



King Faisal University
College of Engineering

Electrical Engineering Department

Digital Logic Design Lab (EE232) Manual

(elaborated by Mashhoor Altarayrah & Slim Chtourou)

Department Chair: **Dr.Mohammad Alarfag**

Course Teacher : **Mashhoor Altarayrah**

Lab Teacher : **Mashhoor Altarayrah**

Academic Year : 2020-2021

Major Topics covered and schedule in weeks:

Topic	Week #	Courses Covered
Introduction and Lab safety.	1	Introduction
Introduction to Digital logic Lab and equipment	2	EE 231
Analyzing and Verifying the Behavior of Logic Gates	3	EE 231
Analyzing and Verifying the Behavior of Logic Gates (Cont.)	4	EE 231
7-Segment Display and Hexadecimal Driver/Decoder	5	EE 231
Using Multisim software as a digital logic design simulator	6	EE 231
Using Multisim software as a digital logic design simulator (Cont.)	7	EE 231
K-Maps and Boolean Algebra	8	EE 231
K-Maps and Boolean Algebra (Cont.)	9	EE 231
Combinational logic Design and Analysis	10	EE 231
Combinational logic Design and Analysis(Cont.)	11	EE 231
Sequential logic Design and Analysis	12	EE 231
Sequential logic Design and Analysis(Cont.)	13	EE 231
Sequential logic Design and Analysis(Cont.)	14	EE 231
Final Exam	15	

Specific Outcomes of Instruction (Lab Learning Outcomes):

١. An ability to analyze and verify the behavior of logic gates (1,3,5,6).
٢. An ability to analyze and use 7-segment display and hexadecimal driver (1,3,5,6).
٣. An ability to use Multisim software to simulate and verify logic gates (1,3,5,6).
٤. An ability to simulate logic circuits using Multisim software. (1,3,5,6).
٥. An ability to build combinational circuits using K-map and Boolean algebra (1, 3, 5, 6).
٦. An ability to design combinational circuits using hardware Kit (1,3,5,6).
٧. An ability to analyze and build flip flops (1,3,5,6).
8. An ability to design sequential circuits using hardware Kit (1,3,5,6).

Student Outcomes (SO) Addressed by the Lab:

z	Outcome Description	Contribution
	General Engineering Student Outcomes	
1.	an ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics	M
2.	an ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors	
3.	an ability to communicate effectively with a range of audiences	L
4.	an ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts	
5.	an ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives	L
6.	an ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions	H
7.	an ability to acquire and apply new knowledge as needed, using appropriate learning strategies	

Table of Content

Contents

Week 3:Experiment 1: Analyzing and Verifying the Behavior of Logic Gates Part-1 : AND, OR, NOT(2, 3, 4 Inputs)	1
Week 4:Experiment 2: Analyzing and Verifying the Behavior of Logic Gates part-2 : NAND, NOR, XOR, XNOR (2, 3, 4 Inputs)	4
Week 5:Experiment 3: 7-Segment Display and Hexadecimal Driver/Decoder: Having Hex Numbers in SSD and using SSD as BCD/Hex Decoder	7
Week 6:Experiment 4: Digital Logic Circuit Modeling & Simulation with Multisim: Introduction and tutorial training of Multisim For Logic Design- TTL Family).....	11
Week 7:Experiment 5: Building and Designing a logic circuits Using Multisim: Adder & Comparator.....	14
Week 8:Experiment 6: Logic Circuit Design Using K-Maps and Boolean Algebra: 3-bits Full Adder and 2-bits Square Taking.....	18
Week 9:Experiment 7: Design and Conduct an Experiment for a simple logic circuit that calculate the Absolute of a Complex Number X: Design and verify a small mathematical system with using digital logic components.	23
Weeks 10 & 11 :Experiment 8: Combinational logic Design and Analysis: Multiplexers (MUX) and De-multiplexers (DMUX) Units.....	24
Weeks 12 & 13: Experiment 9: Sequential logic Design and Analysis: Analyzing and Implementing Flip-Flops.....	31
Week 14 Experiment 10: Design and analyze the synchronous counter circuits: Up/Down Synchronous Counters Using Composed JK-Flip-flop.....	38
APPENDIXES	44
Appendix A: Using An Oscilloscope	44
Appendix B: MultiSim	61

Preface

I. Lab Policies & Issues

1. Attendance Policy:

In accordance with the University Regulations, it is the student's responsibility to be punctual and to attend all classes. If a student fails to attend 25% of the classes, he will be deprived from passing the course.

2. Cheating and Plagiarism Policy:

Plagiarism: Using the words, thoughts, ideas, results, etc., of another person in a written assignment, without acknowledging the source, as if it were the student's own work.

3. General Lab Policy:

- Lab reports will be submitted by each student (or Group) at the deadline specified by the instructor.
- Late reports will get 10% deduction for each late day.
- Cheating and plagiarism are completely prohibited.
- A student must attend the exam, otherwise he will get a zero in this exam unless he has a valid excuse.
- Lab reports should be written according to the instructions of Lab instructor.
- Laboratory and equipment maintenance is the responsibility of not only the Lab staff, but students also.
- The report will be graded on clarity, legibility, and content, not on length nor on the quality of the artwork.
- No laboratory makeup's will be allowed unless prior written documentation is obtained from the department coordinator of Students, certified physician, or Judge for jury duty.
- For approved makeup's, the lab makeup's must be done within one week of the lab unless written documentation from the approved sources listed above justify otherwise. It will be extremely difficult to schedule lab make ups due to the limited number of empty laboratory seats. Please plan accordingly.
- Be responsible for equipment and laboratory maintenance. For example:
 - Keep the lab and benches neat and organized.
 - Use the equipment properly.
 - Return instruments, manuals, tools, components, cables, etc., to the proper storage location.
 - Bring defective equipment to the instructor for repair.
 - Notify the instructor when the stock is about to run out of a certain component.
- In general, keep the following points in mind:
 - Identify lab objectives. An experiment should not be treated as a cookbook procedure. Find a rationale behind each step.

- Come to the lab prepared. Preview the experiment as a homework.
- Keep a lab notebook to record all activities during all lab sessions.
- Finish as much as possible before leaving. This includes acquiring data, interpreting data, answering questions, and revolving uncertainties.
- All data are real. If data look unbelievable, check all the steps carefully. Consult the instructor.
- Safety is first. Change instrument settings slowly. Observe the effect of the most recent change before proceeding with more change. Set voltage/current/power limit. It is important that right from the beginning of your lab work you consider the possible interactions between measuring instruments and the device under test.

4. Specific Outcomes of Instruction (Course Learning Outcomes):

- An Ability to use MDA-Win8086 hardware kit and EMU8086 software emulator to utilize and emulate Intel 8086 Microprocessor. (k).
- An Ability to write assembly language programs for Intel 8086. (b, e, g, k).
- An Ability to evaluate addressing modes for 8086 Intel Microprocessor. (b, e, g).
- An Ability to perform memory & stack access operations for Intel 8086. (b, e, g).
- An Ability to operate Intel 8255A chip to interface 8086 processor with I/O devices such as LEDs, 7SD, DAC, LCD and others. (b, e, g).
- An Ability to recognize the interruption techniques for Intel 8086 Microprocessor. (e)
- An Ability to prepare a small Microprocessors based system such Traffic Light System. (b).

II. Safety Instructions

SAFETY FIRST WHILE IN THE LABORATORY

The basic purpose of laboratory safety is to protect students, researchers, technicians and teachers from the many hazards encountered during the use of various materials and equipments, so safety in the laboratory must be of vital concern to all those engaged in experimental work. It is therefore the responsibility of everyone to adhere strictly to the basic safety precautions provided and to avoid any acts of carelessness that can endanger his life and that of others around him. It is equally important to always abide by all the instructions for conducting the experimental work during the laboratory sessions. A set of information is presented here to safeguard you while in the laboratory.

- No smoking is allowed in the laboratory.
- No food or beverages are allowed in the laboratory.
- Avoid long hair and long sleeved loose clothes and wear lab coat while conducting experiments to minimize the risk of clothing getting caught in the machines.
- Use appropriate personal protective equipment at all time, like Gloves, Safety glasses, Skin Protection, Hearing Protection, and Foot Protection (don't wear open sandals)
- No running, playing, bantering, and kidding in the laboratory.
- Know locations of first aid and all emergency equipments, such as fire alarm, water hoses, fire extinguishers, fire blankets, eyewash stations, and safety showers. Know how to find and use them.

- Use laboratory equipment for its designed purpose.
- Always follow instructions and use only machines and equipment that you are authorized and qualified to operate. If you have any question, consult with your supervisor.
- Know and follow safety rules for specific experiments or tasks.
- Know potential hazards in your work and ways of working safely to prevent such hazards.
- Working alone should be avoided. Someone should always be within call when a laboratory procedure is being performed.
- Avoid mouth contact with any laboratory equipments including pipettes. Use safety filler to fill pipettes.
- Avoid exposure to gases, vapors, and particulates by using a properly functioning laboratory fume hood.
- Use ground fault circuit interrupters where there is a risk of an operator coming in contact with water and electrical equipment simultaneously.
- Follow electrical safety rules and make sure your hands are dry before using electrical equipment, grounding portable electrical tools. Make sure electrical wires are connected properly without short circuit before operating. Wear protective clothing, well-insulated groves and boots, if required.
- Only trained, qualified personnel may repair or modify electrical or any equipment.
- Properly support glass wares using stand, clamps, etc. Also, use proper rings to place round bottom flasks.
- Reduce fire hazard, use shower for fire victims, while fire on clothing, do not run or fan flames, smother flames by wrapping in fire blankets, and spills of flammable solvents can be a source of fire.
- Upon hearing fire alarm, you should evacuate the area and follow emergency procedure.
- Report all injuries including minor scratches, cuts, and burns for First Aid treatment. Corrective actions should be taken to prevent future injuries.
- Report any damage to equipment or instrument and broken glassware to the laboratory instructor as soon as such damage occurs.
- Wash hands upon completion of laboratory procedures and remove all protective equipment including gloves and lab coats.

PROPER CONDUCT IN THE LABORATORY AVOIDS ACCIDENTS.

Safety issues by Eng. Omar Ostah

Week 3: Experiment 1: Analyzing and Verifying the Behavior of Logic Gates Part-1 : AND, OR, NOT(2, 3, 4 Inputs)

<i>Students Names</i>	<i>Student IDs</i>

I. EXPERIMENT BACKGROUND

Keep in mind that computers work on an electrical flow where a high voltage is considered a 1 and a low voltage is considered a 0. Using these highs and lows, data are represented. Electronic circuits must be designed to manipulate these positive and negative pulses into meaningful logic. The digital systems are said to be constructed by using logic gates which can be described by the truth tables. The common logic gates include: inverters, AND, OR, XOR, NAND and NOR gates. A logic gate is an electronic device used to construct logic circuits. All logic gates have inputs and outputs. The state of the output is set by the input states using different rules depending on the gate's type. The different types of gates have different shaped circuit symbols.



Voltages at the inputs can be set to +5v (called 'logic 1' or 'high') or to 0v (called 'logic 0' or 'low'). More detailed description is found in the textbook.

II. MAIN OBJECTIVE

Introduce the basics of circuit wiring, gates behavior for AND, OR, NOT Gates. Familiarize with the use of Digital Circuit pin Diagrams.

Test Standard : IEEE 991-1986

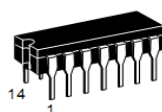
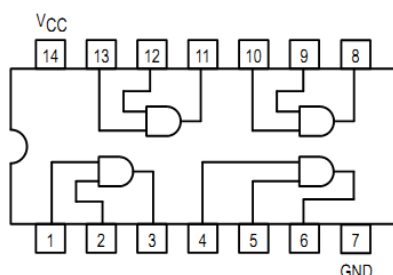
III. EXPERIMENT PROCEDURE

1. Start by reading the experiment introduction and objectives.
2. Connect the Y-0010 Experiment Set with power supply using the provided cable.
3. Prepare the required wires that will be needed to construct the logic circuit.
4. Construct the required digital circuits by connecting the inputs to the input switches provided in the training KIT.
5. Connect the outputs to the output switches LED. Use the LEDs to observe the outputs whether they are 1 (H) or 0 (L) and take notes of them.

6. Apply various combinations of Inputs according to the truth table of the Digital Circuit and observe conditions of LEDs to fill the required tables.
7. Start working with the Experimental work step by step until you finish the required tasks. Also, Solve the required exercises and make your final conclusions about the experiment.
8. Finally, you should write your technical lab report for this experiment and the report will be submitted before the deadline specified by the instructor.

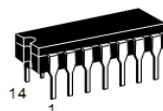
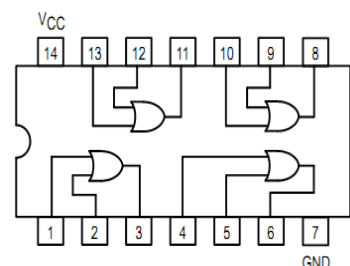
IV. THE REQUIRED DIGITAL CIRCUITS:

1. AND Gate: which is provided into the kit and packed using **74LS08** Integrated Circuit (IC) which contains 4 AND Gates.



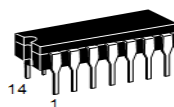
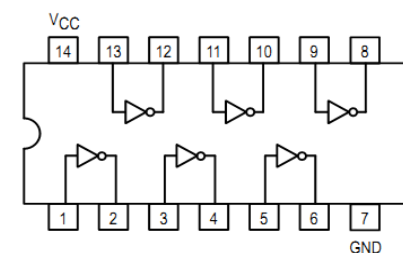
Inputs		Outputs
X	Y	Z= X (AND)Y
0	0	
1	0	
0	1	
1	1	

2. OR Gate: which is provided into the kit and packed using **74LS32** Integrated Circuit (IC) which contains 4 OR Gates.



Inputs		Outputs
X	Y	Z= X (OR)Y
0	0	
1	0	
0	1	
1	1	

3. NOT Gate: which is provided into the kit and packed using **74LS04** Integrated Circuit (IC) which contains 6 NOT Gates. It's also called INVERTER Gate.



Inputs	Outputs
X	Z = Not X
0	
1	

V. EXPERIMENTAL WORK

1. Using the training kit; Construct the AND, OR, Not Circuits and fill the tables for each of them. Then, according to the results you got in the tables, have the truth tables of AND, OR, NOT gates been obtained?
2. From the results you got, what is the arithmetic operations that express AND Gate?
3. From the results you got, what is the arithmetic operations that express OR Gate?
4. From the results you got, what is the arithmetic operations that express NOT Gate?
5. Draw the basic symbol for each of the 2-Input AND, OR, NOT? Also provide the output expression for each gate.
6. Derive the truth table for 3-Input AND Gate, then construct the obtained circuit.

7. Derive the truth table for 3-Input OR Gate, then construct the obtained circuit.

VI. EXPERIMENT EXERCISES

1. Design an AND Gate from the OR-Gate and Invertors. If we want this circuit to work as an OR gate again, what should be added to the circuit? Derive the truth table for both cases.
2. Design an OR Gate from the AND-Gate and Invertors. If we want this circuit to work as an AND gate again, what should be added to the circuit? Derive the truth table for both cases.
3. Derive the truth table for 4-Input AND Gate, then construct the obtained circuit.

Week 4: Experiment 2: Analyzing and Verifying the Behavior of Logic Gates part-2 : NAND, NOR, XOR, XNOR (2, 3, 4 Inputs)

Students Names	Student IDs

I. EXPERIMENT BACKGROUND

A Digital Logic Gate is an electronic device that makes logical decisions based on the different combinations of digital signals present on its inputs. A digital logic gate may have more than one input but only has one digital output. Standard commercially available digital logic gates are available in two basic families or forms, TTL which stands for Transistor-Transistor Logic such as the 7400 series, and CMOS which stands for Complementary Metal-Oxide-Silicon which is the 4000 series of chips. This notation of TTL or CMOS refers to the logic technology used to manufacture the integrated circuit, (IC) or a "chip" as it is more commonly called.



Generally speaking, TTL IC's use NPN (or PNP) type Bipolar Junction Transistors while CMOS IC's use Field Effect Transistors or FET's for both their input and output circuitry. As well as TTL and CMOS technology, simple digital logic gates can also be made by connecting together diodes, transistors and resistors to produce RTL, Resistor-Transistor logic gates, DTL, Diode-Transistor logic gates or ECL, Emitter-Coupled logic gates but these are less common now compared to the popular CMOS family.

Integrated Circuits or IC's as they are more commonly called, can be grouped together into families according to the number of transistors or "gates" that they contain. For example, a simple AND gate may contain only a few individual transistors, whereas a more complex microprocessor may contain many thousands of individual transistor gates. Integrated circuits are categorized according to the number of logic gates or the complexity of the circuits within a single chip with the general classification for the number of individual gates.

More details can be found in the textbook and Instructor's Handouts..

II. MAIN OBJECTIVE

Study and understand the behavior of NAND, NOR, XOR, XNOR Gates.

Introduction of the digital logic design for small logic circuits.

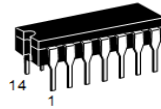
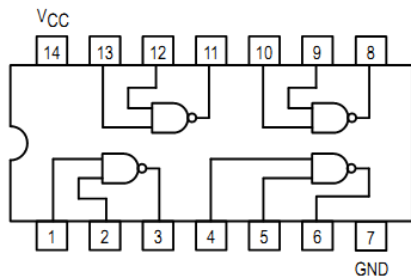
Test Standard : IEEE 991-1986

III. EXPERIMENT PROCEDURE

The procedure for this experiment is the same as the previous one and will be the same for the next two experiments

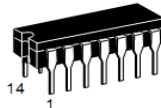
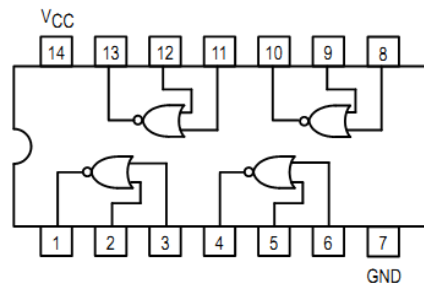
IV. THE REQUIRED DIGITAL CIRCUITS:

1. NAND Gate: which is provided into the kit and packed using 74LS00 Integrated Circuit (IC) which contains 4 NAND Gates.



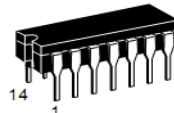
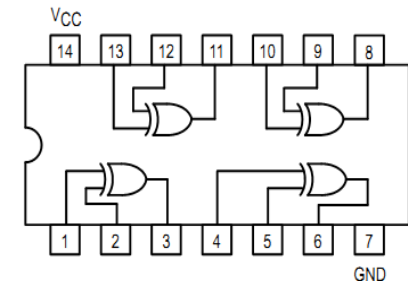
<u>Inputs</u>		<u>Outputs</u>
X	Y	Z= X (NAND) Y
0	0	
1	0	
0	1	
1	1	

2. NOR Gate: which is provided into the kit and packed using 74LS02 Integrated Circuit (IC) which contains 4 NOR Gates.



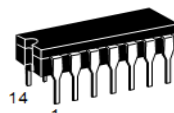
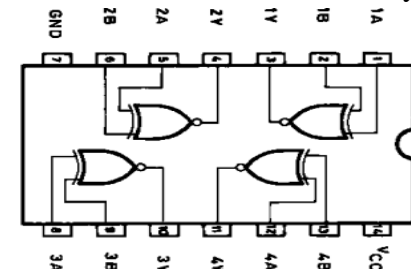
<u>Inputs</u>		<u>Outputs</u>
X	Y	Z= X (NOR) Y
0	0	
1	0	
0	1	
1	1	

3. XOR Gate: which is provided into the kit and packed using 74LS86 Integrated Circuit (IC) which contains 4 XOR Gates. It's also called INVERTER Gate.



<u>Inputs</u>		<u>Outputs</u>
X	Y	Z= X (XOR)Y
0	0	
1	0	
0	1	
1	1	

4. XNOR Gate: which is the 74HC266 that contains 4 XNOR Gates but it's not provided into the kit and can be constructed by using Integrated Circuits (IC) 74LS86 and 74LS04.



<u>Inputs</u>		<u>Outputs</u>
X	Y	Z= X (XNOR)Y
0	0	
1	0	
0	1	
1	1	

V. EXPERIMENTAL WORK

1. Using the training kit; Construct the NAND, NOR, XOR, XNOR Circuits and fill the tables for each of them. Then, according to the results you got in the tables, have the truth tables of NAND, NOR, XOR, XNOR gates been obtained?
2. From the results you got, describe the behavior of NAND Gate?
3. From the results you got, describe the behavior of NOR Gate?
4. From the results you got, describe the behavior of XOR Gate?
5. From the results you got, describe the behavior of XNOR Gate?
6. Using the training kit, what happens if the two Inputs of NAND Gate are connected together?
7. Using the training kit, what happens if the two Inputs of NOR Gate are connected together?
8. Using the training kit, what happens if the two inputs of XOR Gate are connected together?
9. Using the training kit, what happens if the two inputs of XNOR Gate are connected together?
10. Draw the basic symbol for each of the 2-Input NAND, NOR, XOR, XNOR? Also provide the output expression for each gate.
11. Construct the 3-Input NAND Gate Using 2-Input NAND Gates providing the truth table? How many NAND Gates you need? Keep in mind that you are not allowed to use any other type of gates, just NAND.
12. Construct the 3-Input NOR Gate Using 2-Input NOR Gates providing the truth table? How many NOR Gates you need? Keep in mind that you are not allowed to use any other type of gates, just NOR.

VI. EXPERIMENT EXERCISES

1. Construct the 4-Input NAND Gate Using 2-Input NAND Gates providing the truth table? How many NAND Gates you need? Keep in mind that you are not allowed to use any other type of gates, just NAND.
2. Construct the 4-Input NOR Gate Using 2-Input NOR Gates providing the truth table? How many NOR Gates you need? Keep in mind that you are not allowed to use any other type of gates, just NOR.

EXTRA QUESTION: Will Be Graded as a Bonus.

*** Construct the 2-Input XOR Gate Using 2-Input NAND Gates providing the truth table? How many NAND Gates you need? Keep in mind that you are not allowed to use any other type of gates, just NAND.

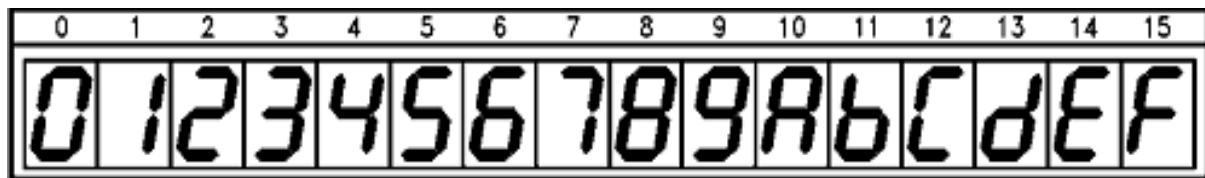
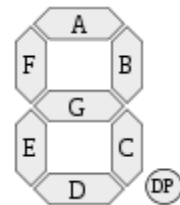
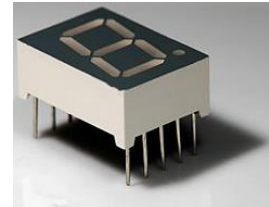
Week 5: Experiment 3: 7-Segment Display and Hexadecimal Driver/Decoder: Having Hex Numbers in SSD and using SSD as BCD/Hex Decoder

I. EXPERIMENT INTRODUCTION.

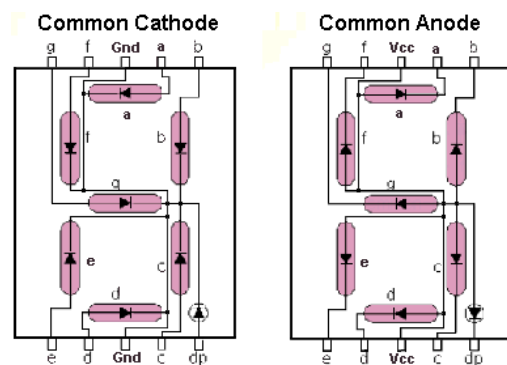
1. Seven-Segment Display (SSD)

Called also seven-segment indicator, it is a form of electronic display device for displaying decimal/Hex numerals that is an alternative to the more complex dot-matrix displays. Seven-segment displays are widely used in digital clocks, electronic meters, and other electronic devices for displaying numerical information.

A seven segment display is composed of seven elements which are LEDs that are small straight bars. According to which LED is excited the display can write numbers from 0 to 9 and letters from A to F (Hex Numbers). There are also fourteen-segment displays and sixteen-segment displays (for full alphanumeric); however, these have mostly been replaced by dot-matrix displays. The segments of a 7-segment display (LEDs) are referred to by the letters A to G, as shown to the right, where the optional DP decimal point (an "eighth segment") is used for the display of non-integer numbers.

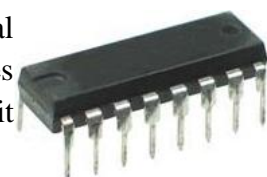


Seven-segment display are used as common anode or common cathode type in different circuits. In common cathode type (as we have in our kit), all the cathodes of the LED are connected together and the anodes are independent. The common cathode is connected to the ground. LEDs to whose anode positive voltage is given will be lighting and others are not. Current limiting resistors should be connected to each anode. For example, if a, b, c, d, g pins are connected to positive voltage through the resistors, in the screen number 3 will be seen.

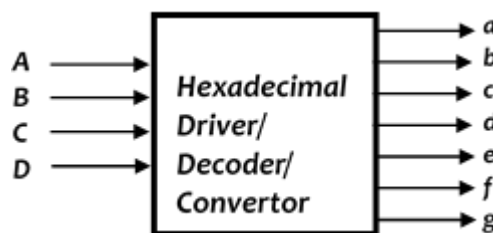


2. Hexadecimal Decoder (Driver):

The 7 segment display can show a 4-bit Binary data as a Hexadecimal number with an appropriate Hex Decoder/ Driver which is an IC that takes the 4-bit binary data (BCD form) and the converts it to a suitable 7-bit code that the seven segment display can use.



For Example; in order to see the number 1 on the display, 0001 is applied to the input of the IC. The output of the Decoder/Converter will be 0000110 (g-f-e-d-c-b-a). Thus, as b and c LEDs have positive voltage, they will be lighting and 1 will be seen on the display.



3. Light-Emitting Diode (LED)

It is a semiconductor light source. LEDs are used as indicator lamps in many devices and are increasingly used for other lighting. Introduced as a practical electronic component in 1962, early LEDs emitted low-intensity red light, but modern versions are available across the visible, ultraviolet and infrared wavelengths, with very high brightness. Light-emitting diodes are used in applications as diverse as replacements for aviation lighting, automotive lighting (particularly brake lamps, turn signals and indicators) as well as in traffic signals. The advantages of LEDs mentioned above have allowed new text and video displays and sensors to be developed, while their high switching rates are also useful in advanced communications technology. Infrared LEDs are also used in the remote control units of many commercial products including televisions, DVD players, and other domestic appliances.



II. MAIN OBJECTIVES.

- Study and understand the design of Seven Segment display and learn how to use it.
- Analyze the usage of seven segment display with Hexadecimal Decoder.
- **Test Standard : IEEE 991-1986**

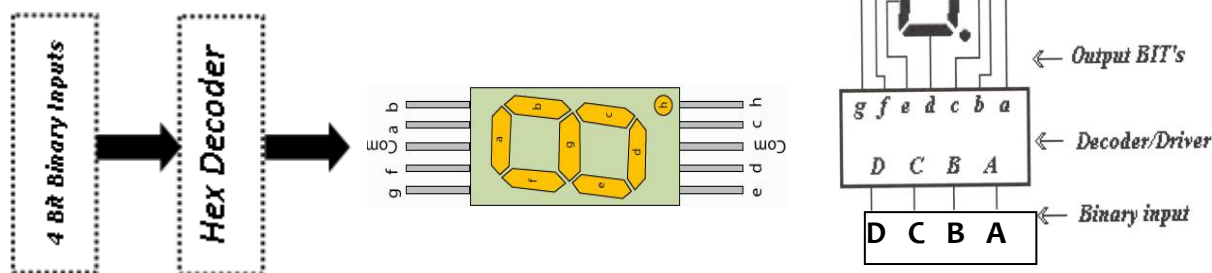
III. EXPERIMENT PROCEDURE.

The procedure for this experiment is the same as the previous one

IV. THE REQUIRED DIGITAL CIRCUITS:

1. Having Hexadecimal Numbers in 7 Segment Display:

A single byte can encode the full state of a 7-segment-display. The most popular bit encodings are gfedcba and abcdefg - both usually assume 0 is off and 1 is on.

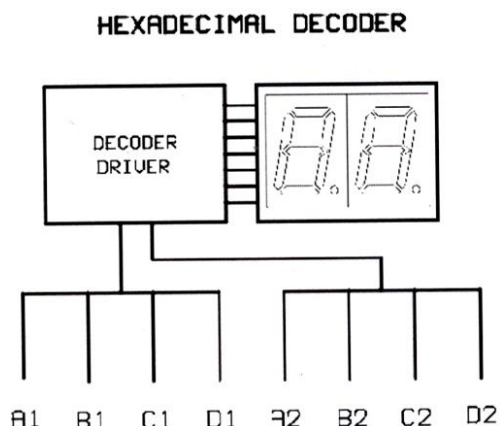


Binary Input	Hex Digit	abcdefg	gfedcba	a	b	c	d	e	f	g
	0	0x7E	0x3F	on	on	on	on	on	on	off
	1									
	2									
	3									
	4									
	5									
	6									
	7									
	8									
	9									
	A									
	b									
	C									
	d									
	E									
	F									

2. Having 2-Digits Hexadecimal Number in 7 Segment Display:

Construct the following circuit which utilize the 2-Seven segment displays and Hexadecimal code Driver IC.

S T E P	Decimal Number	Binary Number				Hexadecimal	
		MSB		LSB		MSB	LSB
		D	C	B	A	(LEFT)	(RIGHT)
1	1						
2		0000		1000			
3	5						
4	3						
5	18						



6			4	6
7	120			
8	105			
9	75			
10		1110	A	
11	191			
12	205	1100		D

V. EXPERIMENTAL WORK

1. Using the training kit; Construct the required Circuits and fill the tables for each of them. Then, according to the results you got in the tables, have the **decimal and hexadecimal numbers** been obtained on the **7-segment display**?
2. From the results you got; describe the behavior of 7-Segment Display when we provide it by the segment code (01111012)?
3. From the results you got; describe the behavior of LED (E) if the BCD input 10012?
4. From the results you got; describe the behavior of Hex Decoder if BCD Input 1100 is used? How bits in its output.
5. How many resistors are used to display the decimal number (4.310) on the 7-segment display? what are the values of these resistors? What are the diodes that will be involved in this (assume common cathode is used).

VI. EXPERIMENT EXERCISES

1. If we want to display the number (0xF0D2) on the 7-segment display.
 - How many bytes are needed for the **BCD Inputs**?
 - How many bytes are needed for the **Hex Decoder Output**?
 - How many bytes are needed for the **Segment Code** of SSD?
 - How many **7-segment displays** are needed?

- What will be **BCD Input** for the decoders?
 - What will be **Segment code** for each 7-segment display (Com. Cathode) ?
 - How many **LEDs** that will be in the **ON** state for this number ?
 - Draw the final design for this circuit
2. Give two real examples of using 7-segment display

Week 6: Experiment 4: Digital Logic Circuit Modeling & Simulation with Multisim: Introduction and tutorial training of Multisim For Logic Design- TTL Family)

I. EXPERIMENT INTRODUCTION.

Multisim is a schematic capture and simulation program for analog, digital and mixed analog/digital circuits, and is one application program of the National Instruments “Circuit Design Suite”, which also includes printed circuit board design tools and an interface to the ELVIS bread boarding platform.

It is suggested that you begin by taking a few minutes to read the short Multisim tutorial, which is presented as Chapter 2 of “Getting Started with NI Circuit Design Suite”, which can be accessed via:

Start Menu > All Programs > National Instruments > Circuit Design Suite 11.0 > Documentation > Getting Started

The basic steps in modeling and analysis of a digital logic circuit are:

1. Open Multisim and create a “design”.
2. Draw a schematic diagram of the circuit (components and interconnections).
3. Define digital test patterns to be applied to the circuit inputs to stimulate the circuit and connect signal sources to the inputs to produce these patterns.
4. Connect the circuit outputs to one or more indicators to display the response of the circuit to the test patterns.
5. Run the simulation and examine the results, copying and pasting Multisim windows into lab reports and other documents as needed.
6. Save the design.

II. MAIN OBJECTIVE.

- Understand the simulation tools for logic design and to verify several elements using Multisim.
- Familiarize with software modeling of logic circuits before building it by hardware.
- **Test Standard : IEEE 991-1986**

III. EXPERIMENT PROCEDURE.

- Start by reading the experiment introduction, tutorials and objectives.
- Open Multisim, create a new project (design) and save it on desktop.
- Prepare the required wires, elements and units that are needed to construct the circuit.
- Construct the required digital circuit by connecting the inputs to the appropriate input suppliers provided in Multisim.
- Connect the outputs to the appropriate output elements. You may use LEDs to observe the outputs whether they are 1 (H) or 0 (L) and take notes of them.
- Apply various combinations of Inputs of the Digital Circuit and observe conditions of outputs to provide/fill the required truth tables.
- Take a screenshot of the designed circuit and include it with your report.
- Save your work under the name given in the required circuit. You should save your work in the desktop inside a folder that named by your group number.
- Solve the required experiment exercises and make your final conclusions about the experiment.

IV. THE REQUIRED DIGITAL CIRCUITS:

Circuit # 1: Design a circuit into MultiSIM which implements the function $D=A \cdot B+C$ using a 2-input AND gate, a 2-input OR gate, and an LED. This will appear as an AND gate with two switches labeled A and B as inputs, an OR gate with one switch labeled C as one input and another from the output of the AND gate, and an LED labeled D as output. Test your circuit by checking the output D for every combination of A, B and C. Doing so should produce the truth table of the circuit. Provide the truth table.

Circuit # 2: Design a circuit into MultiSIM which implements the function for $Z=W \cdot (X+Y)$ using an appropriate gates and elements (you can repeat what you did in circuit 1). Provide the truth table.

V. EXPERIMENTAL WORK

1. Do you understand the basics of Logic design using Multisim? What are the electronic families you worked with in this experiment?
2. Give 2 examples of sources that can be used to provide an input for the digital circuit?
3. Give 2 examples of indicators that can be used to provide an output for the digital circuit?

4. In what other areas can we use Multisim (we mean other than Digital logic Design)?

VI. EXPERIMENT EXERCISES

1. Simulate the design of the function $P = (A+B) \cdot (A'+C+D) \cdot (B'+D')$ with MultiSIM using the appropriate 74LS series chips. Also, Generate the truth table and timing diagram of output versus inputs.

2. Enter a new circuit into MultiSIM which implements the function $Q = (A \cdot B) + (A' \cdot C \cdot D) + (B' \cdot D')$ using standard 74LS series components rather than logic gates. Also, Generate the truth table and timing diagram of output versus inputs.

3. Enter a new circuit into MultiSIM which implements the function $E = A \cdot B + C \cdot D$ using standard 74LS series components rather than logic gates. Also, Generate the truth table and timing diagram of output versus inputs.

Week 7: Experiment 5: Building and Designing a logic circuits Using Multisim: Adder & Comparator

Students Names	Student IDs

I. EXPERIMENT INTRODUCTION

Addition is just what you would expect in computers. Digits are added bit by bit from right to left, with carries passed to the next digit to the left, just as you would do by hand. Subtraction uses addition: the appropriate operand is simply negated before being added.

1. Binary Parallel Adder (74LS83D):

An n-bit adder is a circuit which adds two n-bits numbers, say, A and B. In addition, an n-bit adder will have another single-bit input which is added to the two numbers called the carry-in (C_{in}). The output of the n-bit adder is an n-bit sum (S) and a carry-out (C_{out}) bit. The block diagram of the n-bit adder is shown. If all input bits of the two numbers (A & B) are applied simultaneously in parallel, the adder is termed a Parallel Adder.

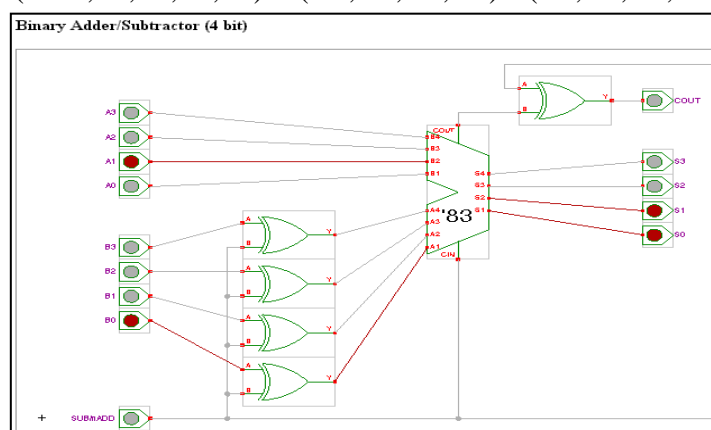
A four-bit adder/Subtractor demonstration. While it is perfectly possible to design a custom circuit for the subtraction operation, it is much more common to re-use an existing adder and to replace a subtraction by a two-complement's addition.

The applet shows how this is done. Click the input switches or type the 's' bind key to control the Sub/nAdd switch, and '1' .. '8' to control the A and B inputs. When the Sub/nAdd input is low (0), the XOR-gates act as non-inverting buffers, and the carry-input to the adder is 0. Therefore, the adder calculates a four-bit sum plus carry-out:

$$(C_{out}, S_3, S_2, S_1, S_0) = (A_3, A_2, A_1, A_0) + (B_3, B_2, B_1, B_0)$$

If the Sub/nAdd input is high (1), the XOR-gates act as inverting buffers, and the carry-input to the adder is 1.

$$(C_{out}, S_3, S_2, S_1, S_0) = (A_3, A_2, A_1, A_0) - (B_3, B_2, B_1, B_0)$$

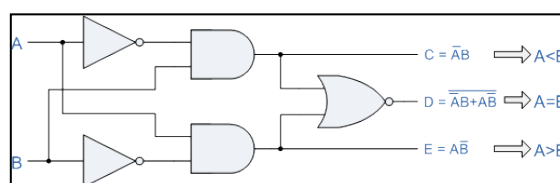


2. Digital Magnitude Comparator:

It is a type of digital comparator that has three output terminals, one each for equality, $A = B$ greater than, $A > B$ and less than $A < B$. The purpose of a **Digital Comparator** is to compare a set of variables or unknown numbers, for example A ($A_1, A_2, A_3, \dots A_n$, etc) against that of a constant or unknown value such as B ($B_1, B_2, B_3, \dots B_n$, etc) and produce an output condition or flag depending upon the result of the comparison.

For example, a magnitude comparator of two 1-bits, (A and B) inputs would produce the following three output conditions when compared to each other. $A > B$, $A = B$, $A < B$. Which means: A is greater than B, A is equal to B, and A is less than A.

This is useful if we want to compare two variables and want to produce an output when any of the above three conditions are achieved. For example, produce an output from a counter when a certain count number is reached. Consider the simple 1-bit comparator below.



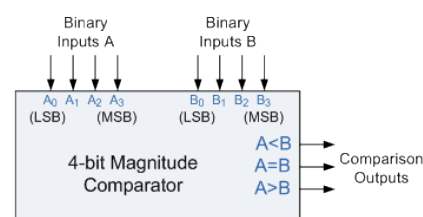
Then the operation of a 1-bit digital comparator is given in the following Truth Table:

Inputs		Outputs		
B	A	$A > B$	$A = B$	$A < B$
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

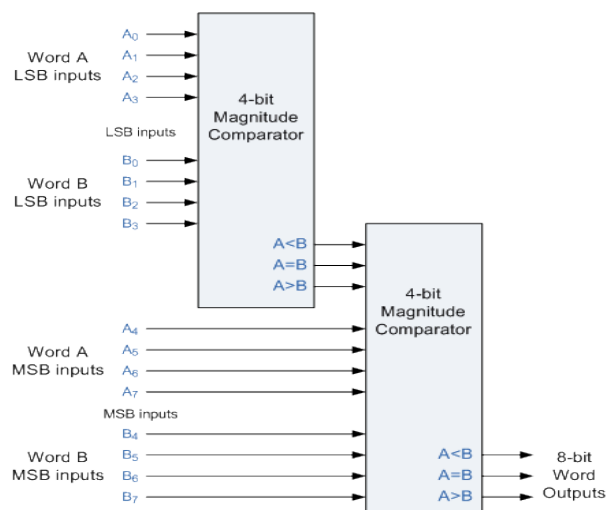
You may notice two distinct features about the comparator from the above truth table. Firstly, the circuit does not distinguish between either two "0" or two "1"s as an output $A = B$ is produced when they are both equal, either $A = B = "0"$ or $A = B = "1"$. Secondly, the output condition for $A = B$ resembles that of a commonly available logic gate, the Exclusive-NOR or Ex-NOR function (equivalence) on each of the n-bits giving: $Q = A \oplus B$

Digital comparators actually use Exclusive-NOR gates within their design for comparing their respective pairs of bits. When we are comparing two binary or BCD values or variables against each other, we are comparing the "magnitude" of these values, a logic "0" against a logic "1" which is where the term **Magnitude Comparator** comes from.

As well as comparing individual bits, we can design larger bit comparators by cascading together n of these and produce a n-bit comparator just as we did for the n-bit adder in the previous tutorial. Multi-bit comparators can be constructed to compare whole binary or BCD words to produce an output if one word is larger, equal to or less than the other. A very good example of this is the 4-bit **Magnitude Comparator**. Here, two 4-bit words ("nibbles") are compared to each other to produce the relevant output with one word connected to inputs A and the other to be compared against connected to input B as shown here:



Some commercially available digital comparators such as the TTL 7485 or CMOS 4063 4-bit magnitude comparator have additional input terminals that allow more individual comparators to be "cascaded" together to compare words larger than 4-bits with magnitude comparators of "n"-bits being produced. These cascading inputs are connected directly to the corresponding outputs of the previous comparator as shown to compare 8, 16 or even 32-bit words.



II. MAIN OBJECTIVE

- Understand the simulation tools for logic design and verify several elements using Multisim
- Understand the concept of Half and Full Adders and Introduce two important MSI circuits: 4-bit adder and 4-bit magnitude comparator.

● **Test Standard : IEEE 991-1986**

III. EXPERIMENT PROCEDURE

The procedure for this experiment is the same as the previous one

IV. THE REQUIRED DIGITAL CIRCUITS:

Circuit # 1: Use Multisim to Analyze the operation of the IC 74LS85 (4-bit Magnitude comparator), by connecting its inputs (A3-A0), (B3-B0) and the inputs (A>B, A<B, A=B) to switches and outputs (A>B, A<B, A=B) to the LEDs and then fill Table below.

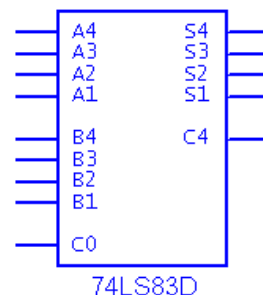
A ₀	in
A ₁	A > B
A ₂	A < B
A ₃	A = B
74LS85 out	
B ₀	A > B
B ₁	A < B
B ₂	A = B
B ₃	

Inputs								Outputs		
Input A				Input B				A>B	A<B	A=B
A3	A2	A1	A0	B3	B2	B1	B0			
0	0	0	1	0	0	1	1			
0	1	1	1	0	1	0	0			
0	1	0	1	0	1	0	1			
0	1	0	1	0	1	0	1			
1	0	1	1	1	0	1	1			
1	0	0	1	1	1	0	0			
1	1	0	0	1	0	1	0			

1	1	1	1	1	1	1	1			
---	---	---	---	---	---	---	---	--	--	--

Circuit # 2: Design a 4-bit parallel adder/subtractor circuit on Multisim using the following two ICs 74LS83 (4-bit parallel adder) and 74LS86 (2-input XOR). The IC 74LS83 has two four bit number input (A4-A1), (B4-B1) and one control signal (C0) and four output number (S4-S1) result and C4(Cout) as shown in Figure.

- When $C_0 = 0$ the circuit works as **Adder**.
- When $C_0 = 1$ the circuit works as **Subtractor**.



Include a screenshot of your design and provide the truth table.

V. EXPERIMENT EXERCISES.

1. Use Multisim to expand the 4-bit Magnitude Comparator to get 8 bit Magnitude Comparator and test it.

Include a screenshot of your design and provide the truth table.

2. Implement a half adder using an AND & XOR gates on Multisim.

Include a screenshot of your design and provide the truth table.

Week 8: Experiment 6: Logic Circuit Design Using K-Maps and Boolean Algebra: 3-bits Full Adder and 2-bits Square Taking

Students Names	Student IDs

I. EXPERIMENT INTRODUCTION

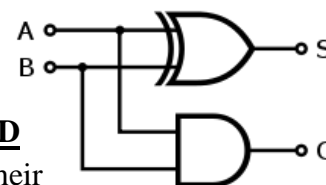
1. Adder

In electronics, an adder or summer is a **digital circuit** that performs **addition** of numbers. In many **computers** and other kinds of processors, adders are used not only in the **arithmetic logic unit(s)**, but also in other parts of the processor, where they are used to calculate addresses, table indices, and similar.

Although adders can be constructed for many numerical representations, such as **binary-coded decimal** or **excess-3**, the most common adders operate on **binary** numbers. In cases where **two's complement** or **ones' complement** is being used to represent negative numbers, it is trivial to modify an adder into an **adder-subtractor**. Other **signed number representations** require a more complex adder.

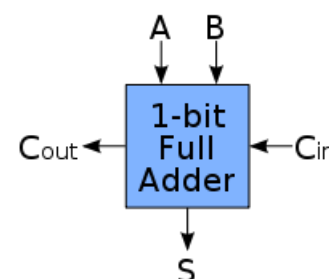
2. Half Adder

A half adder adds two one-bit binary numbers A and B. It has two outputs, S and C (the value theoretically carried on to the next addition); the final sum is $2C + S$. The simplest half-adder design, pictured on the right, incorporates an **XOR gate** for S and an **AND gate** for C. Half adders cannot be used compositely, given their incapacity for a carry-in bit.



3. Full Adder (74LS83)

A full adder adds binary numbers and accounts for values carried in as well as out. A one-bit full adder adds three one-bit numbers, often written as A, B, and C_{in} ; A and B are the operands, and C_{in} is a bit carried in (in theory from a past addition). The circuit produces a two-bit output sum typically represented by the signals C_{out} and S, where $sum = 2 \times C_{out} + S$.



A full adder can be implemented in many different ways such as with a custom **transistor**-level circuit or composed of other gates. One example implementation is with $S = A \oplus B \oplus C_{in}$ and $C_{out} = (A \cdot B) + (C_{in} \cdot (A \oplus B))$.

In this implementation, the final **OR gate** before the carry-out output may be replaced by an **XOR gate** without altering the resulting logic. Using only two types of gates is convenient

if the circuit is being implemented using simple IC chips which contain only one gate type per chip. In this light, C_{out} can be implemented as $C_{out} = (A \cdot B) \oplus (C_{in} \cdot (A \oplus B))$.

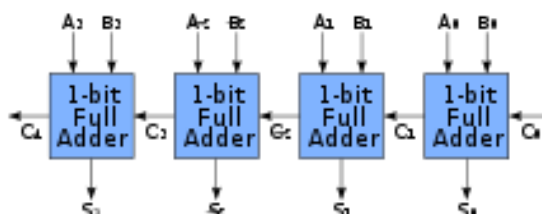
A full adder can be constructed from two half adders by connecting A and B to the input of one half adder, connecting the sum from that to an input to the second adder, connecting C_i to the other input and OR the two carry outputs. Equivalently, S could be made the three-bit XOR of A, B, and C_i , and C_o could be made the three-bit **majority function** of A, B, and C_i .

4. More Complex Adders

a. Carry Ripple Adder:

It is possible to create a logical circuit using multiple full adders to add N-bit numbers. Each full adder inputs a C_{in} , which is the C_{out} of the previous adder. This kind of adder is a ripple carry adder, since each carry bit "ripples" to the next full adder. Note that the first (and only the first) full adder may be replaced by a half adder.

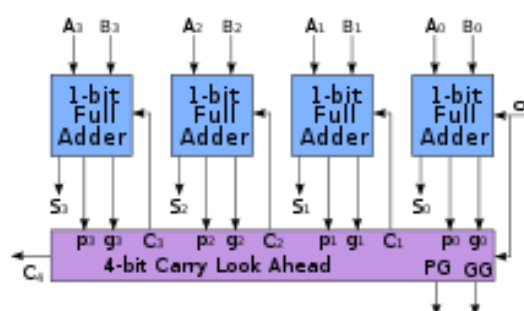
The layout of a ripple carry adder is simple, which allows for fast design time; however, the ripple carry adder is relatively slow, since each full adder must wait for the carry bit to be calculated from the previous full adder. The **gate delay** can easily be calculated by inspection of the full adder circuit. Each full adder requires three levels of logic. In a 32-bit [ripple carry] adder, there are 32 full adders, so the critical path (worst case) delay is 3 (for carry propagation in first adder) + $31 \cdot 2$ (for carry propagation in later adders) = 65 gate delays.



b. Carry-lookahead adders:

To reduce the computation time, engineers devised faster ways to add two binary numbers by using **carry-lookahead adders**. They work by creating two signals (P and G) for each bit position, based on if a carry is propagated through from a less significant bit position (at least one input is a '1'), a carry is generated in that bit position (both inputs are '1'), or if a carry is killed in that bit position (both inputs are '0'). In most cases, P is simply the sum output of a half-adder and G is the carry output of the same adder. After P and G are generated the carries for every bit position are created. Some advanced carry-lookahead architectures are the **Manchester carry chain**, **Brent-Kung adder**, and the **Kogge-Stone adder**.

Some other multi-bit adder architectures break the adder into blocks. It is possible to vary the length of these blocks based on the **propagation delay** of the circuits to optimize computation time. These block based adders include the **carry bypass adder** which will determine P and G values for each block rather than each bit, and the **carry select adder** which

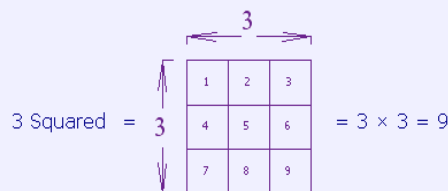


pre-generates sum and carry values for either possible carry input to the block.

c. Square Taking

To square a number, just multiply it by itself.

Example: What is 3 squared?



"Squared" is often written as a little 2 like this:

this means "squared"

$$4^2 = 16$$

This says "4 Squared equals 16"

(the little 2 says the number appears twice in multiplying)

You can also square negative numbers. When you square a negative number you get a positive result. Just the same as if you had squared a positive number:

$$\begin{array}{l} 5 \times 5 = 25 \\ -5 \times -5 = 25 \end{array} \quad \left. \vphantom{\begin{array}{l} 5 \times 5 = 25 \\ -5 \times -5 = 25 \end{array}} \right\} \text{same answer!}$$

d. Other adder designs

carry-save adder, carry-select adder, conditional-sum adder, carry-skip adder, and carry-complete adder.

II. MAIN OBJECTIVES

Build and design the combinational circuits of full Adder and Square Taking by using the K-Maps to get the Minimal Sum of Products (MSP).

Test Standard : IEEE 991-1986

III. EXPERIMENT PROCEDURE

The procedure for this experiment is the same as lab1

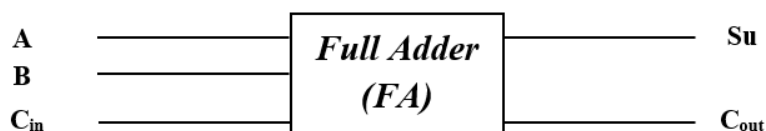
IV. THE REQUIRED DIGITAL CIRCUITS

1. Full Adder Circuit:

A full Adder sums up two 3-bit numbers. Following is the symbol of full adder.

Inputs			Outputs	
A	B	C _{in}	Su	C _{out}

Beside the symbol is the truth table and you have to fill it.

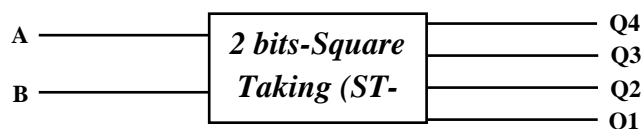


2. Square Taking Circuit:

Here, we will design a circuit

that takes the square of 2-bit numbers. Following is the symbol of full adder.

Beside the symbol is the truth table and you have to fill it.



Inputs		Outputs			
A	B	Q ₄	Q ₃	Q ₂	Q ₁

V. EXPERIMENTAL WORK

1. For Full Adder Circuit.

1. Use K-Maps to simplify both outputs Sum and C_{out} .
2. Draw the gate Representation of both output functions after K-Maps Simplification. (Build it using the KIT then try use all possibilities)

2. For Square Taking Circuit

1. Use K-Map to simplify outputs Q_4 , Q_3 , Q_2 and Q_1 .
2. Draw the gate Representation of both output functions after K-Maps Simplification. (Build it using the KIT then try use all possibilities)

VI. EXPERIMENT EXERCISES

1. Simplify the Q outputs given below in Sum Of Products (SOP) from Using Karnaugh Map and design the circuit using Multisim. (Use only 2-Input Gates)

Inputs			Outputs
A	B	C	Q
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

2. Make the simplification of the Q outputs of the circuits selecting the even numbers of a counter by using a Karnaugh Maps in Sum Of Products form. Also, Design the circuit. ("0" is considered to be even)

<i>Inputs</i>				<i>Outputs</i>
A	B	C	D	Q
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Week 9: Experiment 7: Design and Conduct an Experiment for a simple logic circuit that calculate the Absolute of a Complex Number X: Design and verify a small mathematical system with using digital logic components.

Students Names	Student IDs

Test Standard : IEEE 991-1986

EXPERIMENT DESCRIPTION

You are asked to design an experiment for a simple logic circuit that calculate the Absolute of a Complex Number X. The Complex Number X consist of 2-bit real number A and 2-bit imaginary number B. You need to create the truth table of the circuit and use Karnaugh Map simplification to simplify and determine the logic circuit for each of the outputs.

- A complex number X can be represented as $X = A + jB$. Where A is the real number and B is the imaginary number. $X = 3 + j3$ would mean that X consist of real number $A = 3$ and imaginary number $B = 3$.
- Mathematically, the absolute of a complex number $X = A + jB$ is represented as $Y = \|X\|^2 = A^2 + B^2$.
- Please note:- For a 2-input X, the maximum value of input would be $X = A + jB = 3 + j3$. The output $Y = \|X\| = 9 + 9 = 18$ would require a minimum of 5 bits to represent the entire range of output.
- Create the truth table for the mathematical model above. The input range would be 3-bits for A and 3 bits for B and output Y would be 5 bits.
- Simplify the output Y using Karnaugh Map simplification and build it using standard logics such as AND, OR, Inverters, NAND, NOR. Build and verify the circuit in the lab using the Hardware Kits and The Multisim software.

You need to design the experiment, implement and test it using Digital Kits available in the lab as well as the Multisim software.

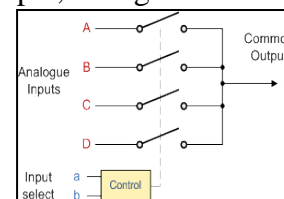
Weeks 10 & 11 :Experiment 8: Combinational logic Design and Analysis: Multiplexers (MUX) and De-multiplexers (DMUX) Units

Students Names	Student IDs

I. EXPERIMENT INTRODUCTION.

Multiplexer (74LS151)

It is a data selector, more commonly called a Multiplexer, shortened to "Mux" or "MPX. It is a combinational logic switching device that operates like a very fast acting multiple position rotary switch. They connect or control, multiple input lines called "channels" consisting of either 2, 4, 8 or 16 individual inputs, one at a time to an output. Then the job of a multiplexer is to allow multiple signals to share a single common output. For example, a single 8-channel multiplexer would connect one of its eight inputs to the single data output. Multiplexers are used as one method of reducing the number of logic gates required in a circuit or when a single data line is required to carry two or more different digital signals. Digital Multiplexers are constructed from individual **analogue switches** encased in a single IC package as opposed to the "mechanical" type selectors such as normal conventional switches and relays. Generally, multiplexers have an even number of data inputs, usually an even power of two, n^2 , a number of "control" inputs that correspond with the number of data inputs and according to the binary condition of these control inputs, the appropriate data input is connected directly to the output. An example of a Mux configuration is 4-to-1 Channel Multiplexer.

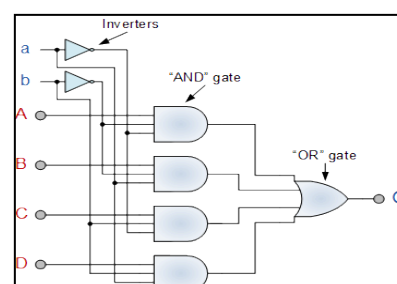


Addressing		Input Selected
b	a	
0	0	A
0	1	B
1	0	C
1	1	D

The Boolean expression for this 4-to-1 Multiplexer above with inputs A to D and data select lines a, b is given as:

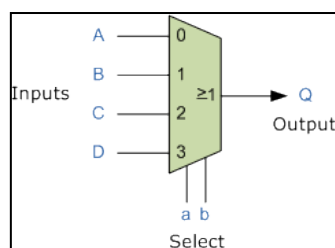
$$Q = abA + abB + abC + abD$$

In this example at any one instant in time only ONE of the four analogue switches is closed, connecting only one of the input lines A to D to the single output at Q. As to which switch is closed depends upon the addressing input code on lines "a" and "b", so for this example to select input B to the output at Q, the binary input address would need to be "a" = logic "1" and "b" = logic "0". Adding more control address lines will allow the multiplexer to control more inputs but each control line configuration will connect only ONE input to the output. Then the implementation of this Boolean expression above using individual logic gates would

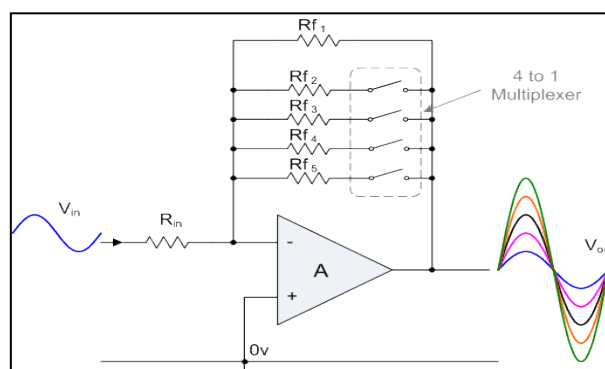


require the use of seven individual gates consisting of AND, OR and NOT gates as shown (4 Channel Multiplexer using Logic Gates).

The symbol used in logic diagrams to identify a multiplexer is as follows.



As well as sending parallel data in a serial format down a single transmission line or connection, another possible use of multi-channel multiplexers is in digital audio applications as mixers or where the gain of an analogue amplifier can be controlled digitally, for example (Digitally Adjustable Amplifier Gain).



Here, the voltage gain of the inverting amplifier depends on the ratio between the input resistor, R_{in} and its feedback resistor, R_f as determined in the **Op-amp** tutorials. A single 4-channel (Quad) SPST switch configured as a 4-to-1 channel multiplexer is connected in series with the resistors to select any feedback resistor to vary the value of R_f . The combination of these resistors will determine the overall gain of the amplifier, (A_v). Then the gain of the amplifier can be adjusted digitally by simply selecting the appropriate resistor combination.

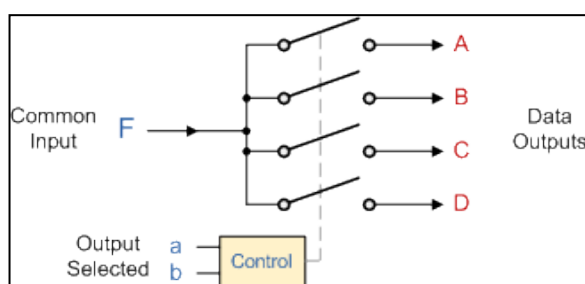
Digital multiplexers are sometimes also referred to as "Data Selectors" as they select the data to be sent to the output line and are commonly used in communications or high speed network switching circuits such as LAN's and Ethernet applications. Some multiplexer IC's have a single inverting buffer (NOT Gate) connected to the output to give a positive logic output (logic "1", HIGH) on one terminal and a complimentary negative logic output (logic "0", LOW) on another different terminal.

It is possible to make simple multiplexer circuits from standard **AND** and **OR** gates as we have seen above, but commonly multiplexers/data selectors are available as standard i.c. packages such as the common TTL 74LS151 8-input to 1 line multiplexer or the TTL 74LS153 Dual 4-input to 1 line multiplexer. Multiplexer circuits with much higher number of inputs can be obtained by cascading together two or more smaller devices. The Multiplexer is a very useful combinational device that has its uses in many different applications such as signal routing, data communications and data bus control. When used with a demultiplexer, parallel data can be transmitted in serial form via a single data link such as a fiber-optic cable or telephone line.

They can also be used to switch either analogue, digital or video signals, with the switching current in analogue power circuits limited to below 10mA to 20mA per channel in order to reduce heat dissipation.

2. Demultiplexer (addressable Latch 74LS259)

The data distributor, known more commonly as a Demultiplexer or "Demux", is the exact opposite of the **Multiplexer** we saw in the previous tutorial. The demultiplexer takes one single input data line and then switches it to any one of a number of individual output lines one at a time. The demultiplexer converts a serial data signal at the input to a parallel data at its output lines (1-to-4 Channel De-multiplexer) as shown below.

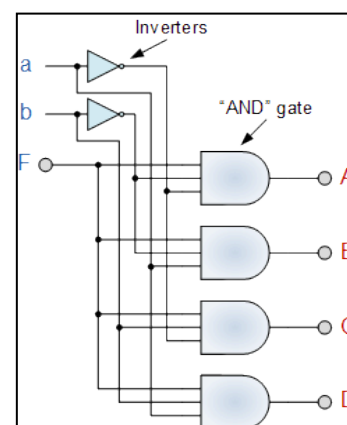


Addressing		Input Selected
b	a	
0	0	A
0	1	B
1	0	C
1	1	D

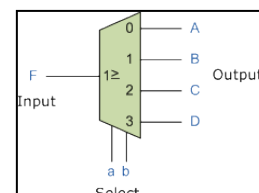
The Boolean expression for this 1-to-4 Demultiplexer above with outputs A to D and data select lines a, b is given as:

$$F = ab A + abB + abC + abD$$

The function of the Demultiplexer is to switch one common data input line to any one of the 4 output data lines A to D in our example above. As with the multiplexer the individual solid state switches are selected by the binary input address code on the output select pins "a" and "b" and by adding more address line inputs it is possible to switch more outputs giving a 1-to-2ⁿ data line outputs. Some standard demultiplexer IC's also have an "enable output" input pin which disables or prevents the input from being passed to the selected output. Also some have latches built into their outputs to maintain the output logic level after the address inputs have been changed. However, in standard decoder type circuits the address input will determine which single data output will have the same value as the data input with all other data outputs having the value of logic "0".



The implementation of the Boolean expression above using individual logic gates would require the use of six individual gates consisting of AND and NOT gates as shown (4 Channel Demultiplexer using Logic Gates). The symbol used in logic diagrams to identify a demultiplexer is as follows.



Standard Demultiplexer IC packages available are the TTL 74LS138 1 to 8-output demultiplexer, the TTL 74LS139 Dual 1-to-4 output demultiplexer or the CMOS CD4514 1-to-16 output demultiplexer. Another type of demultiplexer is the 24-pin, 74LS154 which is a 4-bit to 16-line demultiplexer/decoder. Here the individual output positions are selected

using a 4-bit binary coded input. Like multiplexers, demultiplexers can also be cascaded together to form higher order demultiplexers.

Unlike multiplexers which convert data from a single data line to multiple lines and demultiplexers which convert multiple lines to a single data line, there are devices available which convert data to and from multiple lines and in the next tutorial about combinational logic devices, we will look at **Encoders** which convert multiple input lines into multiple output lines, converting the data from one form to another.

II. MAIN OBJECTIVES.

- Analyze Multiplexer and Demultiplexer circuits by observing their operations and obtaining their truth tables.
- Familiarize with 74LS151 Mux IC and with 74LS259 Addressable Latch Demux IC .
- Test Standard : IEEE 991-1986**

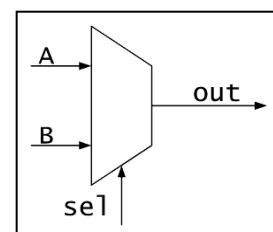
III. EXPERIMENT PROCEDURE.

The procedure for this experiment is the same as lab1

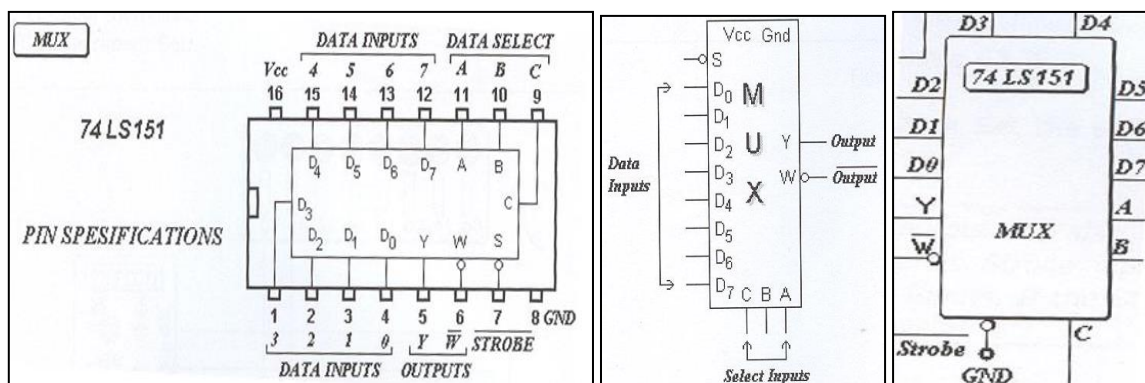
IV. THE REQUIRED DIGITAL CIRCUITS:

1. Multiplexer Circuit (74LS151):

Let's say that we have 2 inputs to a given circuit, and we only have one output. We would like the two inputs to "share" the output, and we therefore need to switch from one input to the other. We have inputs I1 and I2, output O, and a control wire, C. We can say that if C=1, then we connect I1 to O. However, if C=0, then we connect I2 to O. This is a multiplexer.

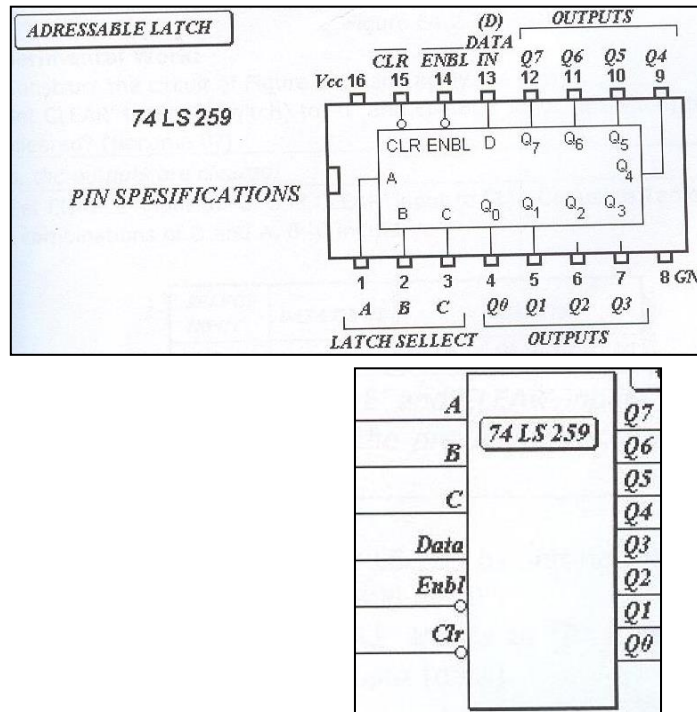
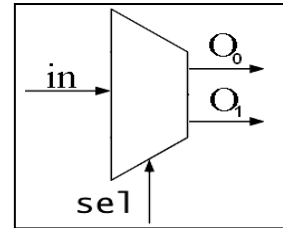


In this part; you will test and analyze the behavior of Mux by wiring the following IC appropriately:



2. 8-Bit Addressable Latch VE Demultiplexer Circuit (74LS259)

we can simply call it Demux. let's say we have a circuit with a single input (I), two outputs (O1 and O2), and a control wire, C. When C=1, we connect I to O1, and when C=0, we connect I to O2. This is a demultiplexer. In this part; you will test and analyze the behavior of Demux by wiring the following IC appropriately:



V. EXPERIMENTAL WORK

1. Using the training kits; Construct the required Circuits for Mux and Demux. Make sure you connect all inputs, outputs and GNDs pins appropriately.
2. What's the purpose of STROPE input in the 74LS151 Mux?
3. For the MUX circuit, set the selection switches (C,B,A) to level "0". What is the Y and W output? Explain your answer.
4. For the MUX circuit, set the data input switches (D3, D4, D5) to level "1" and others to level "0". Also, set the selection switches (C,B,A) to "100". What is the outputs? Explain your answer.
5. For the MUX circuit, set strobe input to "1" (Connect to Vcc). Set the data switches (D0, D2, D4, D6) to "1" and the selection switches (C,B,A) to "110". What is the result? Explain your answer.
6. For the Mux circuit, try several combinations and fill the following table:

Strobe	Data Inputs D ₀ D ₁ D ₂ D ₃ D ₄ D ₅ D ₆ D ₇	Selection Lines			outputs		
		C	B	A	Data Out	Y	W
1	10000000	0	0	0			
0	01000000	0	0	1			
0	11110000	0	1	0			

0	11110000	1 0 1			
0	10101010	1 1 1			
0	10101010	1 1 0			
0	10101010	1 0 0			

7. What are the purposes of ENABLE and CLEAR inputs in the 74LS259 DeMux.
8. What are the purposes of ENABLE and CLEAR inputs in the 74LS259 DeMux.
9. For the Demux circuit, set the Clear Input to "0" and Enable to "1". What is the result of this.
10. For the Demux circuit, set the Clear Input to "1" and Enable to "0". Then complete the following table according to the different combinations of the data input D and the selection lines C,B,A.

Selection Lines			Data Inputs	Data Outputs
C	B	A	D	Q ₇ Q ₆ Q ₅ Q ₄ Q ₃ Q ₂ Q ₁ Q ₀
0	0	0	1	
0	0	1	1	
0	1	0	0	
0	1	1	1	
1	0	0	0	
1	0	1	0	
1	1	0	1	
1	1	1	1	
0	0	1	0	
1	0	1	0	

11. According to the table in part 10, Explain how does 74LS259 IC Works.
12. Set Clear and Enable inputs to "1" and set the selection lines and data input randomly. What is the output of this? How does the 74LS259 work?.
13. How can we clear the outputs of 74LS259? Also, How can we clear the inputs of 74LS259?
14. For the Demux circuit, set the Clear Input to "0" and Enable to "0". Then complete the following table according to the different combinations of the data input D and the selection lines C,B,A.

Selection Lines			Data Inputs	Data Outputs
C	B	A	D	Q ₇ Q ₆ Q ₅ Q ₄ Q ₃ Q ₂ Q ₁ Q ₀
0	0	0	1	
0	0	1	1	
0	1	0	1	
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	1	
0	0	1	1	

1	1	0	0	
0	0	0	0	
1	0	1	0	
0	0	0	1	

15. According to the table in part 14, Explain how does 74LS259 IC Works.

VI. EXPERIMENT EXCERCISES

1. Use Multisim to implement the following function using an 8-to-1 MUX.

$$F(A, B, C, D) = ABC + \overline{A}BD + \overline{A}\overline{B}\overline{C} + AC\overline{D}$$

2. Use Multisim to implement the following using 74138 Decoder & external gates.

$$F_0(A, B, C) = \sum m(1, 3, 4), F_1(A, B, C) = \sum m(0, 2, 4, 7),$$

$$F_2(A, B, C) = \sum m(0, 1, 3, 5, 6), F_3(A, B, C) = \sum m(2, 6)$$

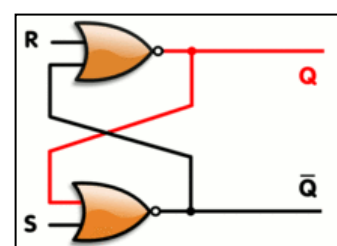
Weeks 12 & 13: Experiment 9: Sequential logic Design and Analysis: Analyzing and Implementing Flip-Flops

Students Names	Student IDs

I. EXPERIMENT INTRODUCTION.

1. Flip-flops (memory element)

flip-flop or latch is a **circuit** that has two stable states and can be used to store state information. The circuit can be made to change state by signals applied to one or more control inputs and will have one or two outputs. It is the basic storage element in **sequential logic**. Flip-flops and latches are a fundamental building block of **digital electronics** systems used in computers, communications, and many other types of systems.



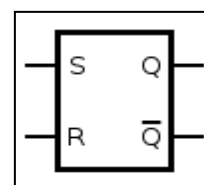
Flip-flops and latches are used as data storage elements. Such data storage can be used for storage of **state**, and such a circuit is described as **sequential logic**. When used in a **finite-state machine**, the output and next state depend not only on its current input, but also on its current state (and hence, previous inputs). It can also be used for counting of pulses, and for synchronizing variably-timed input signals to some reference timing signal.

Flip-flops can be either simple (transparent or opaque) or **clocked** (synchronous or edge-triggered); the simple ones are commonly called latches. The word latch is mainly used for storage elements, while clocked devices are described as flip-flops.

Flip-flops can be divided into common types: the SR ("set-reset"), D ("data" or "delay"), T ("toggle"), and JK types are the common ones. The behavior of a particular type can be described by what is termed the characteristic equation, which derives the "next" (i.e., after the next clock pulse) output, Q_{next} , in terms of the input signal(s) and/or the current output, Q .

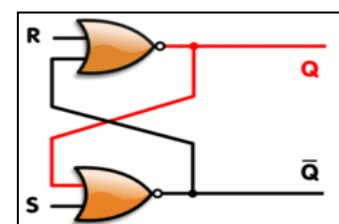
2. SR-Flip-flop (or RS-FF)

RS-flip-flop is the simplest possible memory element. It is constructed by feeding the outputs of two NOR gates back to the other NOR gates input. The inputs R and S are referred to as the Reset and Set inputs, respectively. The figure beside shows the RS Flip-Flop composed of two NOR Gates (An RS flip-flop can also be constructed from NAND gates).



To understand the operation of the RS-flip-flop (or RS-latch), consider the following scenarios:

- $S=1$ and $R=0$: The output of the bottom NOR gate is equal to 0, $Q'=0$.
- Hence both inputs to the top NOR gate are equal to one, thus, $Q=1$.



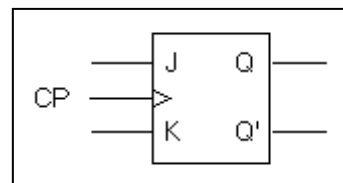
- Hence, the input combination $S=1$ and $R=0$ leads to FF being set to $Q=1$.
- $S=0$ and $R=1$: Similar to the arguments above, the outputs become $Q=0$ and $Q'=1$.
- We say that the flip-flop is reset.
- $S=0$ and $R=0$: Assume the flip-flop is set ($Q=0$ and $Q'=1$), then the output of the top NOR gate remains at $Q=1$ and the bottom NOR gate stays at $Q'=0$.
- Similarly, when the flip-flop is in a reset state ($Q=1$ and $Q'=0$), it will remain there with this input combination.
- Therefore, with inputs $S=0$ and $R=0$, the flip-flop remains in its state.
- $S=1$ and $R=1$: This input combination must be avoided.

R	S	Q	Q'	Comment
0	0			Hold state (No Change)
0	1	1	0	Set
1	0	0	1	Reset
1	1			Avoid (Not Allowed)

We can summarize the operation of the RS-flip-flop by the following truth table. Note, the output Q' is simply the inverse of Q .

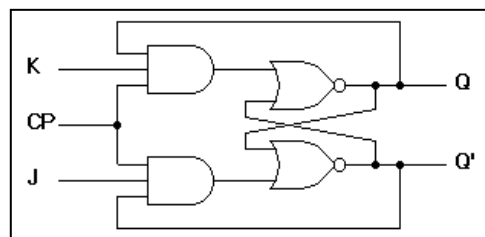
3. Clocked SR-Flip-flop (or CLK-RS-FF)

The clocked SR flip-flop shown in **Figure beside** consists of a basic NOR flip-flop and two AND gates. The outputs of the two AND gates remain at 0 as long as the clock pulse (or CP) is 0, regardless of the S and R input values. When the clock pulse goes to 1, information from the S and R inputs passes through to the basic flip-flop. With both $S=1$ and $R=1$, the occurrence of a clock pulse causes both outputs to momentarily go to 0. When the pulse is removed, the state of the flip-flop is indeterminate, i.e., either state may result, depending on whether the set or reset input of the flip-flop remains a 1 longer than the transition to 0 at the end of the pulse.



4. JK-Flip-flop (or JK-FF)

A JK flip-flop is a refinement of the SR flip-flop in that the indeterminate state of the SR type is defined in the JK type. Inputs J and K behave like inputs S and R to set and clear the flip-flop (note that in a JK flip-flop, the letter J is for set and the letter K is for clear). When logic 1 inputs are applied to both J and K simultaneously, the flip-flop switches to its complement state, i.e., if $Q=1$, it switches to $Q=0$ and vice versa.



A clocked JK flip-flop is shown in **Figure beside**. Output Q is ANDed with K and CP inputs so that the flip-flop is cleared during a clock pulse only if Q was previously 1. Similarly, output Q' is ANDed with J and CP inputs so that the flip-flop is set with a clock pulse only if Q' was previously 1.

Q	J	K	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

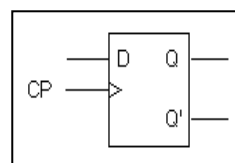
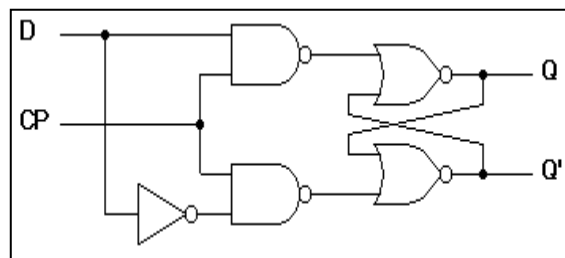
Note that because of the feedback connection in the JK flip-flop, a CP signal which remains a 1 (while $J=K=1$) after the outputs have been complemented once will cause repeated and continuous transitions of the outputs. To avoid this, the clock pulses must have a time duration less than the propagation delay through the flip-flop. The restriction on

J	K	Q	Q'
0	0	No Change	
0	1	0	1
1	0	1	0
1	1	Change	

the pulse width can be eliminated with a master-slave or edge-triggered construction. The same reasoning also applies to the T flip-flop presented next.

5. Data-Flip-flop (or D - FF)

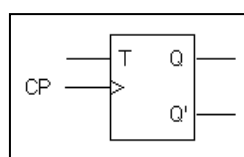
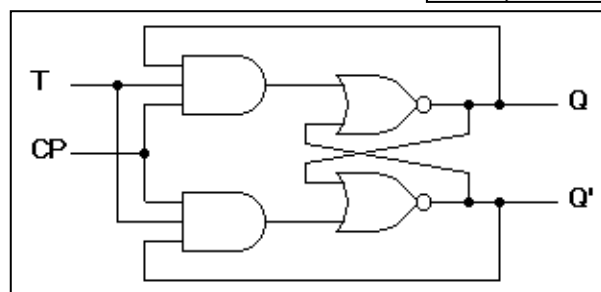
The D flip-flop (Clocked D-FF) shown in **Figure** beside is a modification of the clocked SR flip-flop. The D input goes directly into the S input and the complement of the D input goes to the R input. The D input is sampled during the occurrence of a clock pulse. If it is 1, the flip-flop is switched to the set state (unless it was already set). If it is 0, the flip-flop switches to the clear state.



Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

6. Toggle-Flip-flop (or T - FF)

The T flip-flop is a single input version of the JK flip-flop. As shown in **Figure** below, the T flip-flop is obtained from the JK type if both inputs are tied together. The output of the T flip-flop "toggles" with each clock pulse.



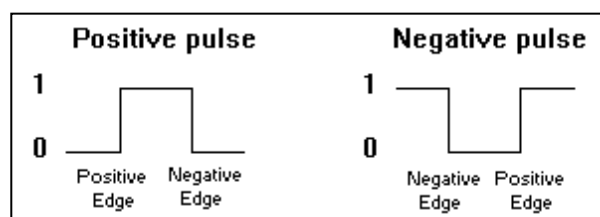
Q	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

7. Triggering of Flip-flops

The state of a flip-flop is changed by a momentary change in the input signal. This change is called a trigger and the transition it causes is said to trigger the flip-flop. The basic circuits of RS-Flip-Flop require an input trigger defined by a change in signal level. This level must be returned to its initial level before a second trigger is applied. Clocked flip-flops are triggered by pulses.

The feedback path between the combinational circuit and memory elements can produce instability if the outputs of the memory elements (flip-flops) are changing while the outputs of the combinational circuit that go to the flip-flop inputs are being sampled by the clock pulse. A way to solve the feedback timing problem is to make the flip-flop sensitive to the pulse transition rather than the pulse duration.

The clock pulse goes through two signal transitions: from 0 to 1 and the return from 1 to 0. As shown in the figure below the positive transition is defined as the positive edge and the negative transition as the negative edge.



The clocked flip-flops already introduced are triggered during the positive edge of the pulse, and the state transition starts as soon as the pulse reaches the logic-1 level. If the other inputs

change while the clock is still 1, a new output state may occur. If the flip-flop is made to respond to the positive (or negative) edge transition only, instead of the entire pulse duration, then the multiple-transition problem can be eliminated.

8. Summary of Flip-flops

- So far we have considered combinatorial logic.
- That is circuits for which the output depends only on the inputs.
- In many instances it is desirable to have the next output depend on the current output.
- A simple example is a counter, where the next number to be output is determined by the current number stored.
- Circuits that remember their current output or state are often called sequential logic.
- Clearly, sequential logic requires the ability to store the current state.
- In other words, memory is required by sequential logic circuits.
- We will investigate basic circuits that have the ability to store bits of data.
- Memory elements can be constructed by appropriately feeding back the output of gates to the input.
- The most basic memory element is an RS flip flop which consists of two NOR gates.
- By adding two more gates (and an inverter) we can construct a D-flip-flop.
- The D-flip-flop has an Enable input and a data input.
- When the Enable input is 1, the data input is read into the flip-flop.
- When the Enable input is 0, the currently stored value is held regardless of the data input.
- D-flip-flops can be used to construct sequential logic circuits such as counters or shift-registers.

II. MAIN OBJECTIVES

- Analyze RS, D, JK, T- Flip-flops and observing their operations and obtaining their truth tables.
- Form RS- Flip-flop by using 74LS02 and 74LS00 ICs and forming D-Flip-flop using JK-Flip-flop and Invertors.
- **Test Standard : IEEE 991-1986**

III. EXPERIMENT PROCEDURE

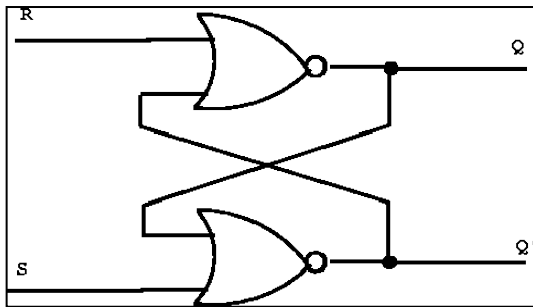
The procedure for this experiment is the same as lab1

IV. THE REQUIRED DIGITAL CIRCUITS

1. Circuit # 1: Basic RS-FF:

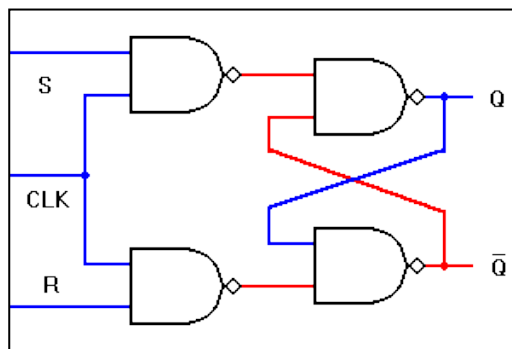
Here, you will use the 74LS02 IC to build the circuit of basic RS-FF on the Y-01 Experiment Set. After building the circuit, you have to fill the truth table.

R	S	Q	Q'
0	0		
0	1		
1	0		
1	1		



Circuit # 2: Clocked RS-FF:

Here you will use the 74LS00 IC to build the circuit of Clocked RS-FF on the Y-01 Experiment Set. After building the circuit, you have to fill the truth table.








Clk	R	S	Q	Q'
1	1	0		
1	0	0		
1	0	1		
0	0	1		
0	0	0		
0	1	0		
0	0	0		
0	0	1		
1	0	1		
1	0	0		
1	1	0		
0	1	0		
0	0	0		

Circuit # 3: Analyzing JK-FF using 74LS76 :

Here you will use the 74LS76 IC to test and analyze the circuit of JK-FF on the Y-01 Experiment Set.

- Regarding the Clock pin, you have to connect it to falling edge pulse generator. After building the circuit, you have to fill the truth table.

CONTROL		INPUTS			OUTPUTS		EXPLANATIONS
$\overline{\text{Preset}}$	$\overline{\text{Clear}}$	J	K	$\overline{\text{CLK}}$	Q	\overline{Q}	
0	0	X	X	X			
1	0	X	X	X			
0	1	X	X	X			
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				
1	1	0	0				



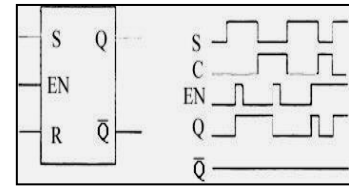
1. Using the training kits; Construct the required Circuits.
 - a. For Cir #1: Use Switches A, B for inputs S, R and LEDs 7, 6 for Outputs Q, Q'.
 - b. For Cir #2: Use Switches A, B, C for inputs S, R, CLK and LEDs 7, 6 for Outputs Q, Q'.
 - c. For Cir #3: Use Switches A, B, C for inputs J, K, CLK and LEDs 7, 6 for Outputs Q, Q'.
 - d. For Cir #4: Use Switches A, B, C, D for inputs CL, \overline{J} , K, Pr $\overline{\text{and}}$ LEDs 7, 6 for Outputs Q, Q'. Also, connect the CLK to the falling edge pulse generator (Pulser 74LS02).
2. For Circuit#1, are the outputs always inverse of each other? Explain.
3. For Circuit#1, When S=1, is Q always 1? Why?
4. For Circuit#1, When R=0, is Q always 0? Why?
5. For Circuit#1, When both R and S are 0, does Q preserve its previous state?
6. For Circuit#2, When does the change in inputs affect the output? Explain
7. For Circuit#2, Why do the changes in inputs not affect the output when CLK is 0?
8. For Circuit#3, According to the results you got in the table; when both J and K are 0 and the CLK pulse occurs, Does the FF preserve its previous state? Explain.
9. For Circuit#3, According to the results you got in the table; when the CLK input falls from 1 to 0, Does it trigger FF? Explain.
10. For Circuit#3, According to the results you got in the table; When are the controls Clear and Preset Effective? Are those controls more prior than J-K and CLK.

VI. EXPERIMENT EXCERSICES

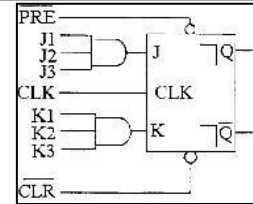
1. Design the RS-FF using NAND Gates. Provide the truth table also.
2. Give one example of the use of an S-R flip-flop?
3. How can the cross-coupled NAND flip-flop be made to have active-HIGH S-R inputs?
4. When is a flip-flop said to be transparent?
5. A J-K flip-flop is in a "no change" condition when_____: (Choose one answer, and explain)
A. J = 1, K = 1. **B.** J = 1, K = 0. **C.** J = 0, K = 1. **D.** J = 0, K = 0.
6. What is the hold condition of a flip-flop?
7. If an active-HIGH S-R latch has a 0 on the S input and a 1 on the R input and then the R input goes to 0, the latch will be_____: (Choose one answer, and explain)
A. SET. **B.** RESET. **C.** CLEAR. **D.** INVALID (NOT ALLOWED).

8. Give one disadvantage of an S-R flip-flop?

9. A gated S-R FF (Latch) and its associated waveforms are shown below. What is wrong and what could be causing the problem?



10. The circuit given below fails to function; the inputs are checked with a logic probe and the following indications are obtained: CLK, J1, J2, J3, K1, K2, and K3 are pulsing. Q and $\overline{\text{CLR}}$ are HIGH. $\overline{\text{Q}}$ and PRE are LOW. What could be causing the problem?

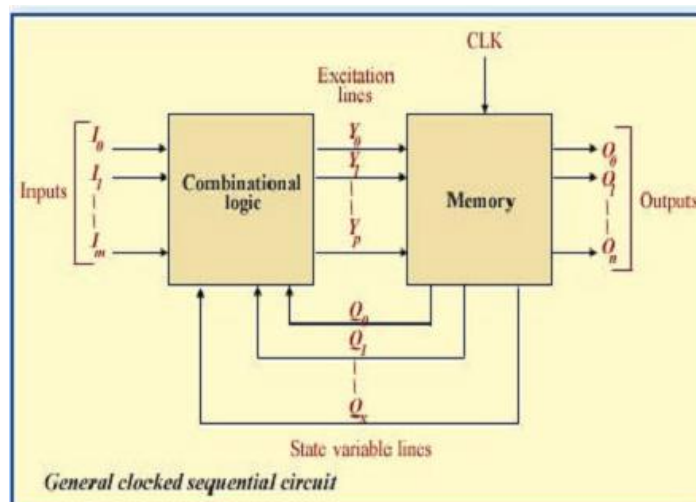


Week 14 Experiment 10: Design and analyze the synchronous counter circuits: Up/Down Synchronous Counters Using Composed JK-Flip-flop

Students Names	Student IDs

I. EXPERIMENT INTRODUCTION.

A synchronous finite-state machine changes state only on the clocking event. The following diagram shows the general sequential circuit that consists of a combinational logic section and a memory section (flip-flops). The Combinational logic module is for us, as the designer, to match the design specifications.



1. Counters

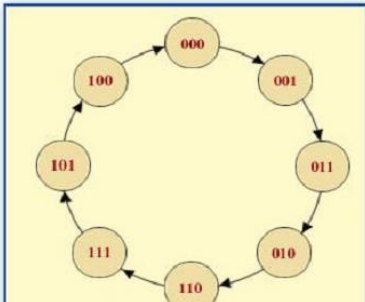
Counter design is a good place for you to start understanding the design process for finite-state machines. Counters are the simplest possible finite-state machines. They typically have only a single input instructing them to count (often just the clock), and their outputs are nothing more than their current state.

2. Counter Design Procedure:

1. Describe a general sequential circuit in terms of its basic parts and its input and outputs.
2. Develop a state diagram for a given sequence.
3. Develop a next-state table for a specific counter sequence.
4. Create a FF transition table.
5. Use K-map to derive the logic equations.
6. Implement a counter to produce a specified sequence of states.

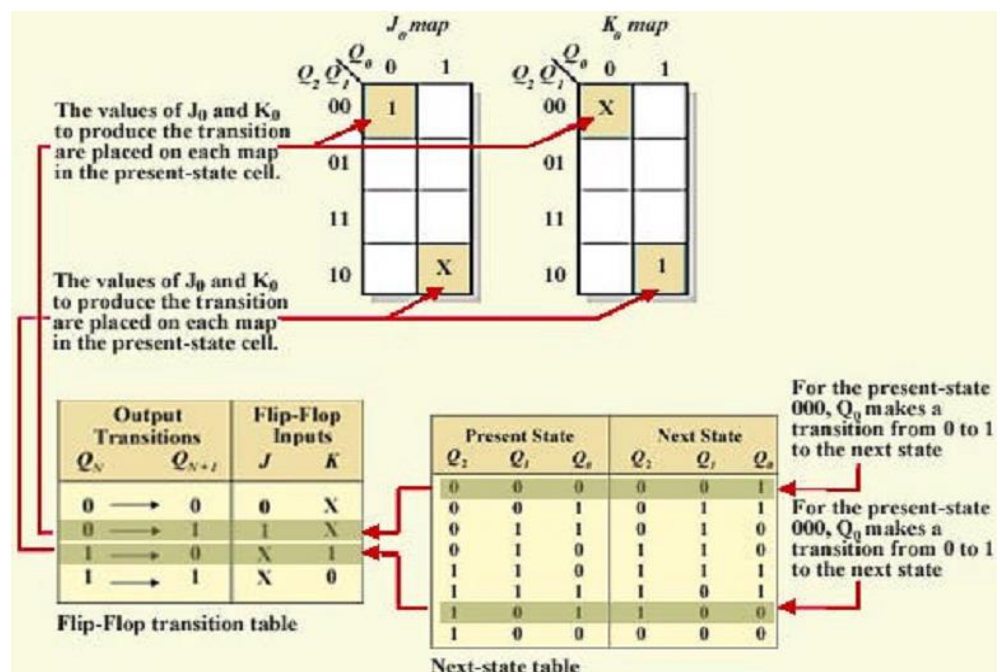
Let's start with a simple counter, a 3-bit binary up-counter. We begin the design process by understanding how the counter is to operate. A convenient way to describe this is with a graphical specification called a state transition diagram.

3. Design the 3-bit Gray code counter

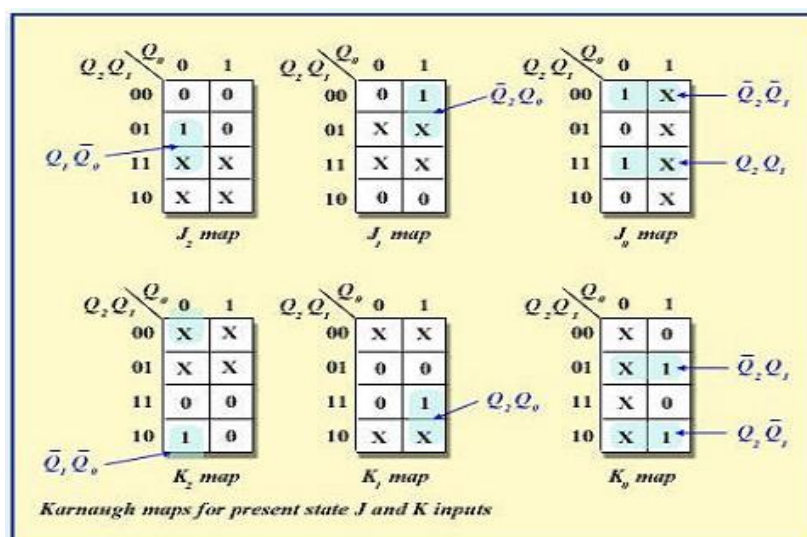
a. Step 1: State Diagram	b. Step 2: Next-State Table	c. Step 3: Flip-Flop Transition Table																																																																																				
<div><p>State diagram for a 3-bit Gray code counter.</p></div>	<table><thead><tr><th colspan="3">Present State</th><th colspan="3">Next State</th></tr><tr><th>Q_2</th><th>Q_1</th><th>Q_0</th><th>Q_2</th><th>Q_1</th><th>Q_0</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></tbody></table>	Present State			Next State			Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	0	0	0	0	0	1	0	0	1	0	1	1	0	1	1	0	1	0	0	1	0	1	1	0	1	1	0	1	1	1	1	1	1	1	0	1	1	0	1	1	0	0	1	0	0	0	0	0	<p>Transition table for a J-K Flip-Flop</p> <table><thead><tr><th colspan="2">Output Transitions</th><th colspan="2">Flip-Flop Inputs</th></tr><tr><th>Q_N</th><th>Q_{N+1}</th><th>J</th><th>K</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td><td>X</td></tr><tr><td>0</td><td>1</td><td>1</td><td>X</td></tr><tr><td>1</td><td>0</td><td>X</td><td>1</td></tr><tr><td>1</td><td>1</td><td>X</td><td>0</td></tr></tbody></table> <p>Q_N: present state, Q_{N+1}: next state X: "don't care"</p>	Output Transitions		Flip-Flop Inputs		Q_N	Q_{N+1}	J	K	0	0	0	X	0	1	1	X	1	0	X	1	1	1	X	0
Present State			Next State																																																																																			
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0																																																																																	
0	0	0	0	0	1																																																																																	
0	0	1	0	1	1																																																																																	
0	1	1	0	1	0																																																																																	
0	1	0	1	1	0																																																																																	
1	1	0	1	1	1																																																																																	
1	1	1	1	0	1																																																																																	
1	0	1	1	0	0																																																																																	
1	0	0	0	0	0																																																																																	
Output Transitions		Flip-Flop Inputs																																																																																				
Q_N	Q_{N+1}	J	K																																																																																			
0	0	0	X																																																																																			
0	1	1	X																																																																																			
1	0	X	1																																																																																			
1	1	X	0																																																																																			

d. Step 4: Karnaugh Maps

The following diagram shows the steps to create separate next states of separate J and K from the current states of J and K.



Karnaugh maps for present-state J and K inputs for the 3-bit Gray code counter.

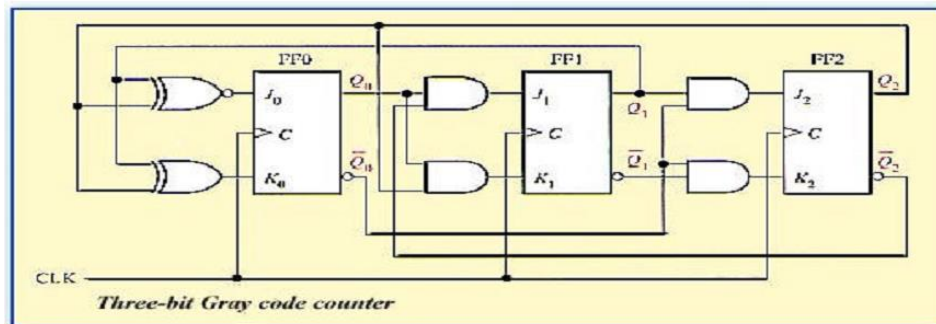


e. Step 5: Logic Expressions for Flip-flop Inputs

The next-state J and K outputs for a 3-bit Gray code counter.

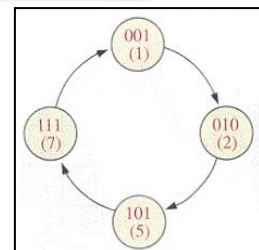
f. Step 6: Counter Implementation

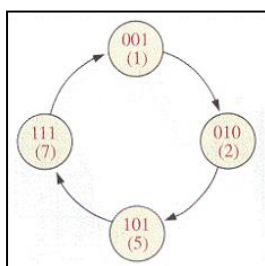
The hardware diagram of the 3-bit Gray code counter



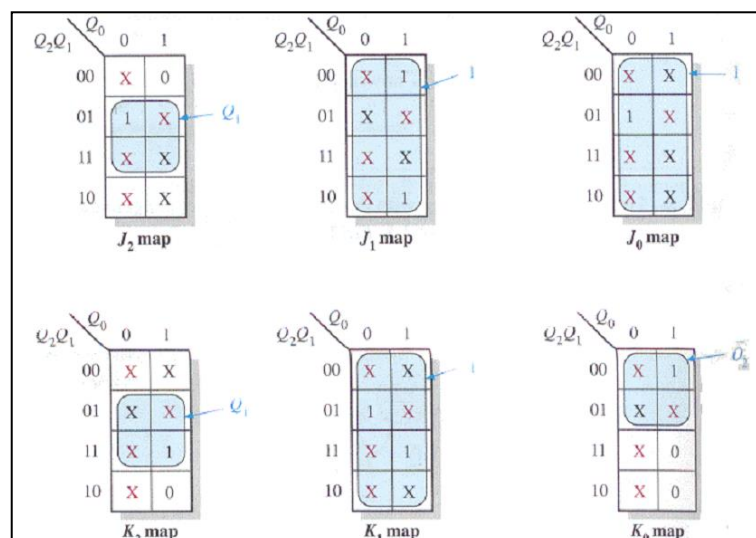
4. Another Example

Design a counter with the irregular binary count sequence shown in the state diagram of Figure below.



a. Step 1: State Diagram	b. Step 2: Next-State Table	c. Step 3: Flip-Flop Transition Table																																																												
	<table><thead><tr><th colspan="3">Present State</th><th colspan="3">Next State</th></tr><tr><th>Q_2</th><th>Q_1</th><th>Q_0</th><th>Q_2</th><th>Q_1</th><th>Q_0</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr></tbody></table>	Present State			Next State			Q_2	Q_1	Q_0	Q_2	Q_1	Q_0	0	0	1	0	1	0	0	1	0	1	0	1	1	0	1	1	1	1	1	1	1	0	0	1	<p>Transition table for a J-K Flip-Flop</p> <table><thead><tr><th colspan="2">Output Transitions</th><th colspan="2">Flip-Flop Inputs</th></tr><tr><th>Q_n</th><th>Q_{n+1}</th><th>J</th><th>K</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td><td>X</td></tr><tr><td>0</td><td>1</td><td>1</td><td>X</td></tr><tr><td>1</td><td>0</td><td>X</td><td>1</td></tr><tr><td>1</td><td>1</td><td>X</td><td>0</td></tr></tbody></table> <p>Q_n: present state, Q_{n+1}: next state X: "don't care"</p>	Output Transitions		Flip-Flop Inputs		Q_n	Q_{n+1}	J	K	0	0	0	X	0	1	1	X	1	0	X	1	1	1	X	0
Present State			Next State																																																											
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0																																																									
0	0	1	0	1	0																																																									
0	1	0	1	0	1																																																									
1	0	1	1	1	1																																																									
1	1	1	0	0	1																																																									
Output Transitions		Flip-Flop Inputs																																																												
Q_n	Q_{n+1}	J	K																																																											
0	0	0	X																																																											
0	1	1	X																																																											
1	0	X	1																																																											
1	1	X	0																																																											

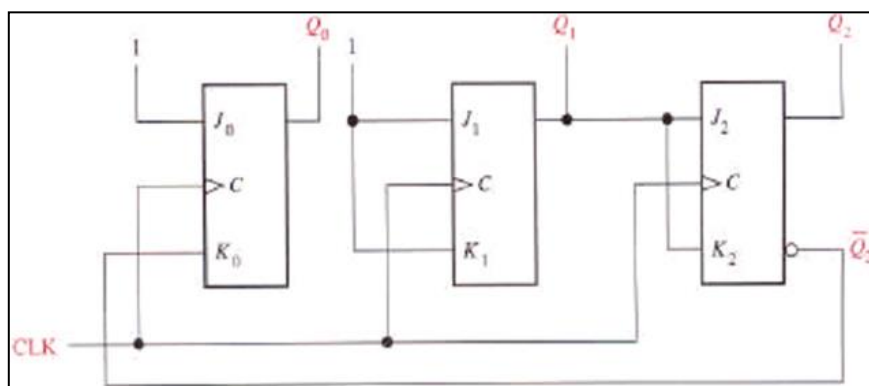
d. Step 4: Karnaugh Maps



e. Step 5: Logic Expressions for Flip-flop Inputs

.....

f. Step 6: Counter Implementation



II. MAIN OBJECTIVES.

- Analyze the working principle of the synchronous counters for both up and down counting.
- Familiarize with counter design steps for any counting policy and size.
- **Test Standard : IEEE 991-1986**

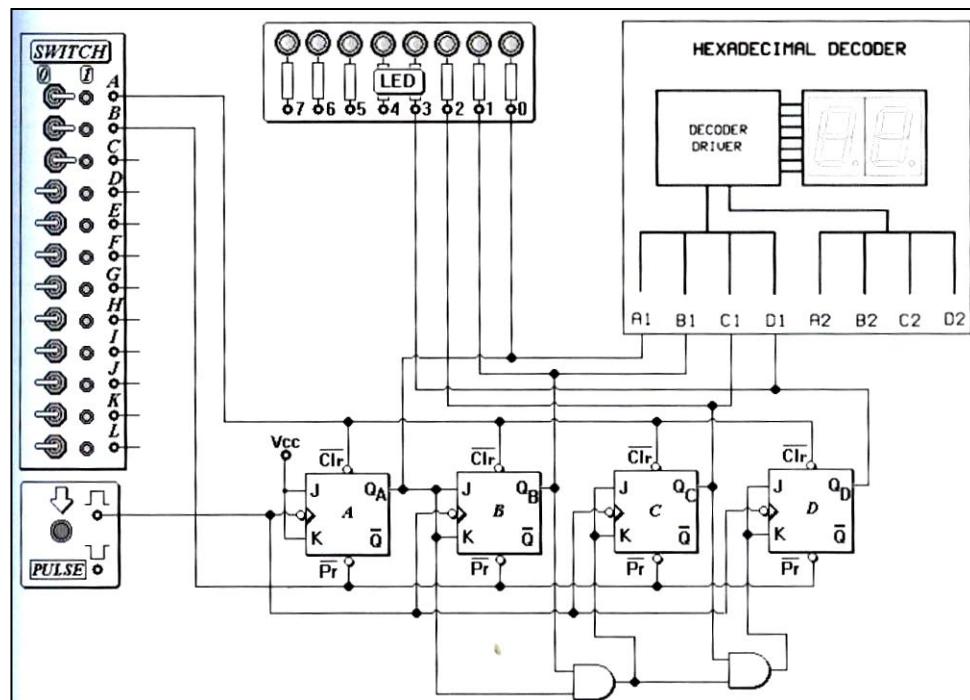
III. EXPERIMENT PROCEDURE.

The procedure for this experiment is the same as lab1

IV. The Required Digital Circuits:

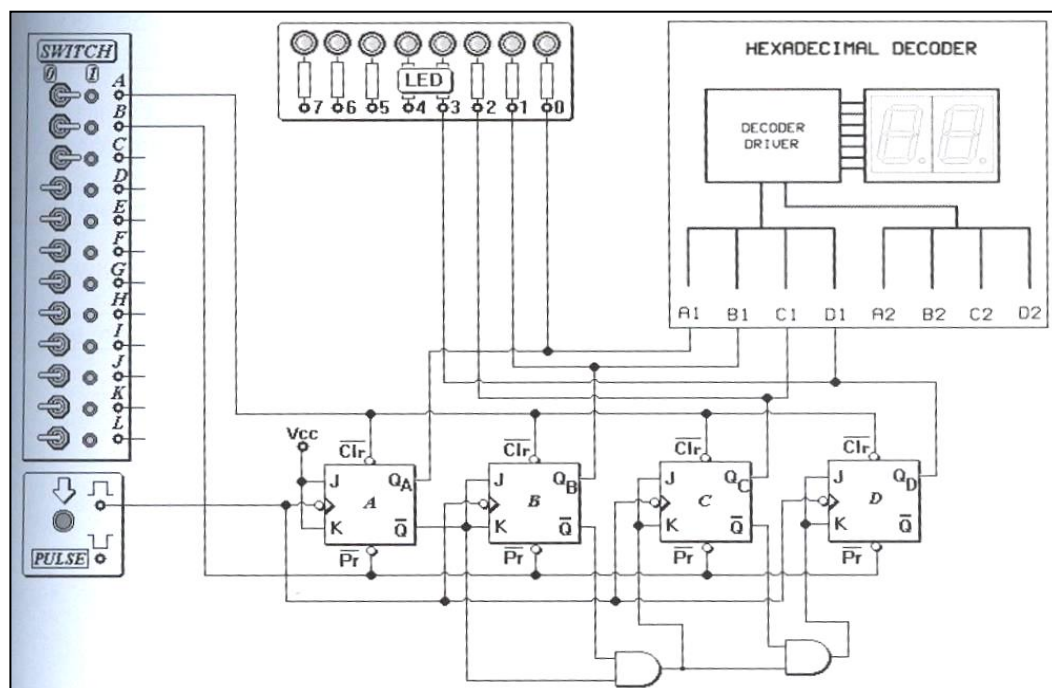
Circuit # 1: Up Synchronous Counters:

here you are about designing a 4-bit hexadecimal synchronous counter in which it count incrementally (0000, 0001 ...1111).



Circuit # 2: Down Synchronous Counters:

here you are about designing a 4-bit hexadecimal synchronous counter in which it count decrementally (1111, 1110 ...0000).



V. EXPERIMENTAL WORK

Pulse	Circuit 1 outputs					Circuit 2 outputs				
	Q _D	Q _C	Q _B	Q _A	HEX	Q _D	Q _C	Q _B	Q _A	HEX
0										
1										
2										

1. Using the training kits; Construct the required Circuits for Up and Down Counters. Make sure you connect all inputs, outputs and GNDs pins appropriately.

2. For Circuit #1, Set the PR' pin (Switch B) to "1" and passivate it. Set the CLR' pin (Switch A) temporarily to "0" and then set it back to "1". Has "0" been seen on the display? Explain why.

3. For Circuit #1, apply the first clock pulse to the FFs using the Pulser? Explain the change in the outputs.

4. For Circuit #1, fill in the table with the applied pulses.

3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				

5. For Circuit #2, Set the CLEAR' input (Switch A) to "1". Set the PRESET' input temporarily to "0" and then set it back to "1". Explain the operation done. What are the outputs

6. For Circuit #2, fill in the table with the applied pulses.

7. Why we call these counters Synchronous?

8. Explain the purpose of Pulser in both circuits?

9. What are the differences you made between circuit#1 and circuit #2?

APPENDIXES

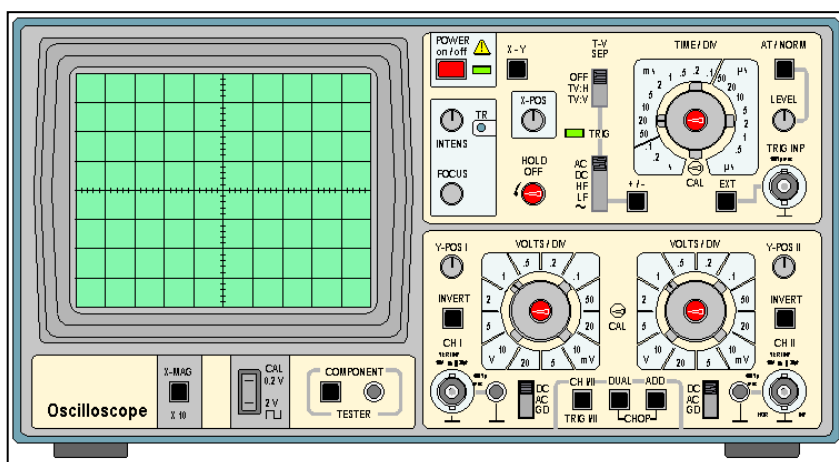
Appendix A: Using An Oscilloscope

What does an oscilloscope do?

An oscilloscope is easily the most useful instrument available for testing circuits because it allows you to *see* the signals at different points in the circuit. The best way of investigating an electronic system is to monitor signals at the input and output of each system block, checking that each block is operating as expected and is correctly linked to the next. With a little practice, you will be able to find and correct faults quickly and accurately.

An oscilloscope is an impressive piece of kit:

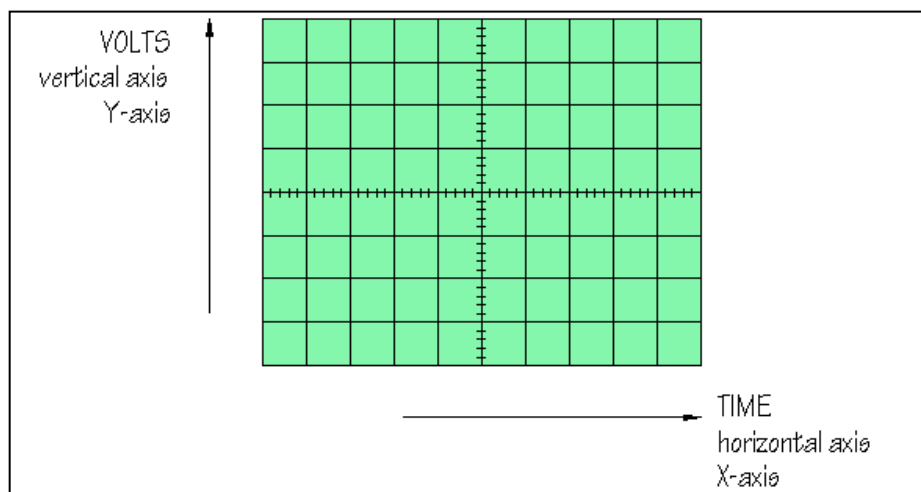
The diagram shows a *Hameg HM 203-6* oscilloscope, a popular instrument in UK schools. Your oscilloscope may look different but will have similar controls.



Faced with an instrument like this, students typically respond either by twiddling every knob and pressing every button in sight, or by adopting a glazed expression. Neither approach is specially helpful. Following the systematic description below will give you a clear idea of what an oscilloscope is and what it can do.

The function of an oscilloscope is extremely simple: it draws a V/t graph, a graph of voltage against time, voltage on the vertical or Y-axis, and time on the horizontal or X-axis.

As you can see, the **screen** of this oscilloscope has 8 squares or divisions on the vertical axis, and 10 squares or divisions on the horizontal axis. Usually, these squares are 1 cm in each direction:



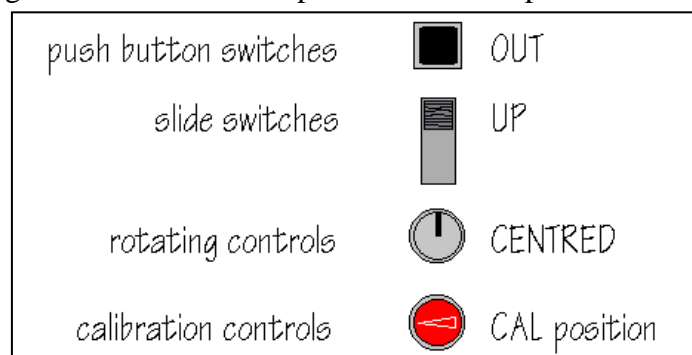
Many of the controls of the oscilloscope allow you to change the vertical or horizontal scales of the V/t graph, so that you can display a clear picture of the signal you want to investigate.

'Dual trace' oscilloscopes display two V/t graphs at the same time, so that simultaneous signals from different parts of an electronic system can be compared.

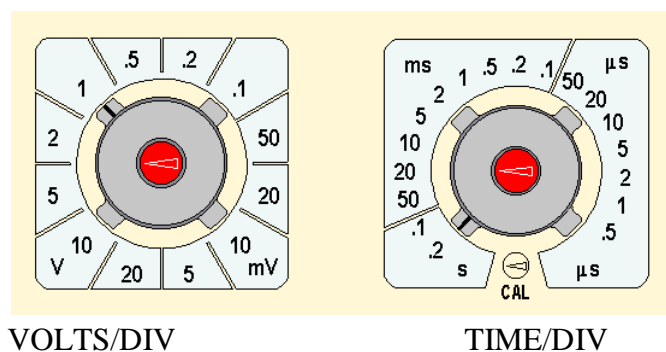
Setting up

1. Someone else may have been twiddling knobs and pressing buttons before you. Before you switch the oscilloscope on, check that all the controls are in their 'normal' positions. For the *Hameg HM 203-6*, this means that:

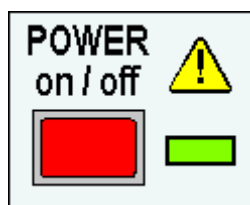
- ✓ all push button switches are in the OUT position
- ✓ all slide switches are in the UP position
- ✓ all rotating controls are CENTRED
- ✓ the central TIME/DIV and VOLTS/DIV and the HOLD OFF controls are in the calibrated, or CAL position
- ✓ Check through all the controls and put them in these positions:



2. Set both VOLTS/DIV controls to 1 V/DIV and the TIME/DIV control to 0.2 s/DIV, its slowest setting:



3. Switch ON, red button, top centre:

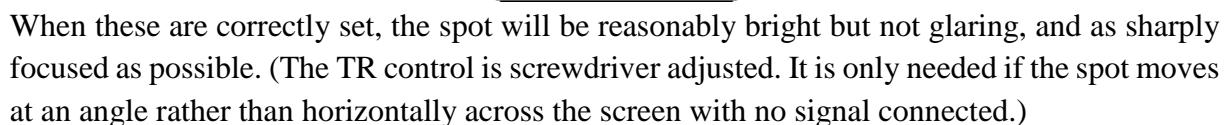


The green LED illuminates and, after a few moments, you should see a small bright spot, or **trace**, moving fairly slowly across the screen.

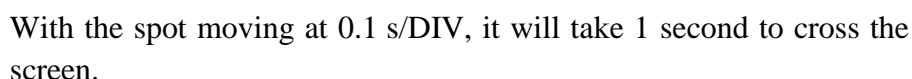
4. Find the Y-POS 1 control:



5. Now investigate the INTENSITY and FOCUS controls:



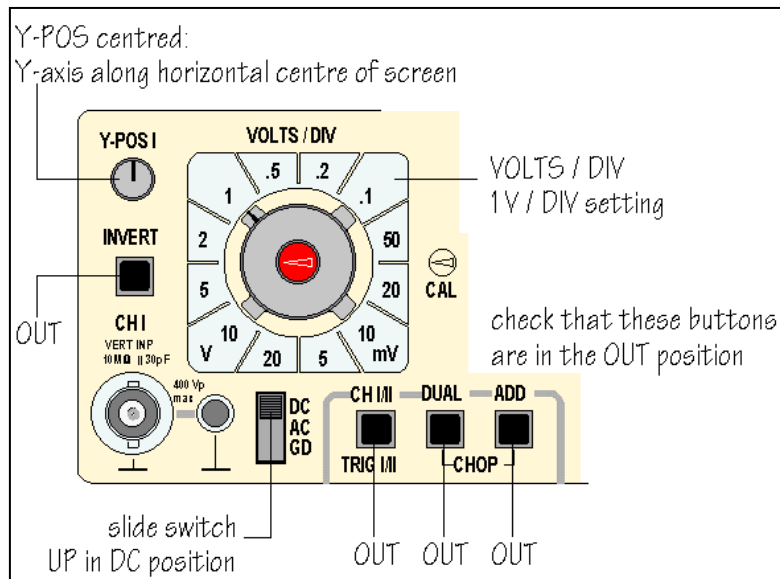
With 10 squares across the screen and the spot moving at 0.2 s/DIV, how long does it take for the spot to cross the screen? The answer is $0.2 \times 10 = 2$ s. Count seconds. Does the spot take 2 seconds to cross the screen? Now rotate the TIME/DIV control clockwise:



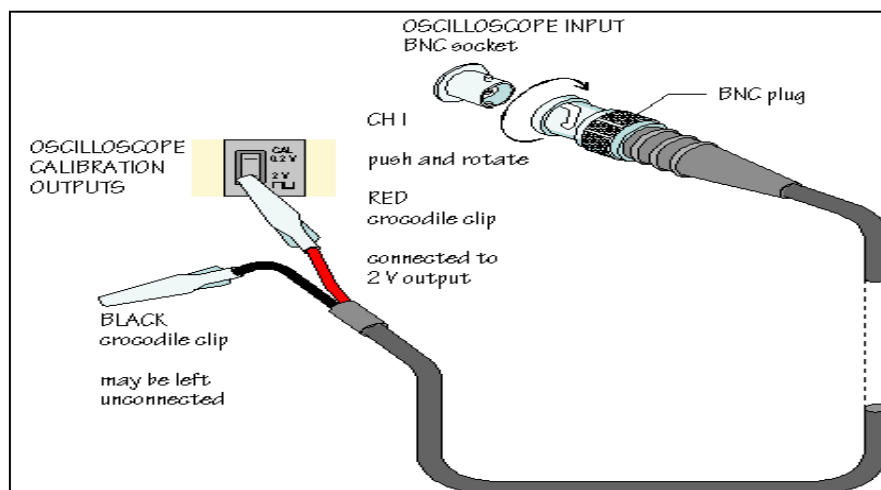
Continue to rotate TIME/DIV clockwise. With each new setting, the spot moves faster. At around 10 ms/DIV, the spot is no longer separately visible. Instead, there is a bright line across the screen. This happens because the screen remains bright for a short time after the spot has passed, an effect which is known as the **persistence** of the screen. It is useful to think of the spot as still there, just moving too fast to be seen.

Keep rotating TIME/DIV. At faster settings, the line becomes fainter because the spot is moving very quickly indeed. At a setting of $10 \mu\text{s}/\text{DIV}$ how long does it take for the spot to cross the screen?

7. The VOLTS/DIV controls determine the vertical scale of the graph drawn on the oscilloscope screen. Check that VOLTS/DIV 1 is set at 1 V/DIV and that the adjacent controls are set correctly:

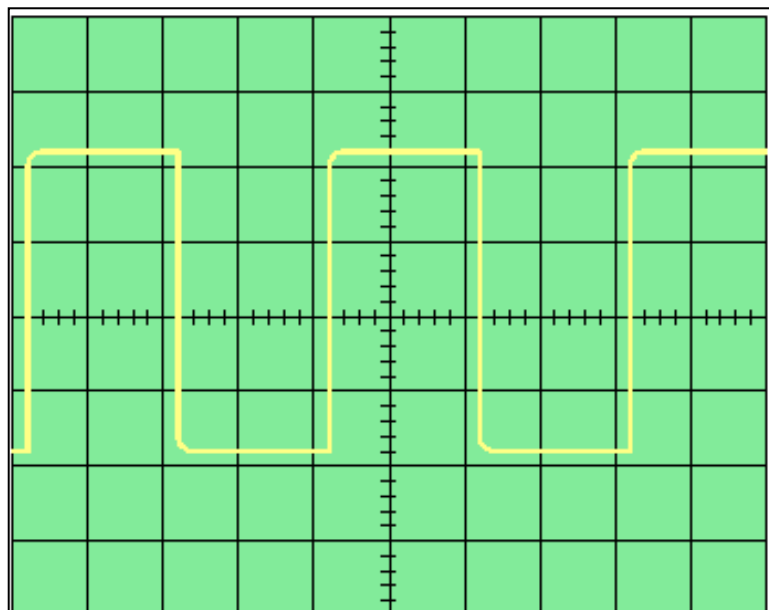


The *Hameg HM 203-6* has a built in source of signals which allow you to check that the oscilloscope is working properly. A connection to the input of channel 1, CH 1, of the oscilloscope can be made using a special connector called a BNC plug, as shown below:

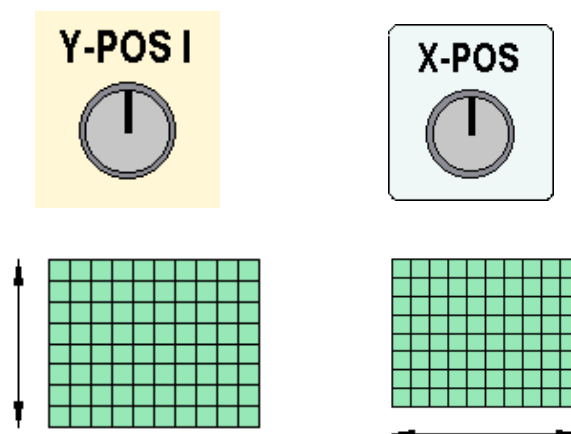


The diagram shows a lead with a BNC plug at one end and crocodile clips at the other. When the crocodile clip from the red wire is clipped to the lower metal terminal, a 2 V square wave is connected to the input of CH 1.

Adjust VOLTS/DIV and TIME/DIV until you obtain a clear picture of the 2 V signal, which should look like this:



Check on the effect of Y-POS 1 and X-POS:



What do these controls do?

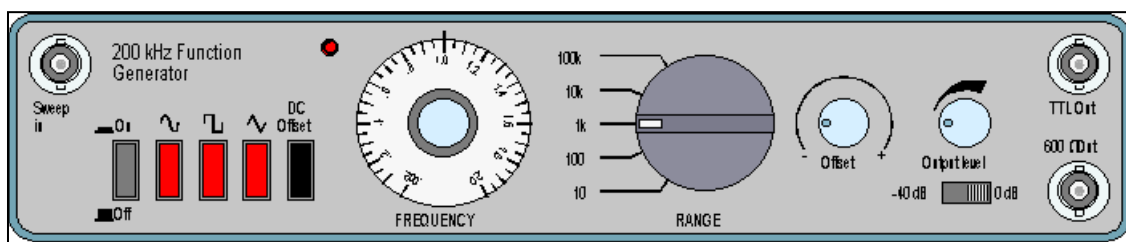
Y-POS 1 moves the whole trace vertically up and down on the screen, while X-POS moves the whole trace from side to side on the screen. These control are useful because the trace can be moved so that more of the picture appears on the screen, or to make measurements easier using the grid which covers the screen.

You have now learned about and used the most important controls on the oscilloscope.

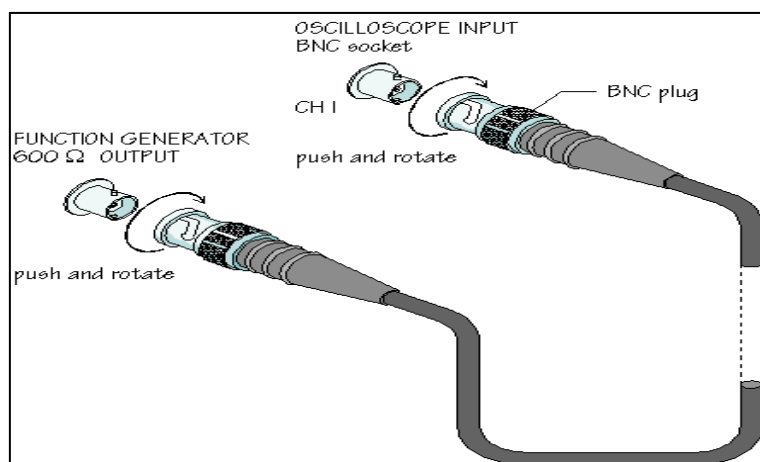
You know that the function of an oscilloscope is to draw a V/t graph. You know how to put all the controls into their 'normal' positions, so that a trace should appear when the oscilloscope is switched on. You know how to change the horizontal scale of the V/t graph, how to change the vertical scale, and how to connect and display a signal. What is needed now is practice so that all of these controls become familiar.

Connecting a function generator

The diagram shows the appearance of a *Thandar TG101* function generator, one of many types used in UK schools:

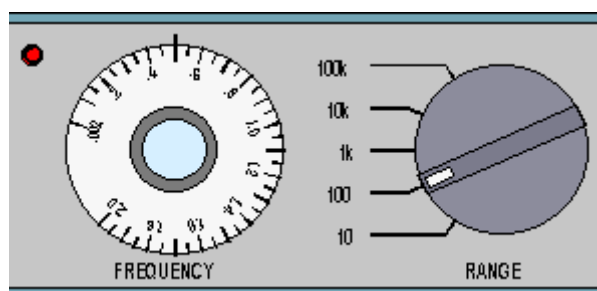


Again, your function generator, or signal generator, may look different but is likely to have similar controls. The *Thandar TG101* has push button controls for On/Off switching and for selecting either sine, square, or triangular wave shapes. Most often the 600 Ω output is used. This can be connected to the CH 1 input of the oscilloscope using a BNC-BNC lead, as follows:



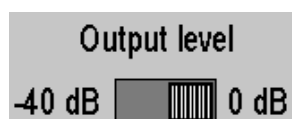
Switch on the function generator and adjust the output level to produce a visible signal on the oscilloscope screen. Adjust TIME/DIV and VOLTS/DIV to obtain a clear display and investigate the effects of pressing the waveform shape buttons.

The rotating FREQUENCY control and the RANGE switch are used together to determine the frequency of the output signal. With the settings shown in the diagram above, the output frequency will be 1 kHz. How would you change these setting to obtain an output frequency of 50 Hz? This is done by moving the RANGE switch to '100' and the FREQUENCY control to '.5':



Experiment with these controls to produce other frequencies of output signal, such as 10 Hz, or 15 kHz. Whatever frequency and amplitude of signal you select, you should be able to change the oscilloscope settings to give a clear V/t graph of the signal on the oscilloscope screen.

The remaining features of the function generator are less often used. For example, it is possible to change the output frequency by connecting suitable signals to the 'Sweep in' input. The DC Offset switch and the Offset control allow you to add a DC voltage component to the output signal producing a complex waveform as described in Chapter 4. The output level switch is normally set to 0 dB:



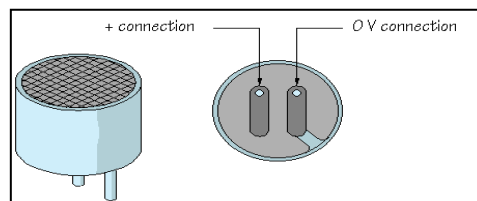
This gives an output signal with a peak amplitude which can be easily adjusted up to several volts. In the -40 dB position, the amplitude of the output signal is reduced to a few millivolts. Such small signals are used for testing amplifier circuits.

The TTL output produces pulses between 0 V and 5 V at the selected frequency and is used for testing logic circuits.

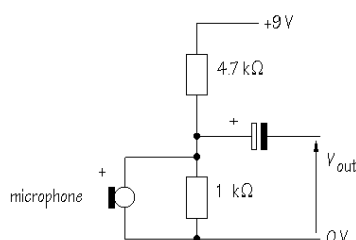
Microphones audio signals and amplifiers

This part of the Practical is an investigation of microphones, audio signals and amplifiers, intended to develop your prototype board skills and giving you experience of using the oscilloscope to monitor signals in a simple circuit. (The operational amplifier circuit used is explained fully in Chapter ?)

The diagram shows an easily available type of microphone, called a **cermet microphone**:



The microphone has separate + and 0 V connections. Can you see that the 0 V connection is connected to the metal case? Check these connections on the real component. To get the microphone to work, you need to provide a voltage across it using a voltage divider circuit:



From the voltage divider formula, the voltage expected across the microphone is:

$$V_{out} = \frac{R_{bottom}}{R_{bottom} + R_{top}} \times V_{in}$$

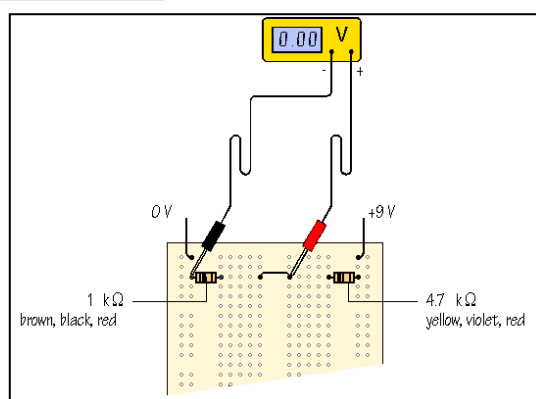
Substituting:

$$V_{out} = \frac{1}{1 + 4.7} \times 9 = 1.58 \text{ V}$$

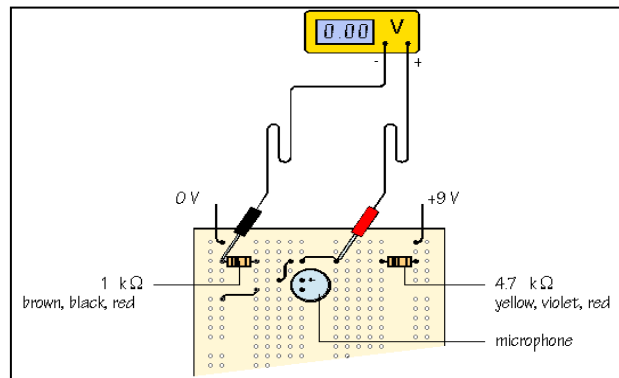
Build the voltage divider part of the circuit on prototype boards as follows:

Measure the voltage between the resistors. How closely does the measured value agree with the calculated one?

Small differences can arise if you have not adjusted the power supply voltage to exactly 9 V and also because the resistors may not have precisely their marked values. Remember, resistors are manufactured to a tolerance, usually $\pm 5\%$, so that their values are not exact.

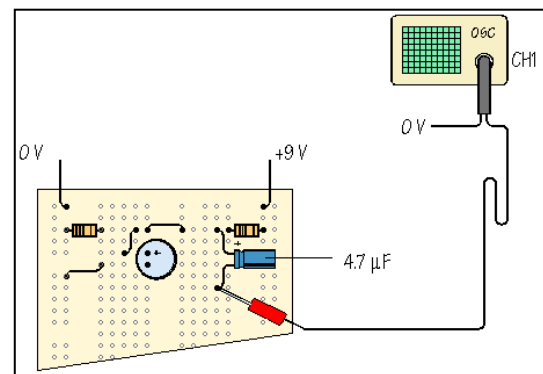


Now add the microphone to the circuit, taking care to get its + and 0 V connections the right way round:



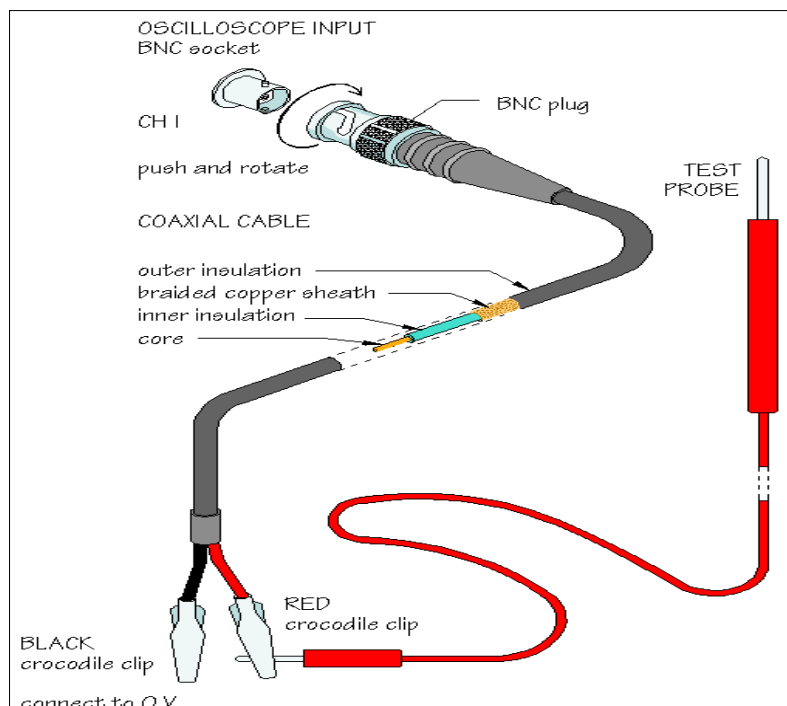
Usually, this results in a small decrease in the voltage divider voltage, because the microphone is now in parallel with the 1 kΩ resistor. In other words, R_{bottom} is reduced. Another way of explaining this is to say that some of the current flows through the microphone, leaving a little less flowing through the 1 kΩ.

Check the polarity of a 4.7 μF or 10 μF capacitor (longer leg positive, stripe negative) and connect this as indicated below:



In this circuit, the capacitor blocks DC voltages, but allows AC voltages, including audio signal, to pass. You will find out more about this in Chapter 5.

The arrangement outlined below is a very convenient way of setting up an oscilloscope to make measurements from the prototype circuit:



Once the crocodile clip corresponding to the black lead has been connected to 0 V, it can be ignored. This leaves the test probe which can be connected to any point in the circuit to monitor the signals present.

Connect the test probe to the prototype circuit as indicated. Increase the sensitivity of the VOLTS/DIV control by rotating it clockwise until you can see changes on the oscilloscope screen when you talk into the microphone. Adjust TIME/DIV until the shape of the signals is clear. In the space below, make a drawing to represent the V/t graph of an audio signal:



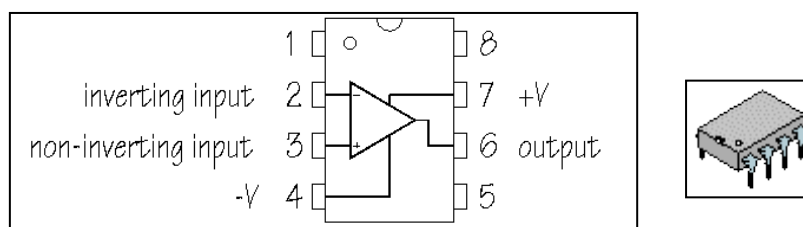
How large is your signal in mV, peak-to-peak amplitude?



What sort of signal is produced if you clap your hands within range of the microphone?

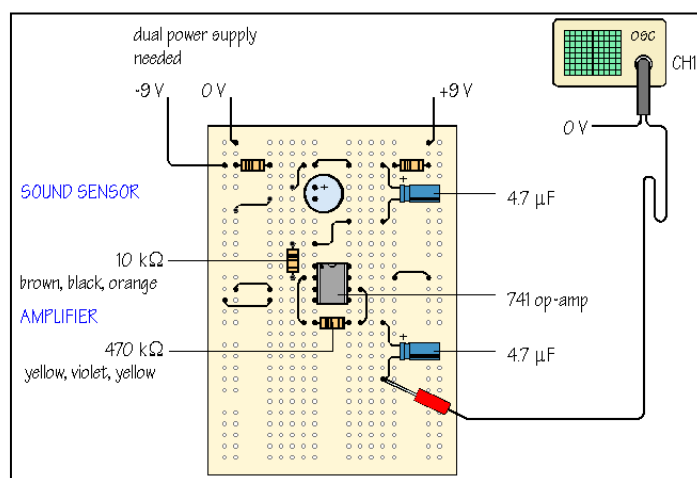


When you talk into the microphone, the signals you get are small. To make them bigger, you need an **amplifier**. One possible circuit is shown below. This uses a 741, one of a large family of integrated circuits called **operational amplifiers**, or **op-amps**:

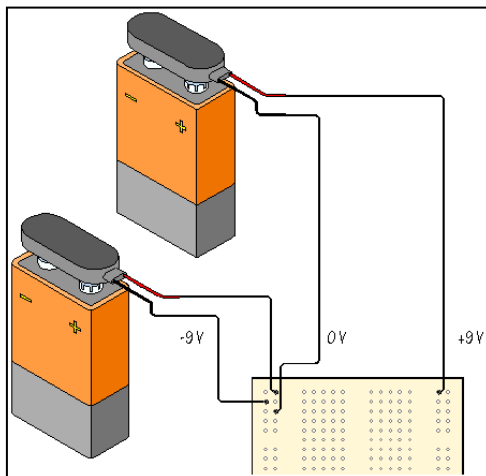


pin connections of a 741 op-amp

The internal circuit of a 741 is quite complicated but it is easy to use the device simply as an amplifying subsystem. It is cheap and easily available. As you can see, the 741 is manufactured in a small plastic package, with 8 connecting pins. These are in a **dual in line**, or **dil** arrangement. With the index mark at the top, pin 1 is on the left and pins are numbered down the left hand side and back up on the right. Often, there is an additional circular mark next to pin 1. This numbering convention is followed on other integrated circuits, whether there are 8, 14, 16, or more pins. Place the 741 across the central gap in the prototype board. Check that pin 1 is correctly located. Now complete the circuit, as follows:



If your power supply does not have **dual power supply** outputs, the +9 V, 0 V, -9 V required can easily be made using two PP3 batteries, connected to the prototype board like this:



If you are unfamiliar with this type of power supply, use a multimeter as a voltmeter, with its black lead connected to 0 V, and touching the positive and negative supply points in turn with the red lead. In one case, the meter will read approximately +9 V, and in the other, approximately -9 V.

Check back with your prototype board and make sure that you have linked the SENSOR subsystem to the AMPLIFIER with a wire link. Monitor the final output of the system using the oscilloscope. How large are your signals now?

The **voltage gain** of the amplifier is given by:

$$\text{voltage gain} = \frac{V_{\text{out}}}{V_{\text{in}}}$$

The way in which this particular op-amp circuit works allows you to choose the voltage gain according to:

$$\text{voltage gain} = -\frac{R_f}{R_{\text{in}}}$$

The minus sign appears because this is an **inverting amplifier** circuit, that is, the output waveform has the same shape as the input waveform, but is turned upside down, or *inverted*, compared with the input waveform. What matters here is that the amplitude of the waveform is increased.

The voltage gain of the circuit is calculated from:

$$\text{voltage gain} = -\frac{470 \text{ k}\Omega}{10 \text{ k}\Omega} = -47$$

V_{out} is inverted and the amplitude of the signal is increased by 47 times. V_{out} after the amplifier should be 47 times larger than the signal from the microphone subsystem. Do your observations using the oscilloscope confirm these changes?

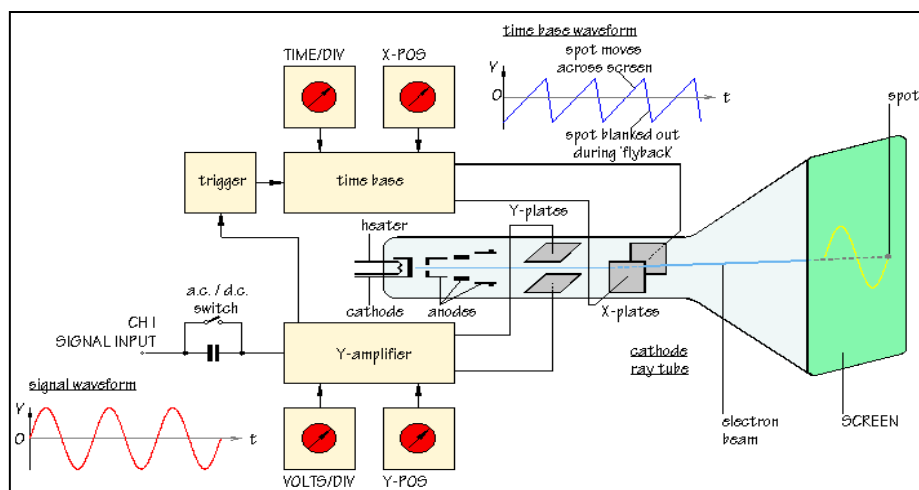


Work through your circuit again using the oscilloscope to monitor the audio signal at different points in the circuit.

You are learning something important here. Developing a circuit is a progressive process. You start with simple subsystems on prototype board and investigate the performance of each subsystem before building the next. There is no point in connecting an amplifier subsystem to a sound sensor which does not work. You need to *know* that the sound sensor is working correctly before building the next stage.

How does an oscilloscope work?

An outline explanation of how an oscilloscope works can be given using the block diagram shown below:



Like a television screen, the screen of an oscilloscope consists of a **cathode ray tube**. Although the size and shape are different, the operating principle is the same. Inside the tube is a vacuum. The electron beam emitted by the heated cathode at the rear end of the tube is accelerated and focused by one or more anodes, and strikes the front of the tube, producing a bright spot on the phosphorescent screen.

The electron beam is bent, or deflected, by voltages applied to two sets of plates fixed in the tube. The horizontal deflection plates, or **X-plates** produce side to side movement. As you can see, they are linked to a system block called the **time base**. This produces a saw tooth waveform. During the rising phase of the sawtooth, the spot is driven at a uniform rate from left to right across the front of the screen. During the falling phase, the electron beam returns rapidly from right to left, but the spot is 'blanked out' so that nothing appears on the screen.

In this way, the time base generates the X-axis of the V/t graph.

The slope of the rising phase varies with the frequency of the sawtooth and can be adjusted, using the TIME/DIV control, to change the scale of the X-axis. Dividing the oscilloscope screen into squares allows the horizontal scale to be expressed in seconds, milliseconds or microseconds per division (s/DIV, ms/DIV, μ s/DIV). Alternatively, if the squares are 1 cm apart, the scale may be given as s/cm, ms/cm or μ s/cm.

The signal to be displayed is connected to the **input**. The AC/DC switch is usually kept in the DC position (switch closed) so that there is a direct connection to the **Y-amplifier**. In the AC position (switch open) a capacitor is placed in the signal path. As will be explained in Chapter 5, the capacitor blocks DC signals but allows AC signals to pass.

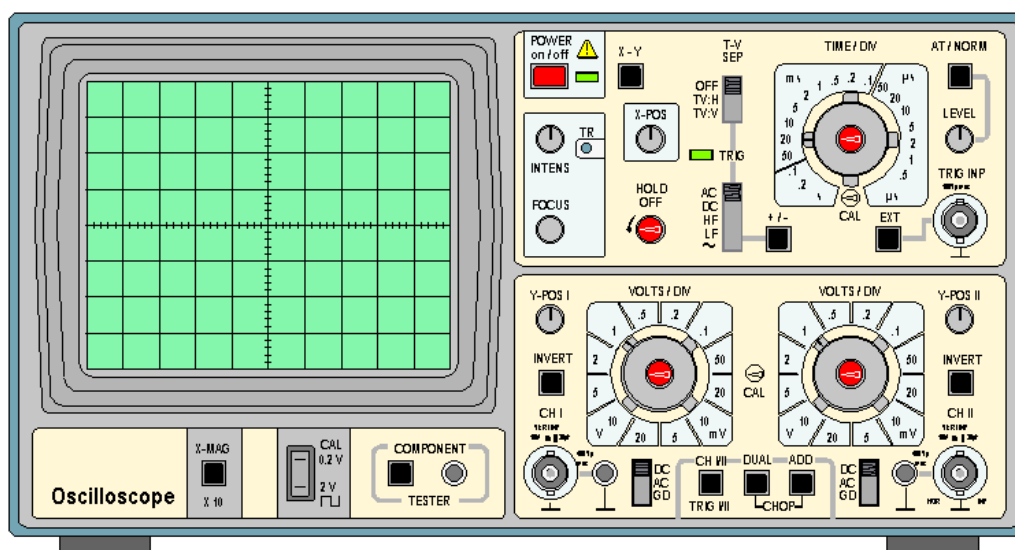
The Y-amplifier is linked in turn to a pair of **Y-plates** so that it provides the Y-axis of the the V/t graph. The overall gain of the Y-amplifier can be adjusted, using the VOLTS/DIV control, so that the resulting display is neither too small or too large, but fits the screen and can be seen clearly. The vertical scale is usually given in V/DIV or mV/DIV.

The **trigger** circuit is used to delay the time base waveform so that the same section of the input signal is displayed on the screen each time the spot moves across. The effect of this is to give a stable picture on the oscilloscope screen, making it easier to measure and interpret the signal.

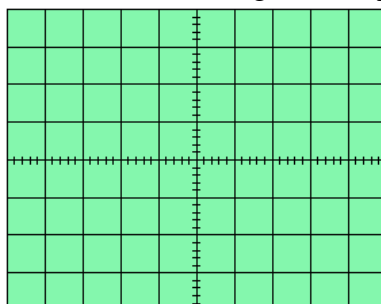
Changing the scales of the X-axis and Y-axis allows many different signals to be displayed. Sometimes, it is also useful to be able to change the *positions* of the axes. This is possible using the X-POS and **Y-POS** controls. For example, with no signal applied, the normal trace is a straight line across the centre of the screen. Adjusting Y-POS allows the zero level on the Y-axis to be changed, moving the whole trace up or down on the screen to give an effective display of signals like pulse waveforms which do not alternate between positive and negative values.

Other oscilloscope controls

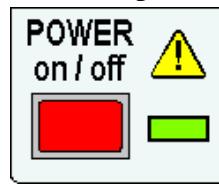
The diagram below is a clickable image map of the *Hameg HM 203-6* oscilloscope. Click on any control to discover its function. Some controls are more useful than others and one or two are rarely if ever used in an introductory electronics course. Click on the small diagram of each control to return to the image map.



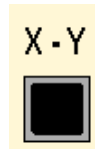
screen: usually displays a V/t graph, with voltage V on the vertical axis and time t on the horizontal axis. The scales of both axes can be changed to display a huge variety of signals.



on/off switch: pushed in to switch the oscilloscope on. The green LED illuminates.



X-Y control: normally in the OUT position.



When the X-Y button is pressed IN, the oscilloscope does not display a V/t graph. Instead, the vertical axis is controlled by the input signal to CH II. This allows the oscilloscope to be used to display a V/V voltage/voltage graph.

The X-Y control is used when you want to display component characteristic curves, or Lissajous figures. (Links to these topics will be added later.)

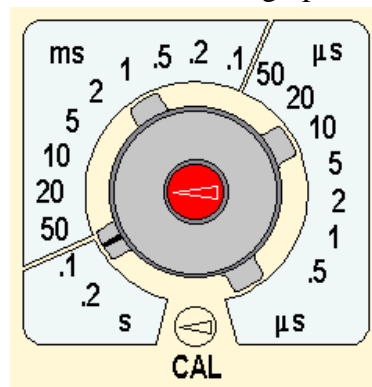
TV-separation: Oscilloscopes are often used to investigate waveforms inside television systems. This control allows the display to be synchronised with the television system so that the signals from different points can be compared.



⚠ MAINS VOLTAGE ⚠

You must **not** try to investigate television systems because of the dangerously high voltages inside. The correct position for this control is OFF.

TIME / DIV: Allows the horizontal scale of the V/t graph to be changed.



trigger controls: This group of controls allows the oscilloscope display to be synchronised with the signal you want to investigate.



When the AT/NORM button is in the OUT position, triggering is automatic. This works for most signals.


If you change the AT/NORM button to its IN position, the most likely result is that the signal will disappear and the oscilloscope screen will be blank. However, if you now adjust the LEVEL control, the display will be reinstated. As you adjust the LEVEL control, the display starts from a different point on the signal waveform. This makes it possible for you to look in detail at any particular part of the waveform.


The EXT button should normally be in its OUT position. When it is pushed IN, triggering occurs from a signal connected to the trigger input, TRIG INP, socket.

The slide switch to the left of TIME/DIV gives additional triggering options. AC is the normal position and is suitable for most waveforms.

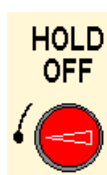
In the DC position, you use the LEVEL control to select a particular DC voltage on the signal waveform where triggering will occur.

The +/- button gives triggering on the upward slope of the signal waveform in the OUT position, and triggering on the downward slope in the IN position.

 **TRIG** The green TRIG LED illuminates when a trigger point is detected.

HF gives triggering in response to high frequency parts of the signal, LF gives triggering for low frequency components and  indicates that triggering will occur at 50 Hz, corresponding to UK mains frequency. You are not likely to need any of these slide switch positions.

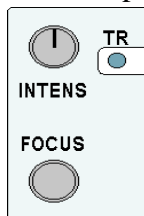
The HOLD OFF control allows you to introduce a delay relative to the trigger point so that a different part of the signal can be seen.



Normally, you will want to leave the HOLD OFF control in its minimum position, as illustrated.

With more experience of using the oscilloscope, you will develop a clear understanding of the functions of the important trigger controls and be able to use them effectively.

intensity and focus: Adjusting the INTENSITY control changes the brightness of the oscilloscope display. The FOCUS should be set to produce a bright clear trace.



If required, TR can be adjusted using a small screwdriver so that the oscilloscope trace is exactly horizontal when no signal is connected.

X-POS: Allows the whole V/t graph to be moved from side to side on the oscilloscope screen.

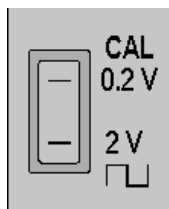


This is useful when you want to use the grid in front of the screen to make measurements, for example, to measure the period of a waveform.

X-MAG: In the IN position, the horizontal scale of the V/t graph is increased by 10 times. For example, if TIME/DIV is set for 1 ms per division and X-MAG is pushed IN, the scale is changed to 0.1 ms per division.

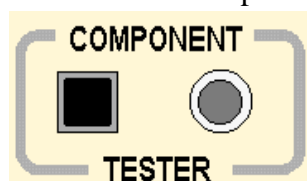


CAL outputs: The top terminal gives a 0.2 V peak to peak square wave, while the lower terminal gives a 2 V peak to peak square wave, both at 50 Hz.



The signals from these outputs are used to confirm that the oscilloscope is correctly calibrated.

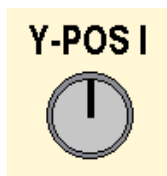
component tester: The output socket provides a changing voltage which allows component characteristic curves to be displayed on the oscilloscope screen.



When the button is IN, the oscilloscope displays a V/V graph, with the component tester voltage connected internally to provide the horizontal axis.

To get normal V/t graph operation the component tester button must be in the OUT position.

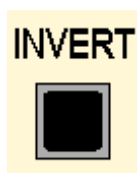
Y-POS I and Y-POS II: These controls allow the corresponding trace to be moved up or down, changing the position representing 0 V on the oscilloscope screen.



To investigate an alternating signal, you adjust Y-POS so that the 0 V level is close to the centre of the screen. For a pulse waveform, it is more useful to have 0 V close to the bottom of the screen.

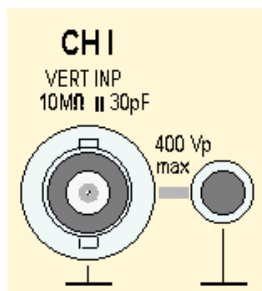
Y-POS I and Y-POS II allow the 0 V levels of the two traces to be adjusted independently.

invert: When the INVERT button is pressed IN, the corresponding signal is turned upside down, or inverted, on the oscilloscope screen.



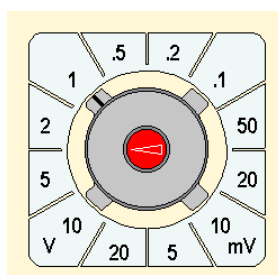
This feature is sometimes useful when comparing signals.

CH I and CH II inputs: Signals are connected to the BNC input sockets using BNC plugs.

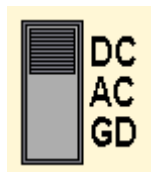


The smaller socket next to the BNC input socket provides an additional 0 V, GROUND or EARTH connection.

VOLTS / DIV: Adjust the vertical scale of the V/t graph. The vertical scales for CH I and CH II can be adjusted independently.



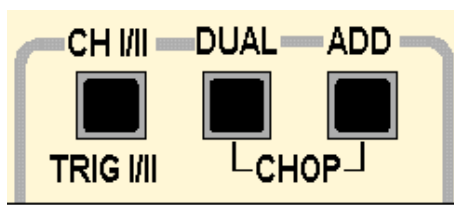
DC/AC/GND slide switches: In the DC position, the signal input is connected directly to the Y-amplifier of the corresponding channel, CH I or CH II. In the AC position, a capacitor is connected into the signal pathway so that DC voltages are blocked and only changing AC signals are displayed.



In the GND position, the input of the Y-amplifier is connected to 0 V. This allows you to check the position of 0 V on the oscilloscope screen.

The DC position of these switches is correct for most signals.

trace selection switches: The settings of these switches control which traces appear on the oscilloscope screen.



The effects of different settings are summarised in the table:

CH I/II	DUAL	ADD	<i>effect of setting</i>
OUT	OUT	OUT	normal operation: only CH I displayed, triggering from CH I
IN	OUT	OUT	only CH II displayed, triggering from CH II
OUT	IN	OUT	CH I and CH II displayed on alternate sweeps, triggering from CH I
IN	IN	OUT	CH I and CH II displayed on alternate sweeps, triggering from CH II
OUT	OUT	IN	CH I and CH II signals added together to produce a single trace, triggering from CH I
IN	OUT	IN	CH I and CH II signals added together to produce a single trace, triggering from CH II
OUT	IN	IN	CH I and CH II displayed simultaneously, triggering from CH I
IN	IN	IN	CH I and CH II displayed simultaneously, triggering from CH II

Settings highlighted in yellow are used frequently. Experience with the oscilloscope will help you to decide which setting is best for a particular application.

For normal operation, all three buttons are in the OUT position.

Appendix B: MultiSim

Step 1. Open Multisim and create a design

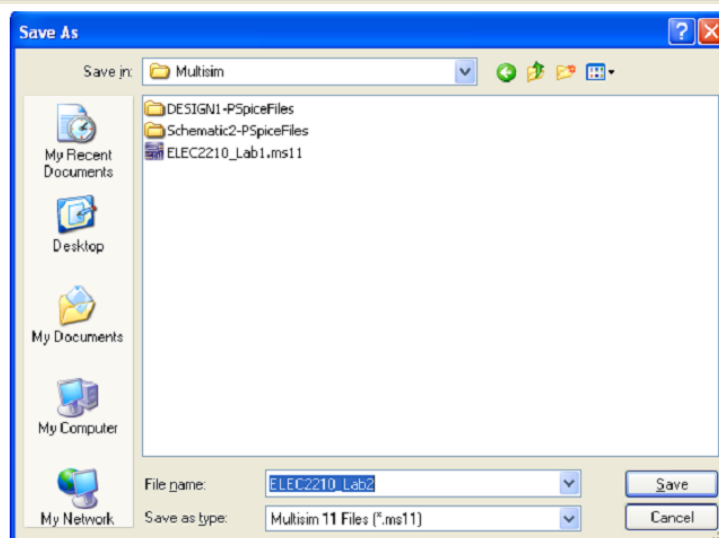
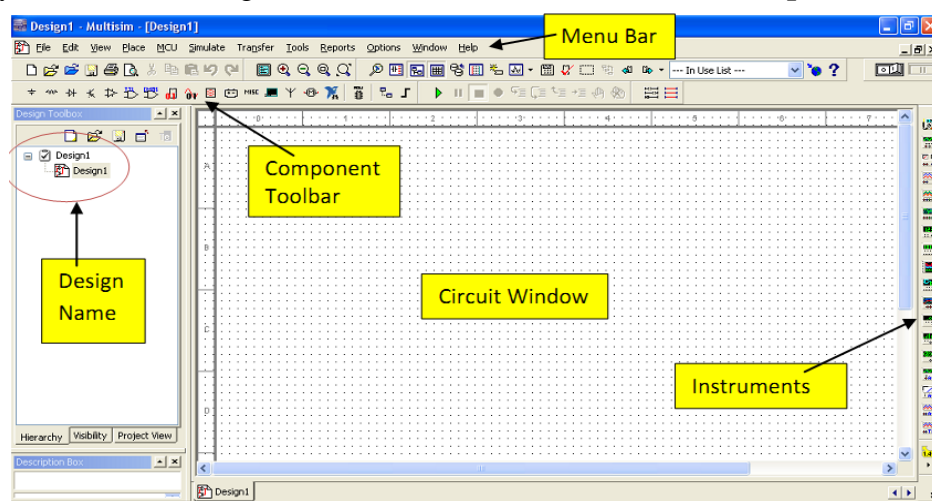
Multisim is opened from the Start Menu:

Start Menu > All Programs > National Instruments > Circuit Design Suite 11.0 > Multisim 11.0
(Version numbers may differ)

This creates a blank design called “Design1”. Save the file with the desired design name via menu bar **File>Save As** to use the standard Windows Save dialog, shown in Figure 2. Navigate to the directory in which you want to save your design, enter the desired file name, and click the Save button. The default file extension for Multisim 11.0 design files is .ms11.

Note: Create a separate directory on a flash drive or your network drive (H:) to be used exclusively for storing your Multisim designs and related files. This will ensure that (1) your files will be accessible from any lab computer and (2) your Multisim files will not be mixed in with your other files. It is also a good idea to periodically make backup copies of your files as protection against loss of data.

A previously created design can be opened via **File>Open**. In the dialog window, navigate to the directory in which the design is stored, select the file, and click the **Open** button.



Step 2. Draw a schematic diagram of the circuit

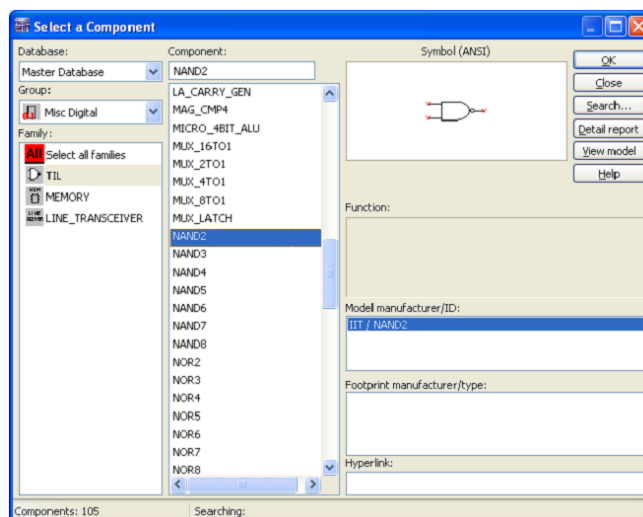
Placing Components: A schematic diagram comprises one or more circuit components, interconnected by wires. Optionally, signal “sources” may be connected to the circuit inputs, and “indicators” to the circuit outputs. Each component is selected from the Multisim library and placed on the drawing sheet in the Circuit Window (also called the Workspace). The Multisim library is organized into “groups” of related components (Transistors, Diodes, Misc Digital, TTL, etc.). Each group comprises one or more “families”, in which the components are implemented with a common technology. For designing and simulating digital logic circuits in ELEC 2210, two groups are to be used: “Misc Digital” (TIL family only) and “TTL”.

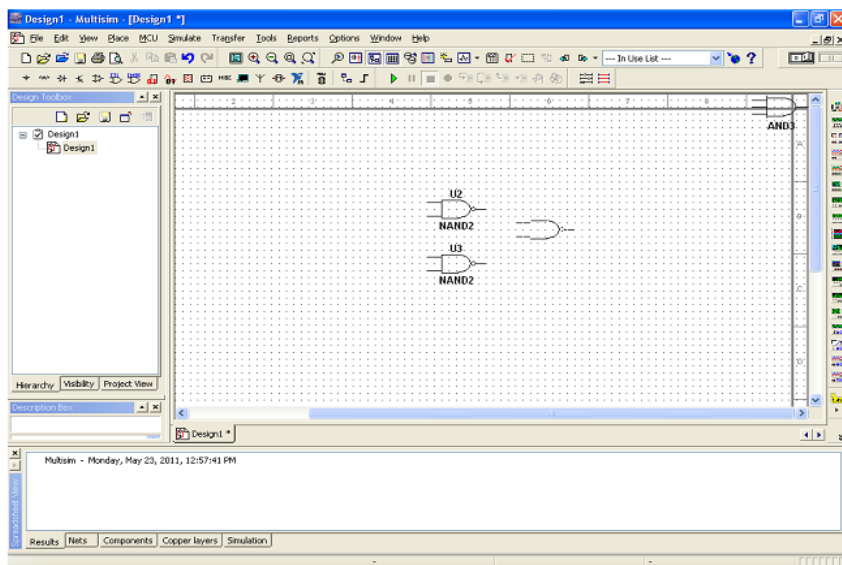
The “Misc Digital” group has three families of components, of which family “TIL” contains models of generic logic gates, flip-flops, and modular functions. These components are technology-independent, which means that they have only nominal circuit delays and power dissipation, unrelated to any particular technology. Generic components can be used to test the basic functionality of a design, whereas realistic timing information requires the use of technology-specific part models, such as those in the TTL group.

To place a component on the drawing sheet, select it via the Component Browser, which is opened via the component toolbar or the menu bar. From the menu bar, select **Place>Component** to open the Component Browser window, illustrated in Figure 3. You can also open this window by clicking on the **Misc Digital** icon in the component toolbar. On the left side of the window, select “Master Database”, group “Misc Digital”, and family “TIL”.

The component panel in the center lists all components in the selected family. Scroll down to and click on the desired gate (NAND2 in Figure 3); its symbol and description are displayed on the right side of the window. Then click the OK button. The selected gate will be shown on the drawing sheet next to the cursor; move the cursor to position the gate at the desired location, and then click to fix the position of the component. The component can later be moved to a different location, deleted, rotated, etc. by right clicking on the component and select the desired action. You may also select these operations via the menu bar Edit menu.

After a component has been positioned, the component browser is redisplayed and additional components can be placed by repeating the above actions. When the last component has been placed, click the **Close** button to close the component selection window. You may return to the component browser at any time to add additional components.

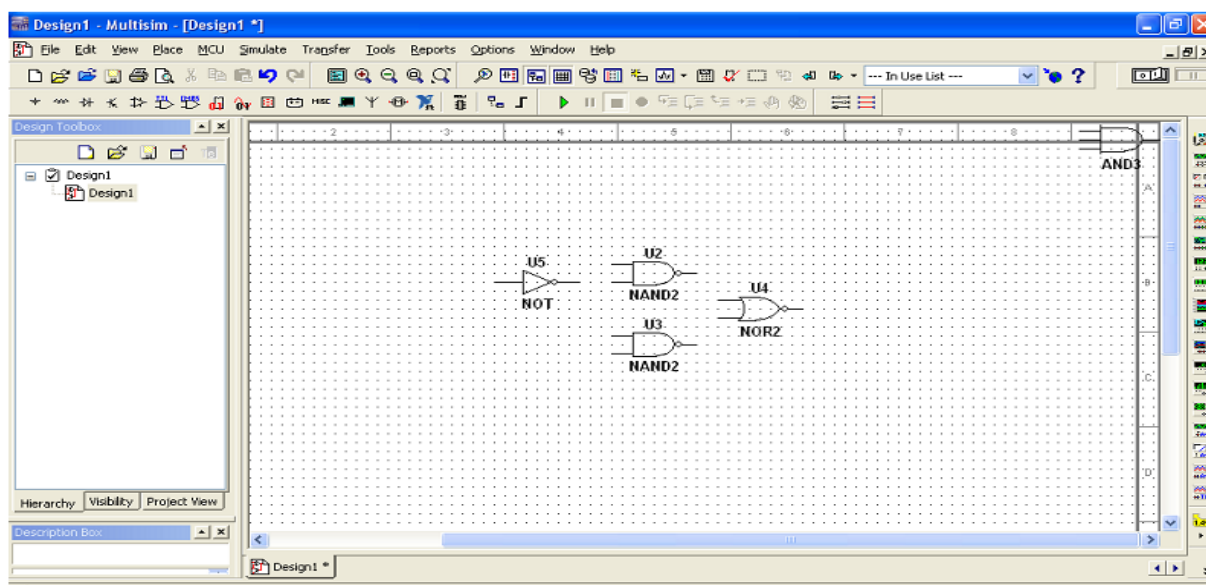




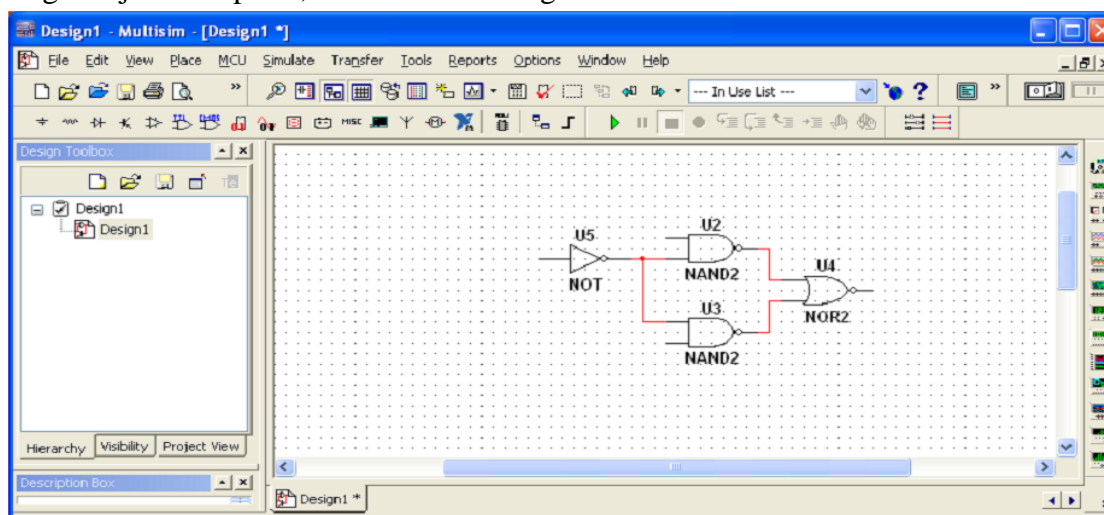
The **TIL** family part naming convention is as follows. Look at the symbol and function panels on the right side of the component selection window to determine the specific function of a selected component.

1. Basic logic gates: **ANDx**, **NANDx**, **NOT**, **NORx**, **ORx**, **XNORx**, **XORx**, where x is the number of gate inputs (2-8).
2. Flip flops: **D_FF**, **JK_FF**, **SR_FF**, **T_FF**, and latches **D_LATCH**, **SR_LATCH**, plus versions that have active-high or active-low asynchronous Set and Reset pins (ex. **D_FF_POSSR** and **D_FF_NEGSR**, respectively).
3. Standard digital modules: multiplexers, decoders, encoders, counters, ALU, BCD-to-7 segment decoder, registers, shift registers, etc.
4. **DIGITAL_PULLUP**, **DIGITAL_PULLDOWN** – to pull a pin to a constant logic 1 and 0, respectively.

Figure 5 shows the schematic diagram with four placed components. Note that each placed gate has a “designator” (U2, U3, U4, U5), which can be used when referring to that gate. You can change a designator by right clicking on the component, selecting Properties, and entering the desired name on the Label tab.



Drawing Wires: Wires are drawn between component pins to interconnect them. Moving the cursor over a component pin changes the pointer to a crosshair, at which time you may click to initiate a wire from that pin. This causes a wire to appear, connected to the pin and the cursor. Move the cursor to the corresponding pin of the second component (the wire follows the cursor) and click to terminate the wire on that pin. If you do not like the path selected for the wire, you may click at a point on the drawing sheet to fix the wire to that point and then you can move the cursor to continue the wire from that point. You may also initiate or terminate a wire by clicking in the middle of a wire segment, creating a “junction” at that point. This is necessary when a wire is to be fanned out to more than one component input. A partially-wired circuit, including one junction point, is illustrated in Figure 6.

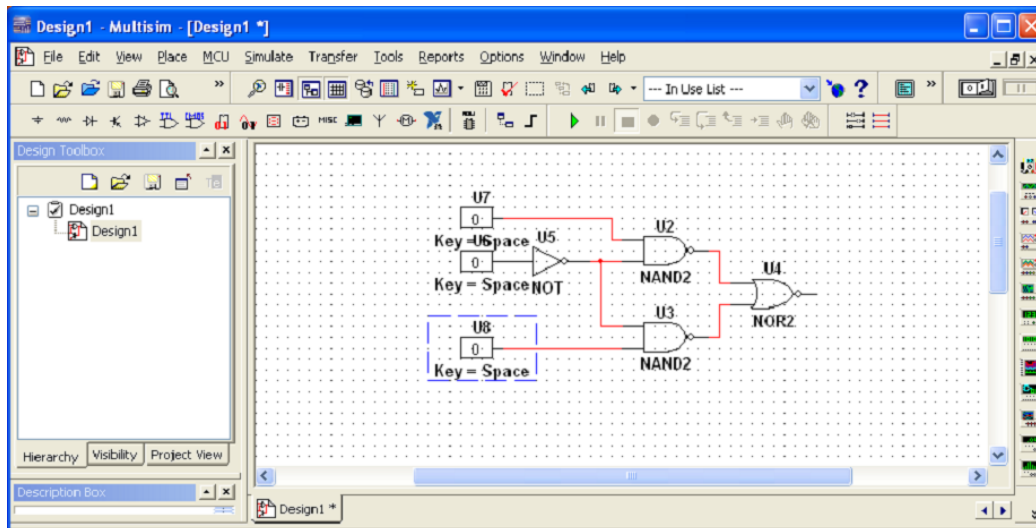


Step 3. Generating test input patterns.

To drive circuit simulations, Multisim provides several types of “sources” to generate and apply patterns of logic values to digital circuit inputs. Sources are placed on the schematic sheet and connected to circuit inputs in the same way as circuit components, selecting them from the “Digital_Sources” family of the “Sources” group in the component browser. Note that there is a **Place Source** shortcut icon in the tool bar. There are three basic digital sources:

1. **DIGITAL_CONSTANT** – this is a box with a constant logic 1 or 0 output, and would be used where the logic values is not to be changed during simulation. To change the output value, right click on the box, select Properties, select the desired value on the Value tab, and click the OK button.
2. **INTERACTIVE_DIGITAL_CONSTANT** – this is a clickable box that can be connected to a circuit input. Clicking on the box toggles its output between 0 and 1. This can be used to interactively change a circuit input during simulation.
3. **DIGITAL_CLOCK** – this is a box that produces a repeating pulse train (square waveform), oscillating between 0 and 1 at a specified frequency. To set the frequency and duty cycle, right click on the box, select Properties, select the desired frequency and duty cycle value on the Value tab, and click the OK button.

Figure 7 shows the circuit of Figure 6 with an **INTERACTIVE_DIGITAL_CONSTANT** connected to each input. Note that the initial state of each is logic 0. Since this circuit has only three inputs, all 8 input patterns can be produced (to generate a truth table for the circuit) by manually toggling the inputs.



Another mechanism to apply all possible input combinations to a circuit is to connect a DIGITAL_CLOCK to each input, with the frequency of the first clock set to some value N , the second to $2N$, the third to $4N$, etc. This produces the pattern shown in Figure 7. Looking at these waveforms, it can be seen that the initial pattern is 000, changing to 001, 010, etc.

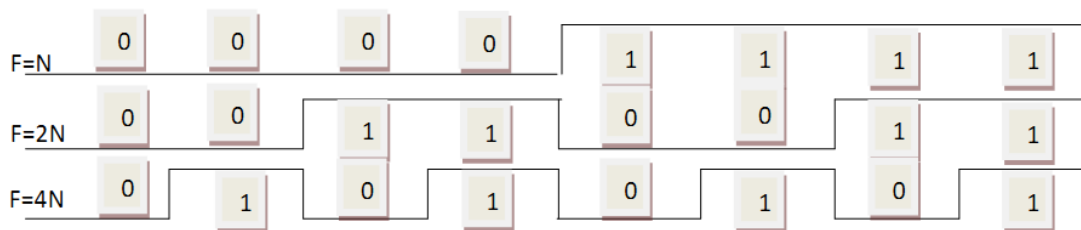
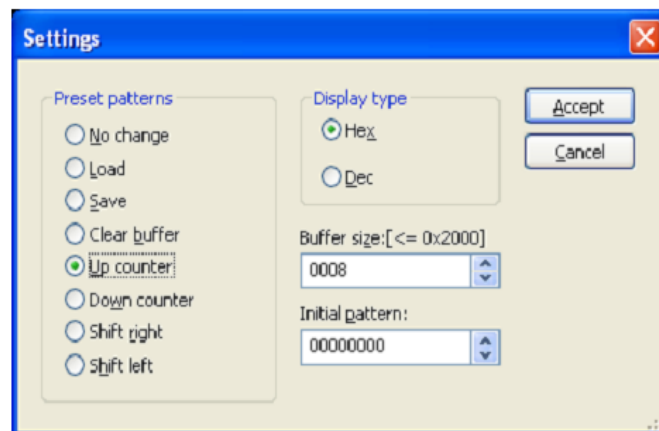
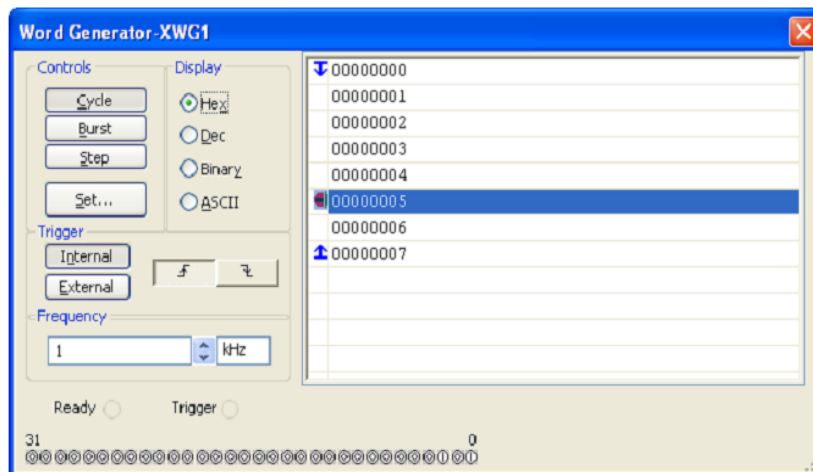
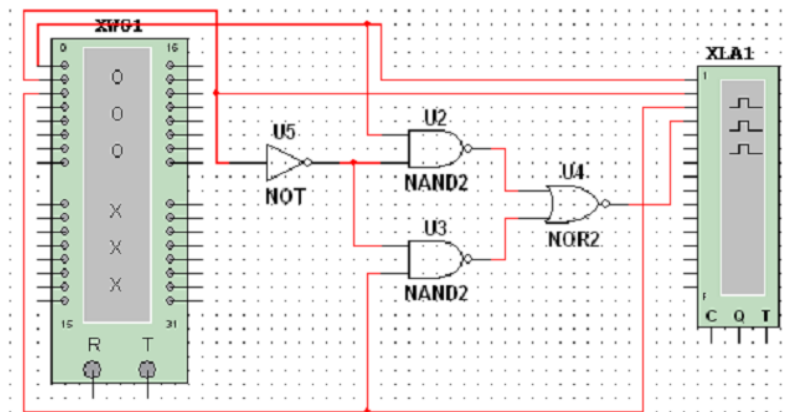


Figure 7. Three clocks used to produce a sequence of 8 digital patterns

Of course, DIGITAL_CLOCK sources would also be used to drive the clock inputs of sequential circuits.

One additional mechanism for generating digital patterns is the “Word Generator” instrument, which can be added to the circuit via the shortcut icon on the right side of the main window, or via the menu bar **Simulate>Instruments>Word Generator**. Figure 8 shows a Word Generator (XWG1) connected to the three circuit inputs. The Word Generator produces a sequence of patterns, each containing from 1 to 32 bits.

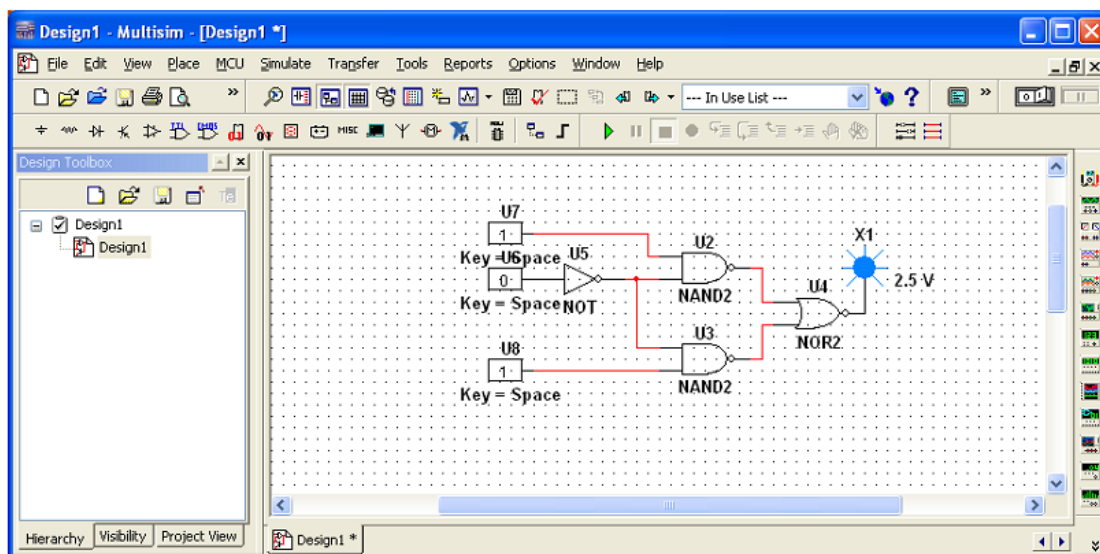
To specify the patterns and the rate at which they should be produced, double click on the Word Generator symbol, producing the window shown in Figure 9. The Control buttons on the left of this window allow patterns (1) to be continuously applied and repeated (Cycle button), (2) a single set of patterns to be applied (Burst button), or (3) a single pattern to be applied (Step button). The patterns displayed in this window will be applied in the order listed, and can be entered manually or generated automatically. For automatically-generated patterns, press the Set button, producing the Settings window of Figure 10. In this example, “Up Counter” is selected to produce a sequence of binary numbers, “Buffer Size” is set to 8 to limit the sequence to 8 numbers, and “Initial pattern” is set to 0. Note that the 8 binary numbers are displayed in the Word Generator window of Figure 9. The rate at which the patterns are to be applied to the circuit is specified via the Frequency box of the Word Generator window. In Figure 9, the frequency has been set to 1KHz, which means that 1000 patterns will be generated each second, or one pattern every 1msec.



Step 4. Connect circuit outputs to indicators

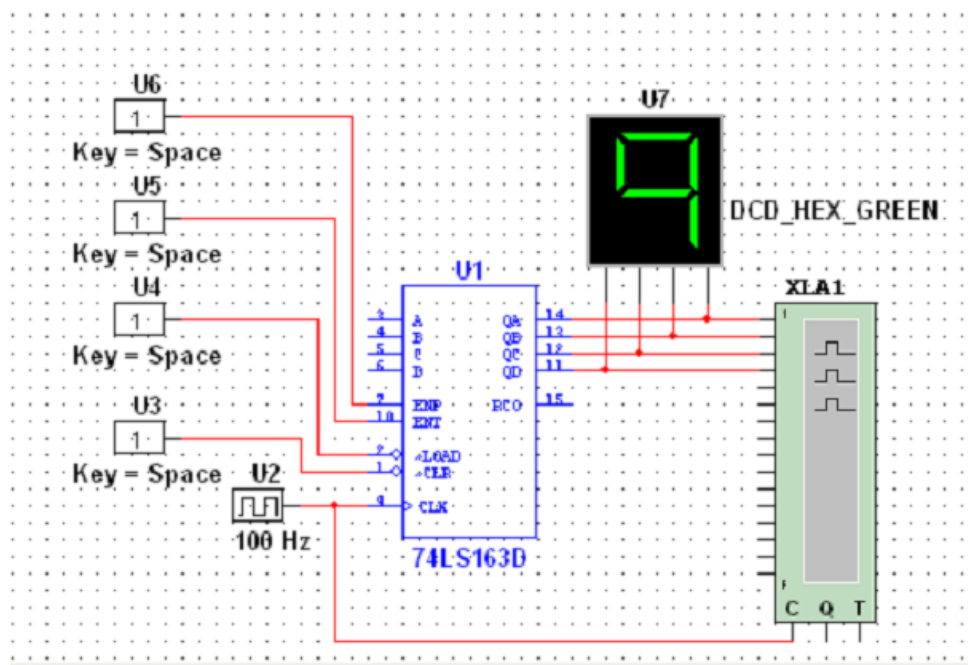
To facilitate studying the digital circuit output(s), Multisim provides a variety of “indicators”. For digital simulation, the most useful are digital “probes”, hex displays, and the Logic Analyzer instrument. A probe, illustrated in Figure 11, displays a single digital value as ON or OFF (the probe is “illuminated” in Figure 11, indicating an ON condition). The PROBE family of the Indicators group includes a generic **PROBE_DIG** and several **PROBE_DIG_color** indicators (color = **BLUE**, **GREEN**, **ORANGE**, **RED**, **YELLOW**).

The probe in Figure 11 is **PROBE_DIG_BLUE**. This circuit can be verified by manually changing the three **INTERACTIVE_DIGITAL_CONSTANT** inputs to each of the 8 possible combinations, and recording the probe value for each combination to create a truth table.



The **HEX_DISPLAY** family contains a variety of displays that show multi-bit values as hexadecimal numbers. This would be useful for such circuits as binary counters, as illustrated in Figure 12.

In this example, the **DCD_HEX_GREEN** indicator is a “decoded” hex display digit, which means that the indicator displays the hex number corresponding to the states of its four inputs.

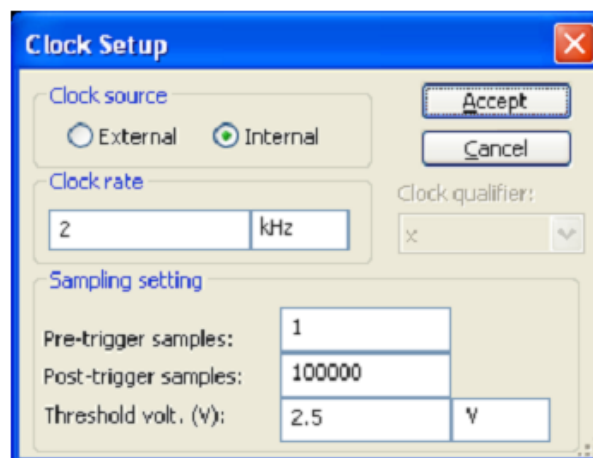
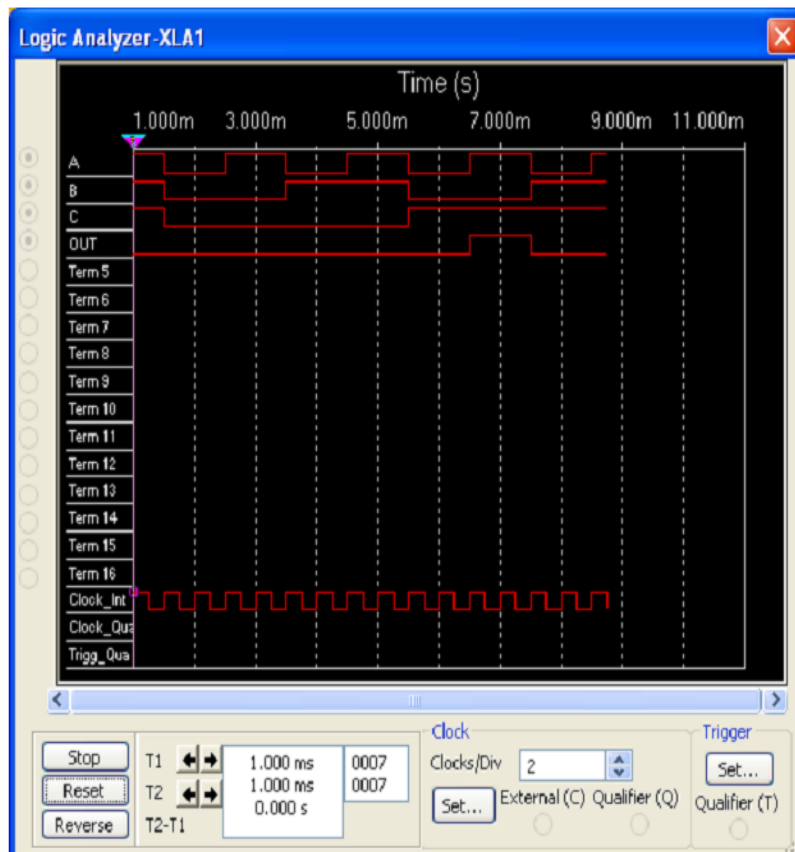


Another useful indicator is the Logic Analyzer instrument, shown in Figures 8 and 12. A logic analyzer is an instrument that captures and displays sequences of digital values over time, with the sequences displayed as waveforms rather than as tables of numbers.

The Multisim Logic Analyzer instrument can capture and display up to 16 signals. Samples are triggered either by an internal clock (N samples per second) or by an external clock.

Figure 13 shows the Logic Analyzer display of the three inputs and the output of the circuit in Figure 8, showing the response of the circuit to the burst of Word Generator values shown in Figure 9, plus the internal clock that was used to capture samples.

The logic analyzer must be configured to capture values at the correct sampling times. This is done by clicking on the Set button in the Clock area under the logic analyzer display, opening the Clock Setup window shown in Figure 14. Since the word generator was configured to generate patterns at a rate of 1KHz, a clock rate of 2KHz was set for the logic analyzer to make it capture two values for each input pattern.



Step 5. Run the simulation

A simulation is initiated by pressing the Run (green arrow) button in the toolbar or via the menu bar via **Simulate>Run**. Alternatively, simulation can be initiated from a Word Generator by pressing the Cycle, Burst, or Step buttons.

You may capture any window and paste it into a Word or other document for generating reports. An individual window is captured by pressing the ALT and Print Screen keys concurrently. You may then “paste” the captured window into a document via the editing features of that document. To capture a circuit diagram in the main window, the simplest method is via the menu bar **Tools>Capture** Screen Area. This produces a rectangle whose corners can be stretched to include the screen area to be captured; the “copy” icon on the top left corner is pressed to copy the area, which may then be pasted into a document. The circuit images in Figures 8 and 12 were copied in this manner.

Step 6. Save the design and close Multisim

The simplest way to save a design is to click the Save icon in the Design Toolbar on the left side of the window, directly above the design name. Alternatively, you may use the standard menu bar **File>Save**. As mentioned earlier, you should save all designs in a special course directory on either your network (H:) drive or on a flash memory device. Multisim is exited as any other Windows program.

SOLUTIONS

Experiment 1

1.1. Serial monitor | MDA8086 kit 9.5

WWW.MIDASENG.COM

1.2. PC

| 0A | 1234

3.10. R IP

3.11. R AX

Experiment 2

Exercises

Q-1:

char, octal decimal and binary number. It provides 2 modes: 8-bits & 6-bits. It can convert a number/character from any system to the others.

Experiment 3

Experimental Work

Result 1: AX= 002B BX= 390B FL= 0004 (PF=1)

BL ; BL= BL - 1 and save the result in BL.

- **CBW**

X+Y+W-V+U

MOV DI,6h ; a=6h

MOV DH,7h ; b=7h

Experiment 4

Experimental Work

Result.1: opened.

Result.2:

```
MOV AX, 01234H
MOV
,[BX+SI+2H]
INC BYTE PTR[1H]
IN AL,02H
IN AL,DX
TABLE DB 20H, 30H, 40H, 50H
MOV BX, OFFSET TABLE
MOV AX, SEG TABLE
JMP START
```

```
ret
```


Experiment 5

Experimental work

```
org 100h
mov ax,0          ;2
mov ss,ax
mov sp,0540h
mov ax,1234h      ;3
mov bx,5678h
mov cx,3abch
mov dx,0a0bh
call
```

and that set ZF & PF. IF is set to response to interruption.

Result.2:

Table 1: Instructions used in the program

a=(REG/MEM) and b=(REG/MEM/IMMEDIATE0

Experiment 6

Result & Discussion

```
CODE SEGMENT
ASSUME    CS:CODE,DS:CODE,ES:CODE,SS:CODE
SOFWARE_INT_1 EQU 0021H      ;2
VAL_2 EQU 0000H
VAL
```

CS:IP with address of subroutine while INT does what call does and push flag register.

ISR_1 will execute when we did the interrupt INT SOFRWARE_INT_1.

Experiment 7

Display the time program:

```
org 100H
mov dx, offset msg
mov ah,9
int 21H
TIME
i db 0,10,'il est maintenant $'
```

Experiment 8

Experiment Exercises

Q-1: If we set the 8255 control register to 10001111B, the chip is active and group A is set to mode 0. Port

OP TIMER2

JMP START

INT 3

CODE ENDS

END

Experiment 9

CODE SEGMENT

ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE

;

instruction to return to label AGAIN and repeat the program.

JMP AGAIN

Experiment 10

Full program:

```
CODE SEGMENT
ASSUME CS:CODE,DS:CODE,ES:CODE,SS:CODE
CTRL EQU 1FH ;2 definitions;DEFINE CONTROL WORD
PPIB EQU
```

. More bar level turned on indicates more voltage applies across (analog scale). DAC and ADC are very important because the real world data is analog.

Experiment 11

- 1-the cursor is blinking
- 2- LCDC
- 11-line 1 & 2: 107F & 1091
- 12-