer the post-lab questions

5.7 Grading

Pre-lab Work	20 points
Lab Performance	30 points
Post-lab Work	20 points
Lab report	30 points

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff in-charge

The lab report will be graded as follows (for the 30 points):

VERILOG code for each experiments	15 points
Output signal waveform for all experiments and its truth table	15 points

EXP 6: Counters

6.1 Introduction

The purpose of this experiment is to introduce the design of Synchronous Counters. The student should also be able to design n-bit up/down Counter.

6.2 Software tools Requirement

Equipments:

Computer with Modelsim Software

Specifications:

HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Modelsim - 5.7c, Xilinx - 6.1i.

Algorithm

STEP 1: Open ModelSim XE II / Starter 5.7C

STEP 2: File -> Change directory -> D:\<register number>

STEP 3: File -> New Library -> ok

STEP 4: File -> New Source -> Verilog

STEP 5: Type the program

STEP 6: File -> Save -> <filename.v>

STEP 7: Compile the program

STEP 8: Simulate -> expand work -> select file -> ok

STEP 9: View -> Signals

STEP 10: Select values -> Edit -> Force -> input values

STEP 11: Add -> Wave -> Selected signals -> Run

STEP 12: Change input values and run again

6.3 Logic Diagram

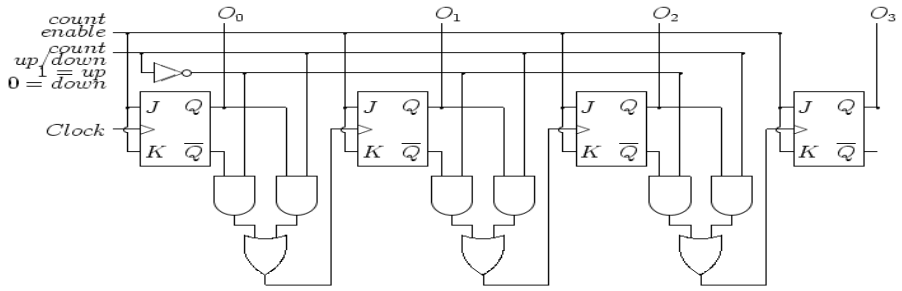


Figure 6.3.1 Updown Counter

6.4 Pre Lab questions

1. How does synchronous counter differ from asynchronous counter?

2. How many flip-flops do you require to design Mod-6 counter.

3. What are the different types of counters?

4. What are the different types of shift registers?

5. How many f/fs are needed for n-bit counter?

6. What is meant by universal shift register?

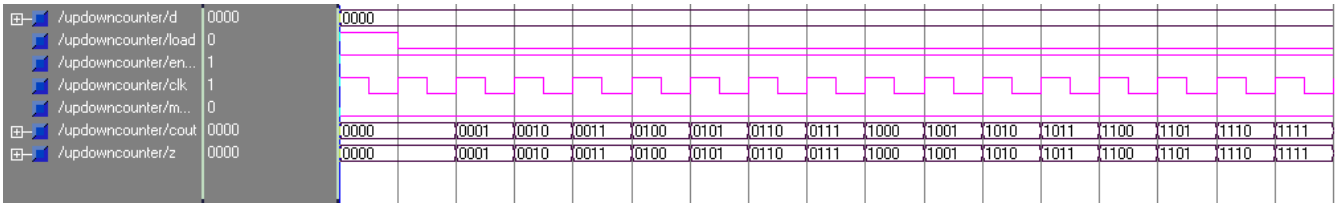
VERILOG Program

Up Down Counter

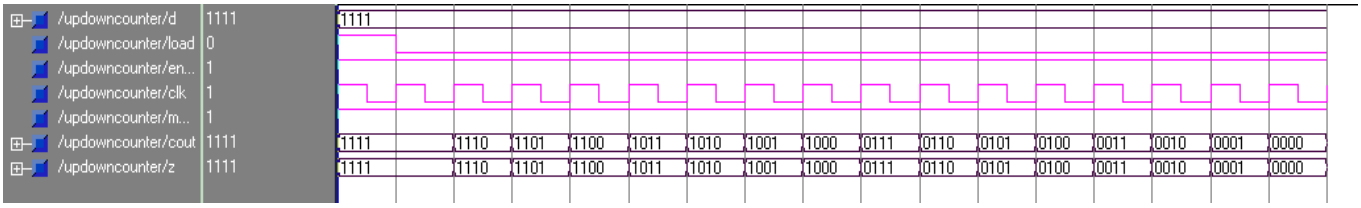
```
moduleupdown(out,clk,reset,updown);

output [3:0]out;

inputclk,reset,updown;

reg [3:0]out;

always @(posedgeclk)

if(reset) begin

out<= 4'b0;

end else if(updown) begin

out<=out+1;

end else begin

out<=out-1;

end

endmodule
```

Output Waveform:

UP COUNTER



DOWN COUNTER



6.5 Post Lab questions

1. Write the use of enable and reset signal.

2. What are the different types of shift registers?

3. How many f/fs are needed for n-bit counter?

4. What is the function of generic statement?

5. Design mod-6 counter using d flf and write the VERILOG code.

6.6 Lab Report

Each individual will be required to submit a lab report. Use the format specified in the Lab

Report Requirements document available on the class web page. Be sure to include the following items in your lab report:

Lab cover sheet with staff verification for circuit diagram

Answer the pre-lab questions

Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.

Answer the post-lab questions

6.7 Grading

Pre-lab Work	20 points
Lab Performance	30 points
Post-lab Work	20 points
Lab report	30 points

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff in-charge

The lab report will be graded as follows (for the 30 points):

VERILOG code for each experiments	15 points
Output signal waveform for all experiments and its truth table	15 points

EXP 7: Toggle a Port bit in 8051

7.1 Introduction

The purpose of this experiment is to Toggle a Port bit in 8051. The student should also be able to control Port Pin in 8051.

7.2 Software tools Requirement

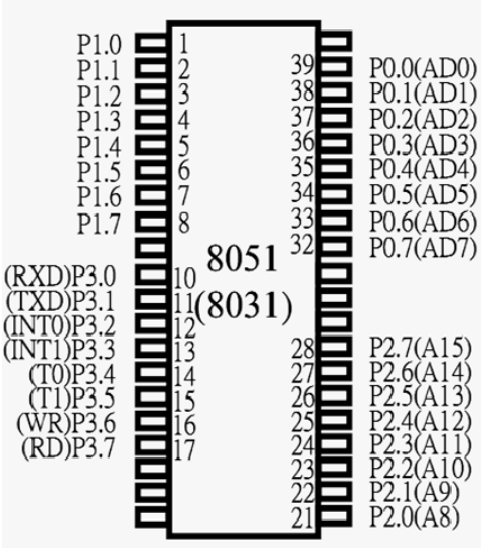
Equipments:

Computer with Keil µversion II Software

Specifications: HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Keil µversion II

7.3 Pin Description of 8051



7.4 Pre lab questions

1. Write an 8051 C program to toggle bits of P1 continuously forever with some delay.

2. What is the use of Watchdog timer?

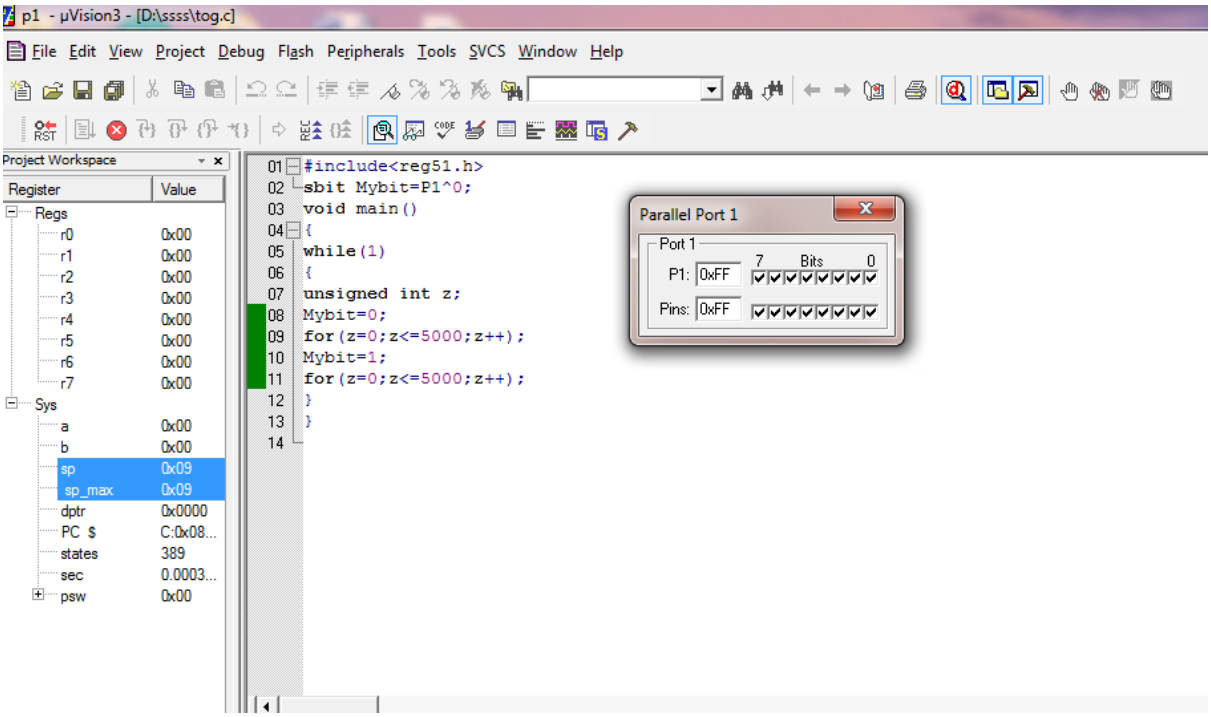
3. What is sbit, sbyte?

4. What is DPTR?

5. What is Power ON Reset?

7.5 Embedded C Program

```
#include<reg51.h>

sbit Mybit=P1^0;

void main()

{

while(1)

{

unsigned int z;

Mybit=0;

for(z=0;z<=5000;z++);

Mybit=1;

for(z=0;z<=5000;z++);

}

}
```



7.6 Post lab:

1. A door sensor is connected to the P1.1 pin, and a buzzer is connected to P1.7. Write an 8051 C program to monitor the door sensor, and when it opens, sound the buzzer. You can sound the buzzer by sending a square wave of a few hundred Hz.

2. Write an 8051 C program to get the status of bit P1.0, save it, and send it to P2.7 continuously.

7.7 Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab

Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

Lab cover sheet with staff verification for circuit diagram

Answer the pre-lab questions

Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.

Answer the post-lab questions

7.8 Grading

Pre-lab Work	20 points
Lab Performance	30 points
Post-lab Work	20 points
Lab report	30 points

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff in-charge

The lab report will be graded as follows (for the 30 points):

Embedded C code for each experiments	15 points
Output signal for all experiments and its model calculation	15 points

EXP 8: Bitwise Operators Using 8051

8.1 Introduction

The purpose of this experiment is to implement bitwise operators using 8051. The student should also be able to implement Logical Operations in 8051.

8.2 Software tools Requirement

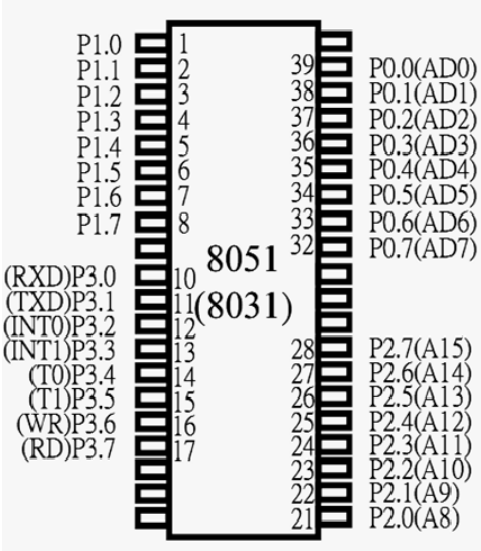
Equipments:

Computer with Keil µversion II Software

Specifications: HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Keil µversion II

8.3 Pin Description of 8051



8.4 Pre lab questions

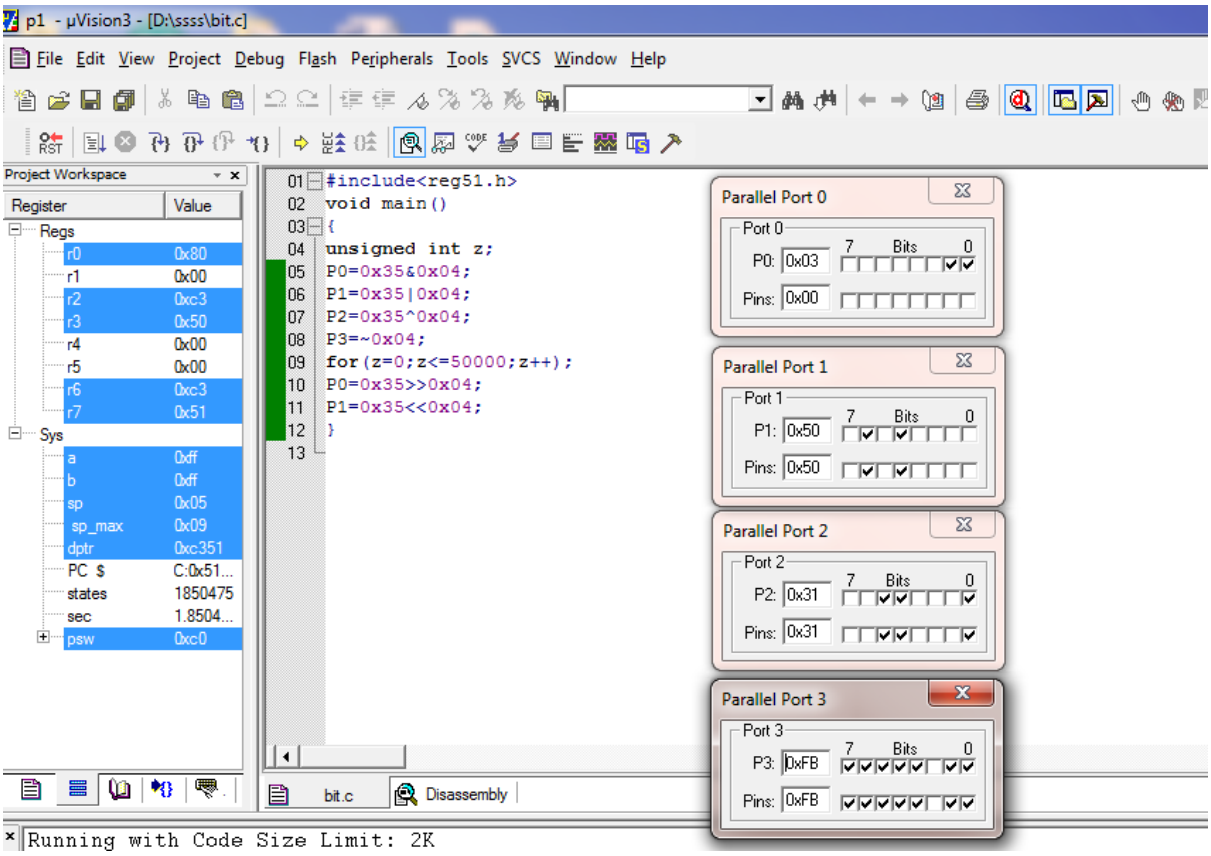
1. What are the advantages of using Bitwise operations ?

2. Which bitwise operator is suitable for turning off a particular bit in a number?

3. Which bitwise operator is suitable for checking whether a particular bit is on or off?

8.5 Embedded C Program

```
#include<reg51.h>

void main()

{

unsigned int z;

P0=0x35&0x04;

P1=0x35|0x04;

P2=0x35^0x04;

P3=~0x04;

for(z=0;z<=50000;z++);

P0=0x35>>0x04;

P1=0x35<<0x04;

}
```



8.6 Post lab:

1. Which bitwise operator is suitable for turning on a particular bit in a number?.

2. Write the Embedded C Program for Bit Operations.

8.7 Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab

Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

Lab cover sheet with staff verification for circuit diagram

Answer the pre-lab questions

Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.

Answer the post-lab questions

8.8 Grading

Pre-lab Work 20 points

Lab Performance 30 points

Post-lab Work 20 points

Lab report 30 points

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff in-charge

The lab report will be graded as follows (for the 30 points):

Embedded C code for each experiments 15 points

Output signal for all experiments and its model calculation 15 points

EXP 9: Arithmetic Operations using 8051

9.1 Introduction

The purpose of this experiment is to implement Arithmetic Operations in 8051. The student should also be able to implement Logical Operations in 8051.

9.2 Software tools Requirement

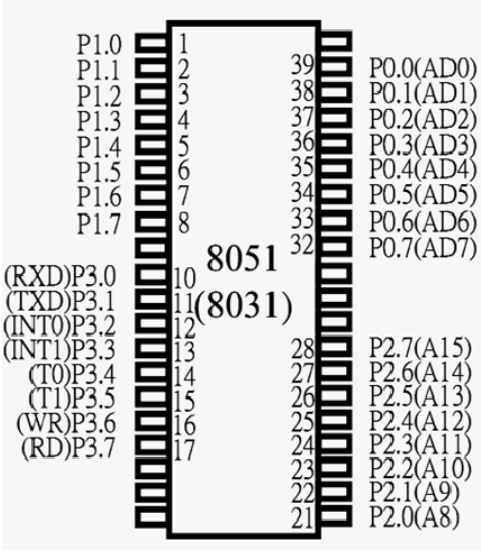
Equipments:

Computer with Keil μversion II

Software Specifications: HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Keil μversion II

9.3 Pin Description of 8051



9.4 Pre lab questions

1. What are the basic units of a microprocessor ?

2. List the features of 8051.

3. What are the types of serial communication ?

4. Define Program Counter.

9.5 Embedded C Program

```
#include<reg51.h>

void main()

{

int a,b,c;

while(1)

{

a=P0;

b=P1;c=P2;

switch(c)

{
```

```
case 0X00:

P3=a+b;

break;

case 0x01:

P3=a-b;

break;

case 0x02:

3=a*b;

break;

case 0x03:

P3=a*a;

break;

}

}

}
```

9.6 Postlab:

1. List the various registers used in 8051.

2. What is program status word.

3. Define stack pointer.

9.7 Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab

Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

Lab cover sheet with staff verification for circuit diagram

Answer the pre-lab questions

Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.

Answer the post-lab questions

9.8 Grading

Pre-lab Work	20 points
Lab Performance	30 points
Post-lab Work	20 points
Lab report	30 points

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff in-charge

The lab report will be graded as follows (for the 30 points):

Embedded C code for each experiments	15 points
Output signal for all experiments and its model calculation	15 points

EXP 10: Delay Operators in 8051

10.1 Introduction

The purpose of this experiment is to introduce delay operators 8051. The student should also be able to write ISR for various Interrupts in 8051.

10.2 Software tools Requirement

Equipments:

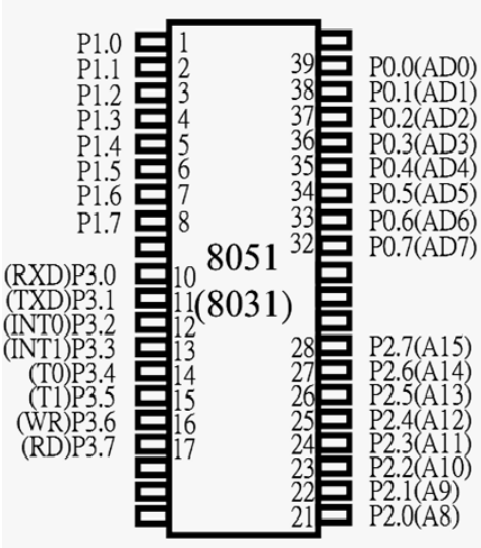
Computer with Keil µversion II Software

Specifications:

HP Computer P4 Processor – 2.8 GHz, 2GB RAM, 160 GB Hard Disk

Softwares: Keil µversion II

10.3 Pin Description of 8051



10.4 Pre lab questions

1. Write an 8051 C program to gets a single bit of data from P1.7 and sends it to P1.0.

2. What is ISR?

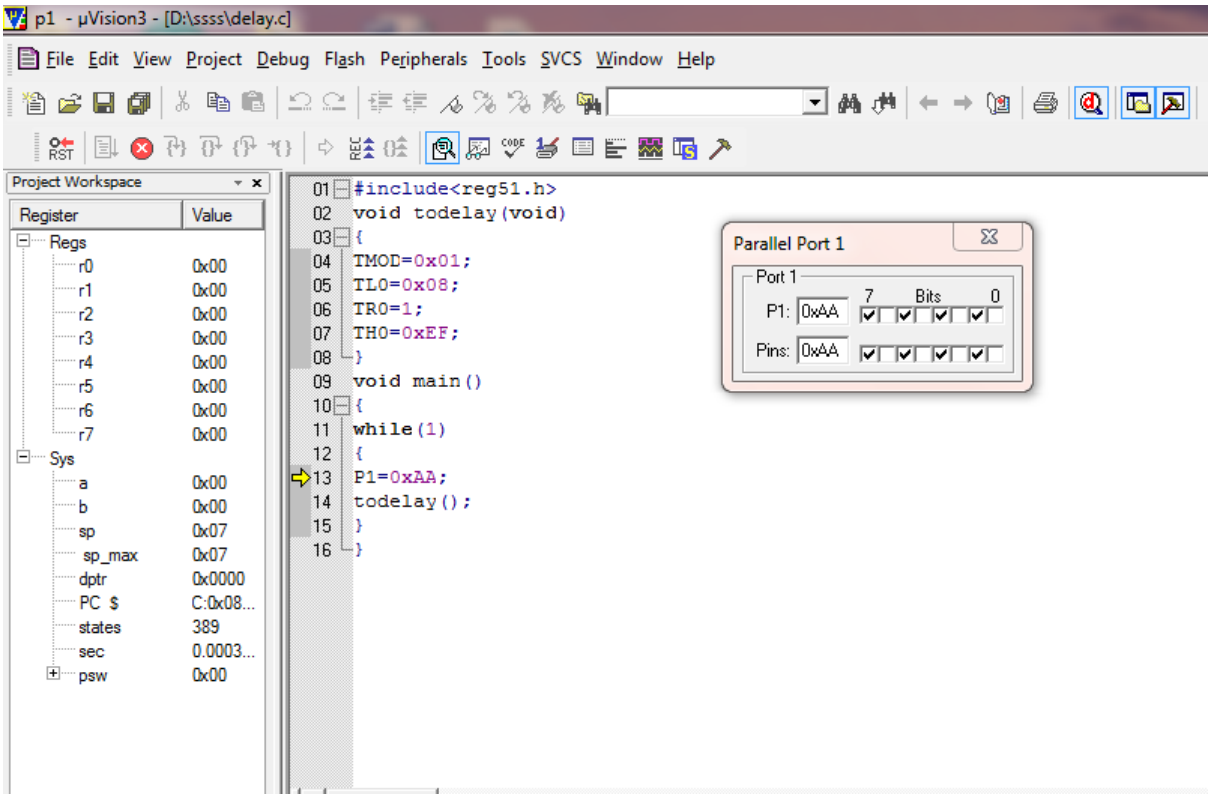
3. Name the two ways to access Interrupts?

4. What is Power ON Reset?

10.5 Embedded C Program

```
#include<reg51.h>

void todelay(void)

{

TMOD=0x01;

TL0=0x08;
```

```
TR0=1;

TH0=0xEF;

}

void main()

{

while(1)

{

P1=0xAA;

todelay();

}

 }
```



10.6 Postlab:

1. Compare Microprocessor and Microcontroller.

2. What are the basic units of a microcontroller.?

3. Define instruction set.

10.7 Lab Report

Each individual will be required to submit a lab report. Use the format specified in the "Lab

Report Requirements" document available on the class web page. Be sure to include the following items in your lab report:

Lab cover sheet with staff verification for circuit diagram

Answer the pre-lab questions

Complete paper design for all three designs including K-maps and minimized equations and the truth table for each of the output signals.

Answer the post-lab questions

10.8 Grading

Pre-lab Work	20 points
Lab Performance	30 points
Post-lab Work	20 points
Lab report	30 points

For the lab performance - at a minimum, demonstrate the operation of all the circuits to your staff in-charge

The lab report will be graded as follows (for the 30 points):

Embedded C code for each experiments	15 points
Output signal for all experiments and its model calculation	15 points