****

**LAB RECORD**

**BACHELOR OF TECHNOLOGY**

**B.Tech. CS&E Semester (3)**

**(Academic Session : 2021-2025)**

**Course Title : Digital Electronics And Computer Organization**

**Course Code : CSE207**

**Enrollment No. : A7605221191**

**Name of Student : AMAN GUPTA**

**Date of Submission :**

**Signature of Student :**

**Grade/Marks Obtained :**

**Faculty Name & Signature: Dr. Atul Srivastava(Assistant Professor)**

**Department of Computer Science & Engineering**

**Amity School of Engineering & Technology**

**Amity University, Lucknow Campus**

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Lab Objective** | **Date** | **Signature** |
| 1. | To Stimulate Half Adder Circuit. |  |  |
| 2. | Design to verify full adder circuit using OR-CAD and Pspice. |  |  |
| 3. | To simulate and verify the Half Subtractor and the Full Subtractor. | **Data Structures Using C** | **FFFFFF**  **FFFFFF** |
| 4. | Introduction to 8085 Microprocessor |  |  |
| 5. | Program to add and subtract using 8085 Microprocessor. |  |  |
| 6. | To simulate logical part of the Arithmetic Logic Unit |  |  |
| 7. | To simulate a 4-bit adder/subtractor circuit using ORCAD |  |  |
| 8. | To simulate and study tri-state buffer. |  |  |
| 9. | To simulate common bus with multiplexers. |  |  |
| 10. | To simulate a 4-bit adder/subtractor circuit using ORCAD. |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Open Ended Lab Objective** | **Date** | **Signature** |
| 1. | To make 16:1 MUX using 4:1 MUX |  |  |
| 2. | To simulate and verify the Boolean expressions using ORCAD and Pspice. |  |  |

**INTRODUCTION TO DECO**

In the Current world of electronics, the term Digital is typically linked with a computer since the name Digital is derived from the way computers do function, by counting numbers. For many years, digital electronics was used only in Computer systems. However, digital electronics is currently employed in a wide range of applications. The following are a few examples of when digital electronics is employed extensively.

* Industrial process control
* Television
* Communication system
* Medical equipment
* Radar
* Navigation

**Digital Signal**

A digital signal is one with a limited number of unique values. Digital transmissions aren't always continuous. Switches are used to provide input in the digital electronic calculator. This information is transformed into an electrical signal with two distinct levels or values. One of these is referred to as low level, while the other is referred to as high level. One of the two levels will always be present in the signal. The term "digital signal" refers to this sort of signal. The following are some examples of digital signals.

* Binary Signal
* Octal Signal
* Hexadecimal Signal

**Introduction to ORCAD**

OR-CAD is a closed-source software tool package primarily used for electronic design automation (EDA). Electronic design professionals and electronic technicians mostly use the programme to produce electronic schematics and electronic prints for producing printed circuit boards.

It is developed in C and C++ and runs on the Microsoft Windows operating system. OrCAD is a portmanteau that reflects the roots of the firm and its software: CAD + Oregon. OrCAD is a PCB design and analysis software package that comprises a schematic editor (Capture), an analog/mixed-signal circuit simulator (PSpice), and a PCB board layout solution (PCB Designer Professional).

**ORCAD capture**

OrCAD Capture is a schematic capture programme that is included in the OrCAD circuit design package. Capture, unlike NI Multisim, does not have built-in simulation facilities; instead, it outputs netlist data to the simulator, OrCAD EE. To create circuit boards, Capture may export Verilog or VHDL hardware descriptions of the circuit schematic, and netlists to circuit board designers such as OrCAD Layout and Allegro, among others.

A component information system (CIS) is included in Capture, which links component package footprint data or simulation behavior data to the circuit symbol in the design. Customers may share and sell add-ons and design materials on the OrCAD Capture Marketplace. Provide-ons like these may be used to modify the design environment and add new features and capabilities. Customers may share and sell add-ons and design materials on the OrCAD Capture Marketplace. These add-ons may be used to personalize the design environment and add new features and functions. Capture may connect to any database that follows Microsoft's ODBC standard, for example. Data from an MRP, ERP, or PDM system may be accessed directly and used in the component decision-making process.

**Introduction to PSPICE**

OrCAD EE PSpice is a SPICE circuit simulator application for analogue and mixed-signal circuit modelling and verification. Personal Simulation Program with Integrated Circuit Emphasis (PSpice) is an abbreviation for Personal Simulation Program with Integrated Circuit Emphasis.

OrCAD EE is used to conduct simulations for circuits created in OrCAD Capture, and it can also interface with MATLAB/Simulink via the Simulink to PSpice Interface (SLPS). With schematic entry, native analogue, mixed signal, and analytic engines, OrCAD Capture and PSpice Designer provide a full circuit simulation and verification solution.

Micro-sim commercialized PSpice in 1984, which was a modified version of the academically created SPICE. OrCAD bought Micro-sim ten years later, in 1998. PSpice Designer and PSpice Designer Plus are two different versions of OrCAD PSpice Designer.

OrCAD Capture and OrCAD PSpice solution are included in OrCAD PSpice Designer. The PSpice Advanced Analysis simulation engine is available as an upgrade option for PSpice Designer Plus, allowing for functional simulation and increased design yield and dependability.

OrCAD EE is a more advanced version of the PSpice simulator that features automated circuit optimization as well as waveform recording, visualization, analysis, curve fitting, and post-processing. OrCAD EE has a large collection of physical component models, including over 33,000 analogue and mixed-signal devices as well as mathematical functions. OrCAD EE also comes with a model editor, parameterized model support, auto-convergence and checkpoint restart, numerous internal solvers, and a magnetic component editor, among other features.

**INTRODUCTION TO INTEGRATED CIRCUIT**

In 1952, Geoffrey W. A. Dummer presented the concept of an integrated circuit for the first time. However, the endeavor to construct it was unsuccessful. Jack Kilby came up with another idea. He came up with the notion of making tiny ceramic wafers with a small discrete component on each one. After that, all of these wafers may be connected together to make a compact circuit. However, although being created for the US army, this concept failed to gain traction and was abandoned.

In the year 1958, the notion of IC was initially established. Since then, this Conception has advanced to greater technical heights than any other concept, allowing for the downsizing of numerous components in the digital world, including mobile phones, computers, laptops, and so many more.

The invention of vacuum tubes ushered in the digital era. Vacuum-based computers were uncommon and costly. Transistors eventually took their place, as they were faster to operate and smaller in size, as well as cheaper, less power hungry, and more dependable. Then came the discovery of integrated circuits, which completely changed how computers were used. Even the average person is familiar with its uses, such as smart phones and computers, because to its compact size, low cost, and excellent dependability.

Due to two causes, discrete circuits were phased out in favor of integrated circuits. The first is the use of space. Transistors, resistors, diodes, capacitors, and other discrete components make up discrete circuitry. According to the circuitry requirements, each of them is soldered on to printed circuit boards (PCB). In the end, the PCB will take up a lot of room. Another disadvantage is that due to the large number of components used, the soldered components will be less reliable. Engineers were compelled to create microcircuits that were more reliable and took up less space as a result of both of these reasons.

Due to their excellent dependability and small size, the ICs have also found use military applications, cutting-edge communication systems, and industrial applications. A modern integrated circuit (IC) the size of a fingernail contains million transistors and other discrete components. An integrated circuit, often known as a microchip, is a tiny chip composed of a semiconductor material such as silicon that contains a collection of discrete circuits.

**Advantages of Integrated Circuits**

* Miniature in stature. The IC shrinks dramatically as a result of the production method used to integrate active and passive components on a silicon chip. It May be at least a thousand times smaller than a discrete circuit.
* It requires more effort and money to build hundreds of discrete circuits on a PCB for the same logic. However, the cost of manufacturing hundreds of ICs will be extremely inexpensive and will take much less time.
* If a single transistor fails in a discrete circuit, the entire circuit may cease to function. This transistor requires desoldering and replacement. It's tough to figure out which part of the system has failed. Because changing a whole IC is inexpensive, this problem can be avoided with an IC.
* Because all of the components in an IC are manufactured relatively near to one another, they are ideal for tiny signal operation because there will be no stray electrical pickup. There will be no exterior projections because all of the components are manufactured inside the chip.

**Revision Program**

**Objective:** Design and verify basic, universal and exclusive logic gates using ORCAD and Pspice.

**Software Used:** ORCAD, Pspice.

|  |  |  |
| --- | --- | --- |
| **S.No.** | **LOGIC GATE** | **IC NUMBER** |
| 1. | AND gate | IC7408 |
| 2. | OR gate | IC7432 |
| 3. | NOT gate | IC7404 |
| 4. | NAND gate | IC7400 |
| 5. | NOR gate | IC7402 |
| 6. | XOR gate | IC7486 |
| 7. | XNOR gate | IC (7486+7404) |

**Theory:** A logic gate is an idealistic model of computing or a real electronic device that implements a Boolean function, which is a logical operation that creates a single binary output from one or more binary inputs. The phrase may refer to an ideal logic gate, such as one with zero rise time and limitless fan-out, or it may refer to a non-perfect physical device, depending on the context.

Electronic logic gates are not the same as their relay-and-switch counterparts. They're a lot quicker, use a lot less electricity, and are a lot smaller (all by a factor of a million or more in most cases). In addition, there is a structural difference. Between its input and output, the switch circuit creates a continuous metallic route for electricity to travel (in either direction). The semiconductor logic gate, on the other hand, functions as a high-gain voltage amplifier that sinks a little current at its input and outputs a low-impedance voltage. A semiconductor logic gate's output and input are not connected; thus, current cannot flow between them.

The ability to cascade standardized integrated circuit logic families, such as the 7400 and 4000, is another significant benefit. This means that one gate's output can be connected to the inputs of one or more other gates, and so on. Systems of various complexity may be constructed without the designer having to be Concerned about the internal workings of the gates, as long as the constraints of each integrated circuit are taken into account.

**LOGIC SYMBOLS**

**AND gate (IC7408)**

The AND gate is a basic digital logic gates that implements logical combination from mathematical logic — it behaves according to the truth table below. A HIGH output (1) results only if all the inputs to the AND gate are HIGH (1). If none or not all inputs of the AND gate are HIGH, LOW output results. The function can be extended to any number of inputs.

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **OUT** |
| **0** | **0** | **0** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **1** |

**A**

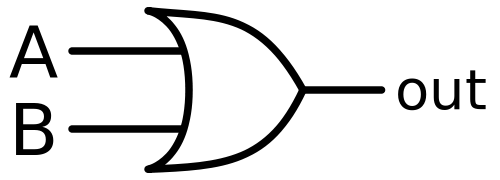
**B**

**OUT**

**OR gate (IC7432)**

The OR gate is a digital logic gate that implements logical disjunction (V) from mathematical logic — it behaves according to the truth table above. A HIGH output (1) results if one or both the inputs to the gate are HIGH (1). If neither input is high, a LOW output (0) results. In another sense, the function of OR effectively finds the maximum between two binary digits, just as the complementary AND function finds the minimum.

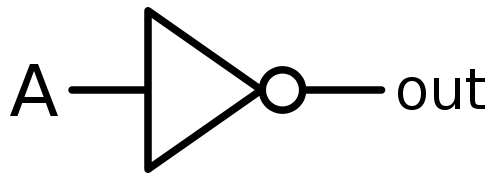
|  |  |  |
| --- | --- | --- |
| **A** | **B** | **OUT** |
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **1** |

****

**NOT gate (IC7404)**

In digital logic, an inverter or NOT gate is a logic gate which implements logical negation. In mathematical logic it is equivalent to the logical negation operator.

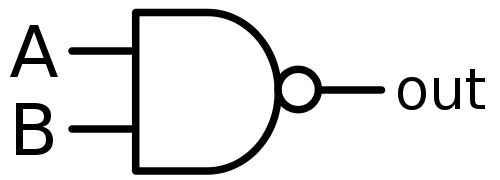
|  |  |
| --- | --- |
| **A** | **OUT** |
| **0** | **1** |
| **1** | **0** |



**NAND gate (IC7400)**

In digital electronics, a NAND gate (NOT-AND) is a logic gate which produces an output which is false only if all its inputs are true; thus, its output is complement to that of an AND gate. A LOW (0) output results only if all the inputs to the gate are HIGH (1); if any input is LOW (0), a HIGH (1) output results. A NAND gate is made using transistors and junction diodes.

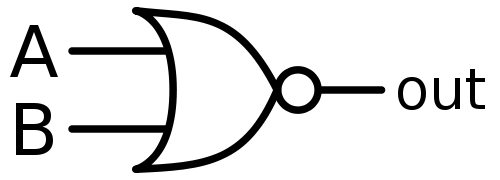
|  |  |  |
| --- | --- | --- |
| **A** | **B** | **OUT** |
| **0** | **0** | **1** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |



**NOR gate (IC7402)**

The NOR gate is a digital logic gate that implements logical NOR - it behaves according to the truth table to the right. A HIGH output (1) results if both the inputs to the gate are LOW (0); if one or both input is HIGH (1), a LOW output (0) results. NOR is the result of the negation of the OR operator. It can also in some senses be seen as the inverse of an AND. gate. NOR is a functionally complete operation—NOR gates can be combined to general any other logical function. It shares this property with the NAND gate. By contrast, the OR operator is monotonic as it can only change LOW to HIGH but not vice versa.

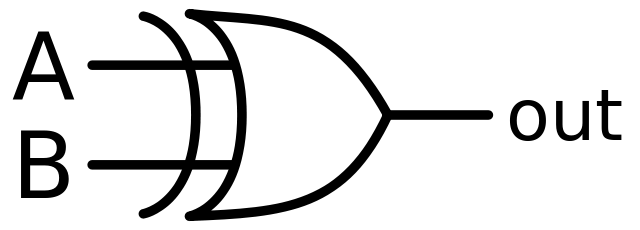
|  |  |  |
| --- | --- | --- |
| **A** | **B** | **OUT** |
| **0** | **0** | **1** |
| **0** | **1** | **0** |
| **1** | **0** | **0** |
| **1** | **1** | **0** |



**XOR gate (IC7486)**

XOR gate (sometimes EOR, or EXOR and pronounced as Exclusive OR) is a digital log gate that gives a true (1 or HIGH) output when the number of true inputs is odd. An XOR gate implements an exclusive or from mathematical logic; that is, a true output results one, and only one, of the inputs to the gate is true. If both inputs are false (0/LOW) or bot are true, a false output result. XOR represents the inequality function, i.e., the output is true if the inputs are not alike otherwise the output is false.

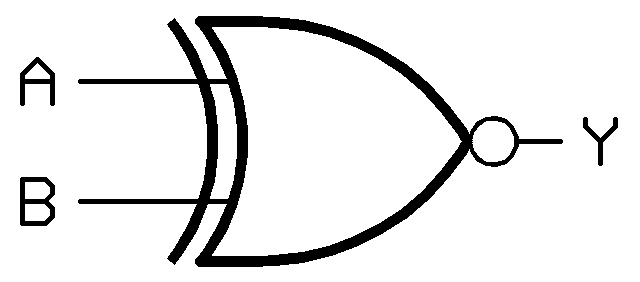
|  |  |  |  |
| --- | --- | --- | --- |
| **No of 1’s even/odd** | **A** | **B** | **OUT** |
| **Even** | **0** | **0** | **1** |
| **Odd** | **0** | **1** | **0** |
| **Odd** | **1** | **0** | **0** |
| **Even** | **1** | **1** | **0** |



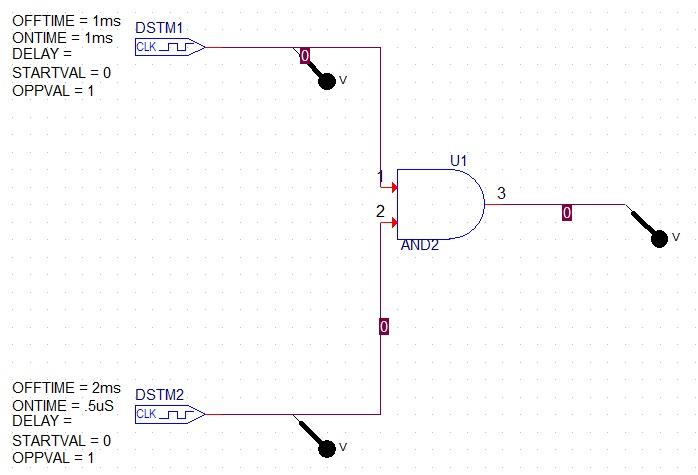
**XNOR gate (IC 7486+7404)**

The XNOR gate (sometimes ENOR, EXNOR or NXOR and pronounced as Exclusive NOR) is a digital logic gate whose function is the logical complement of the Exclusive O (XOR) gate.[1] It is equivalent to the logical connective from mathematical logic, also known as the material biconditional. The two-input version implements logical equality behaving according to the truth table to the right, and hence the gate is sometimes called an "equivalence gate". A high output (1) results if both of the inputs to the gate are the same. one but not both inputs are high (1), a low output (0) result.

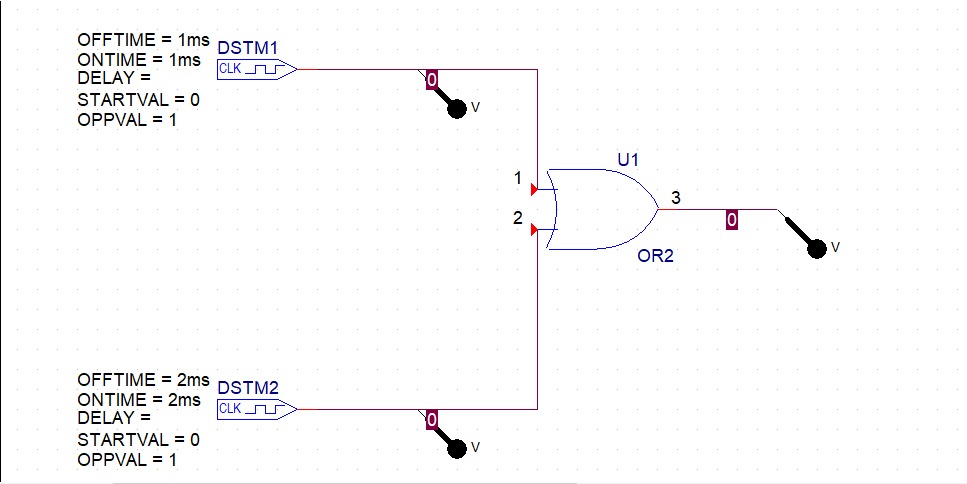
|  |  |  |  |
| --- | --- | --- | --- |
| **No of 1’s even/odd** | **A** | **B** | **OUT** |
| **Even** | **0** | **0** | **1** |
| **Odd** | **0** | **1** | **0** |
| **Odd** | **1** | **0** | **0** |
| **Even** | **1** | **1** | **1** |



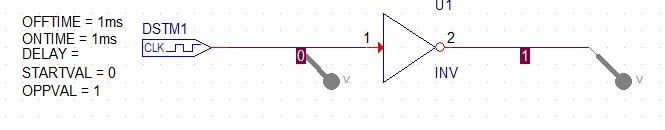
**OR-CAD Schematic Diagram**

**AND gate (IC7408)**

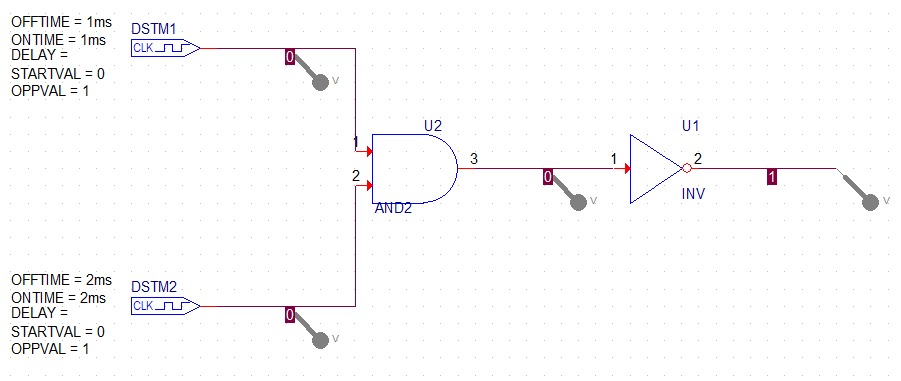
**OR gate (IC7432)**



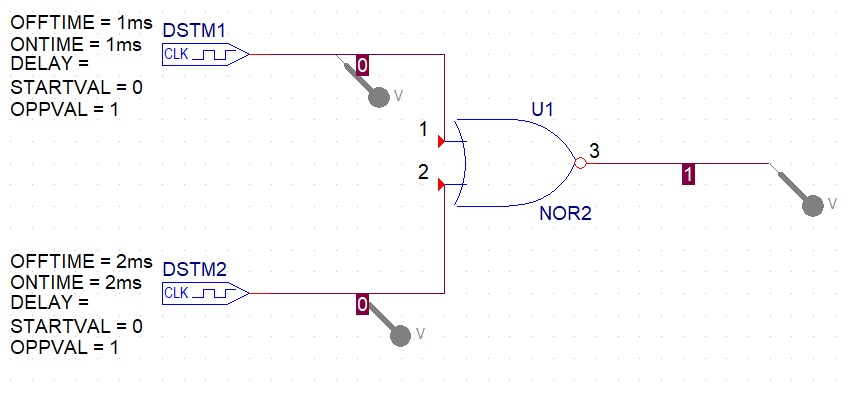
**NOT gate (IC7404)**



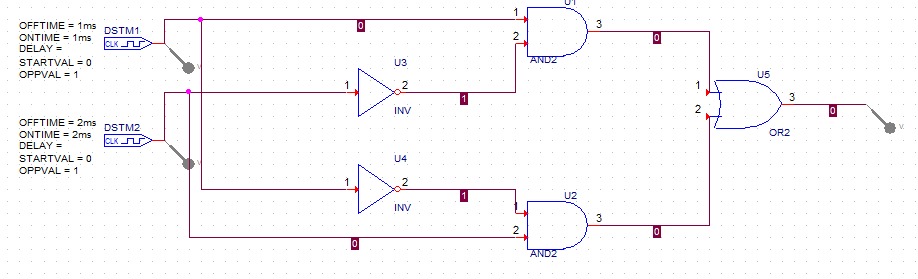
**NAND gate (IC7400)**



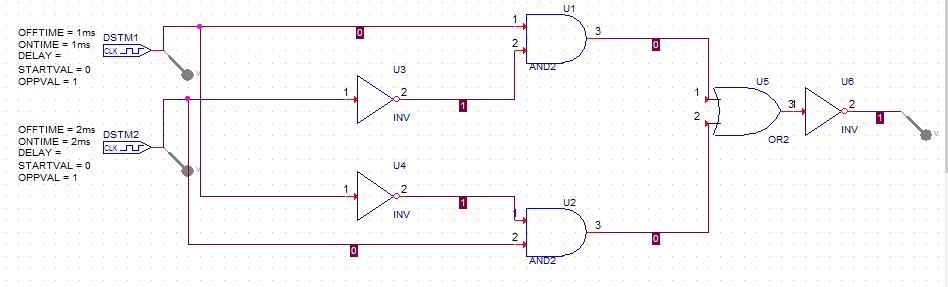
**NOR gate (IC7402)**



**XOR gate (IC7486)**

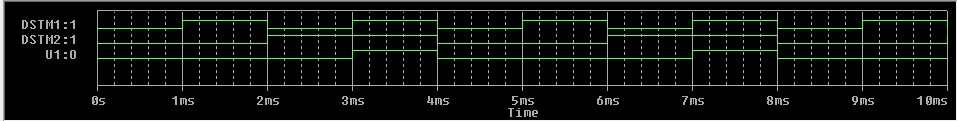


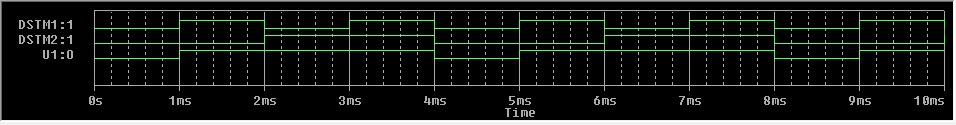
**XNOR gate (IC7486+7404)**



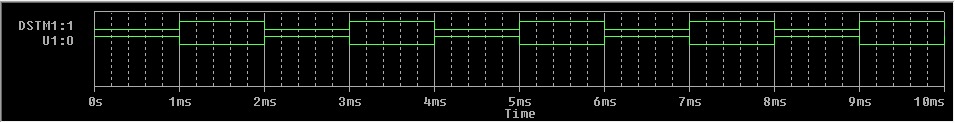
**Wave-forms**

**AND gate (IC7408)**

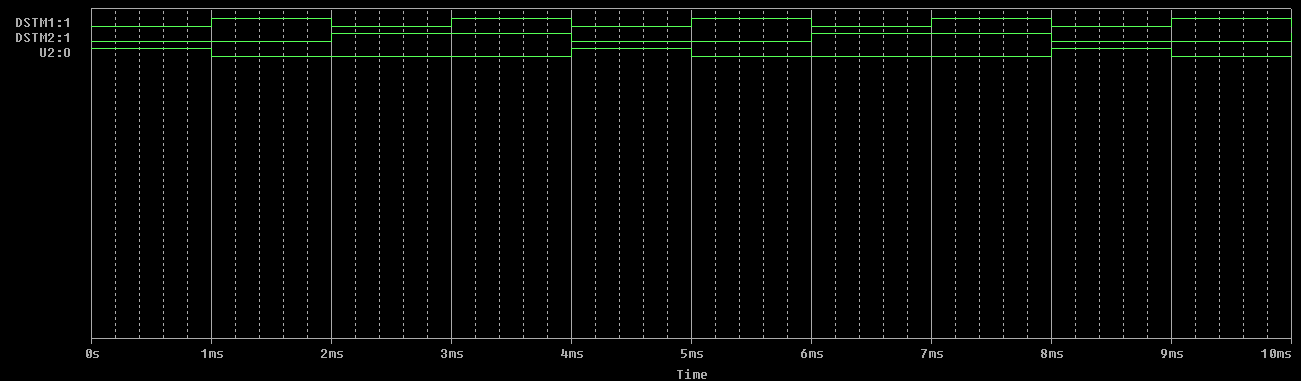
****

**OR gate (IC7432)**

**NOT gate (IC7404)**

****

**NAND gate (IC7400)**



**NOR gate (IC7402)**

A picture containing text, monitor, indoor, screen

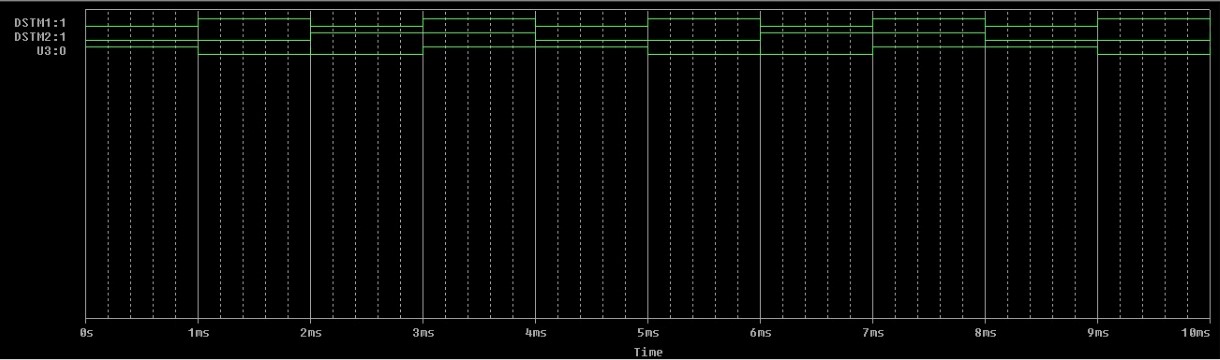
Description automatically generated

**XOR gate (IC7486)**

A picture containing text, electronics, computer

Description automatically generated

**XNOR gate (IC7486+7404)**



**Conclusion:** All logic gates are defined and verified using OR-CAD.

**Experiment 1**

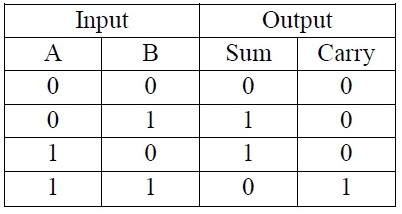
**Objective:** To simulate Half Adder circuit.

**Software Used:** ORCAD, Pspice.

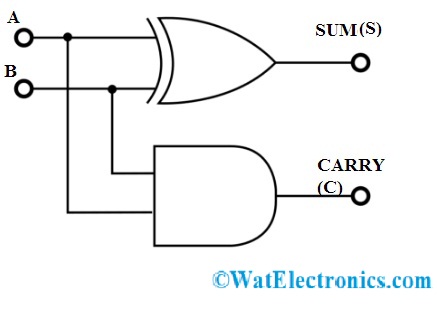
**Theory:** A half adder is used to add two single-digit binary numbers and results into a two-digit output. It is named as such because putting two half adders together with the use of an OR gate results in a full adder. In other words, it only does half the work of a full adder.

The adder works by combining the operations of basic logic gates, with the simplest form using only a XOR and an AND gate. This can also be converted into a circuit that only has AND, OR and NOT gates. This is especially useful since these three simpler logic gate ICs (integrated circuits) are more common and available than the XOR IC, though this might result in a bigger circuit since three different chips are used instead of just one.

**Truth Table**

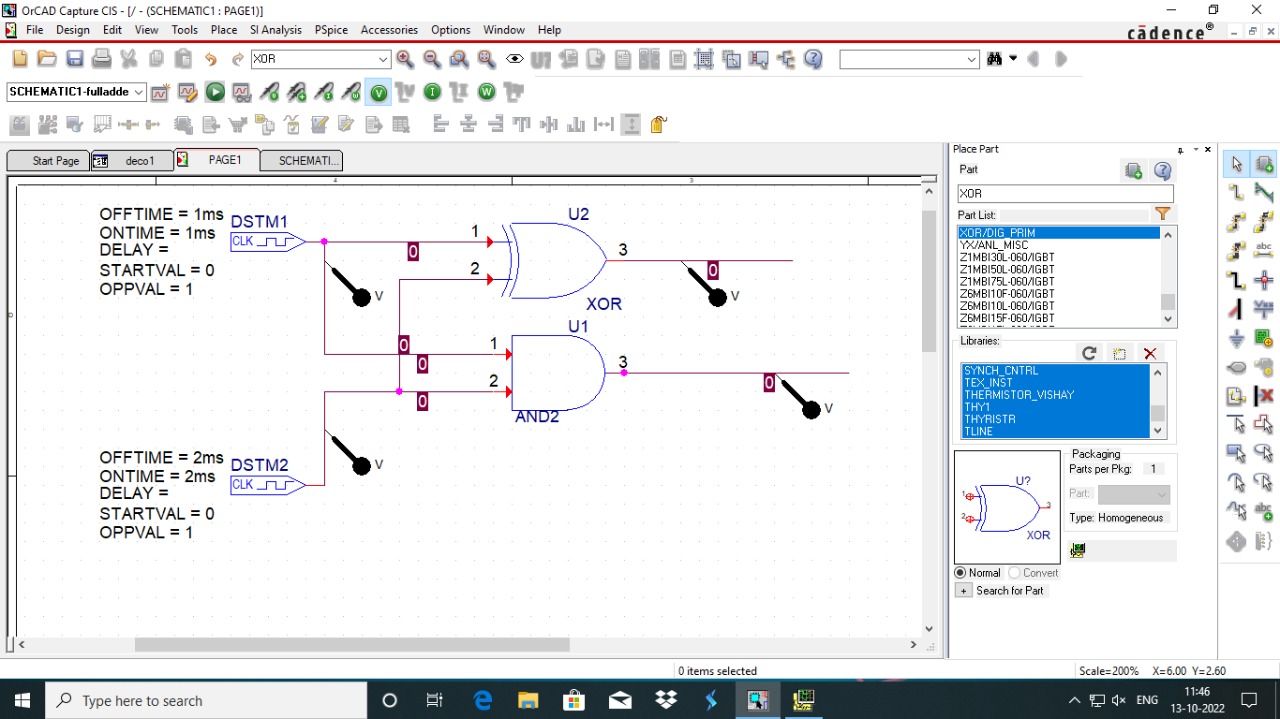


**Circuit Diagram**

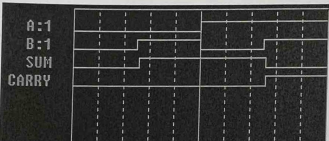
****

**Schematic Diagram**

**Half-Adder**



**Waveform**

****

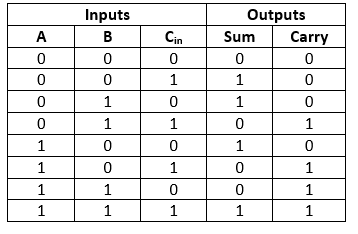
**Conclusion:** All circuits are defined and verified using OR-CAD.

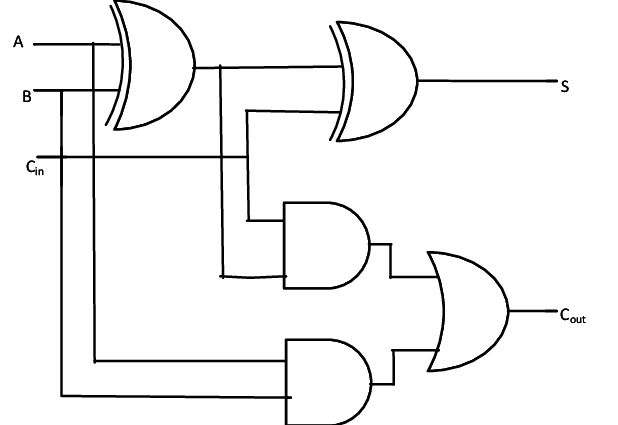
**Experiment 2**

**Objective:** Design to verify full adder circuit using OR-CAD and Pspice

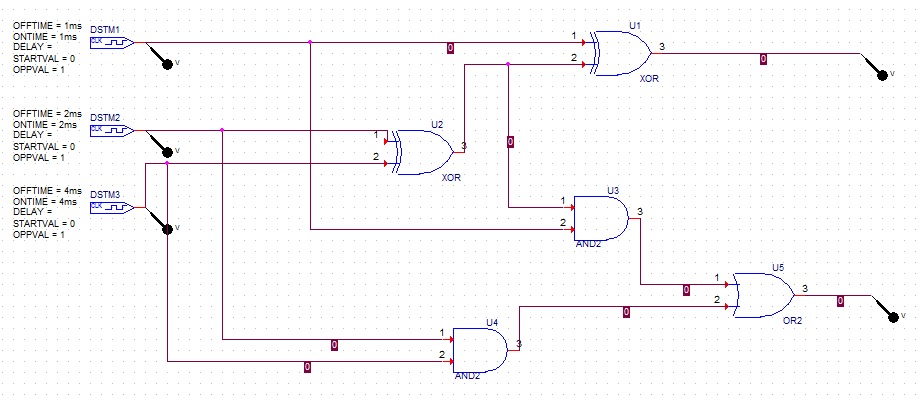
**Software Used:** ORCAD, Pspice.

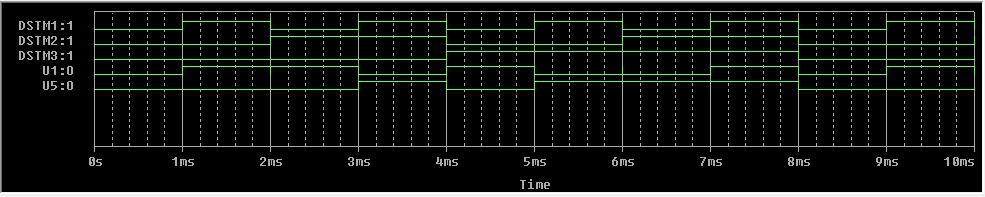
**Theory:** The full-adder has three inputs and two outputs. The first two inputs are A and B and the third input is an input carry. When a full adder logic is designed, we will be able to string eight of them together to create a byte-wide adder and cascade the carry bit from one adder to the next. A full adder circuit can be implemented with the help of two half adder circuits. The first will have the half adder to add A and B to produce a partial Sum. The second half adder logic can be used to add input carry to the Sum produced by the first half adder to get the final sum output. If any of the half adder logic produces a carry, there will be an output carry.





**Schematic Diagram**



**Waveform**

**Conclusion:** All circuits are defined and verified using OR-CAD.

**Experiment 3**

**Objective:** To Simulate and Verify the Half Subtractor and the Full Subtractor.

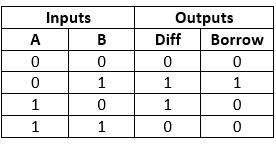
**Software Used:** ORCAD, Pspice.

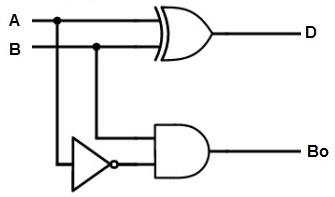
**Theory:** Subtractor circuits take two binary numbers as input and subtract one binary input from the other binary number input. Similar to adders, it gives out two outputs, difference and borrow (carry-in the case of Adder). There are two types of subtractors.

1. Half Subtractor
2. Full Subtractor

**Half Subtractor:**

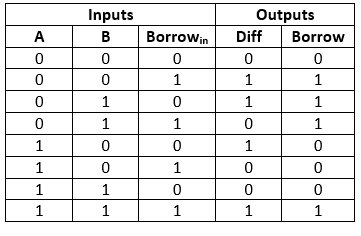
The half-subtractor is a combinational circuit which is used to perform subtraction of two bits. It has two inputs, A (minuend) and B (subtrahend) and two outputs Difference and Borrow. The logic symbol and truth table are shown below.

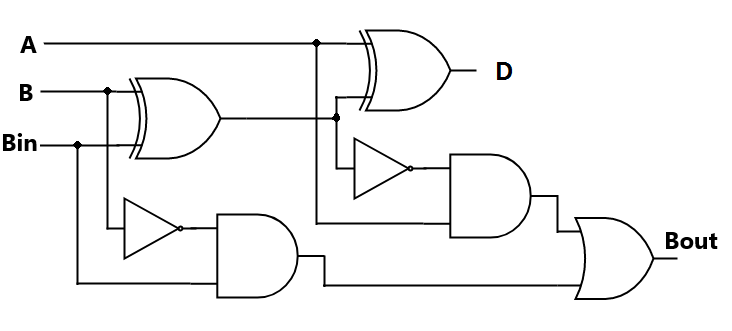
**Truth Table:**



**Full Subtractor:**

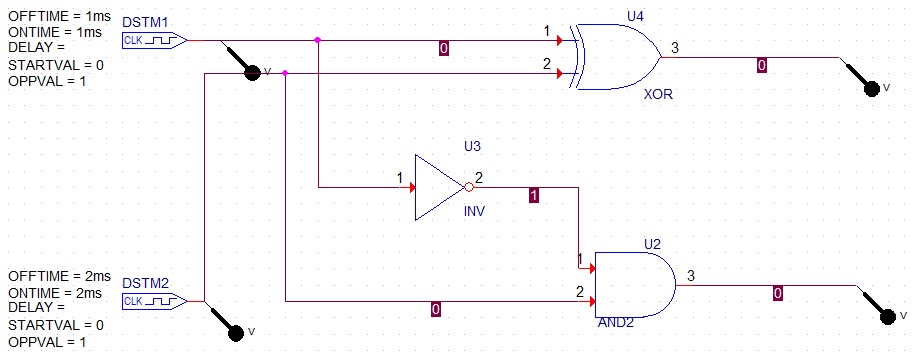
A full subtractor is a combinational circuit that performs subtraction involving three bits, namely A (minuend), B (subtrahend), and Bin (borrow-in). It accepts three inputs: A (minuend), B (subtrahend) and a Bin (borrow bit) and it produces two outputs: D (difference) and Bout (borrow out). The logic symbol and truth table are shown below.

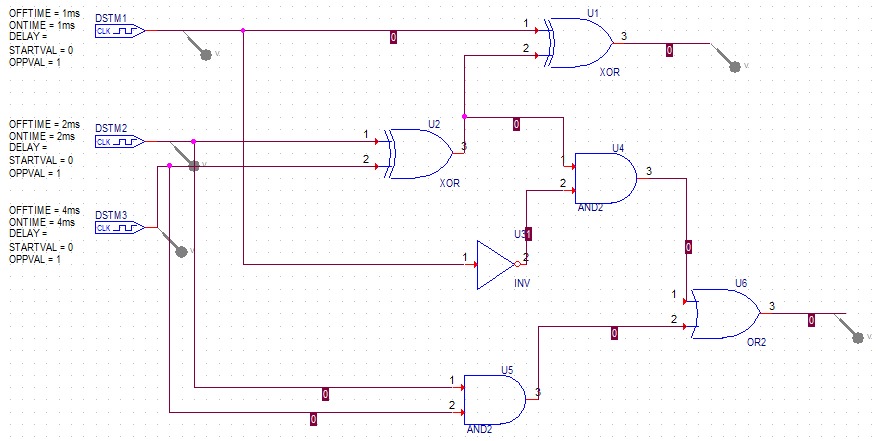
**Truth Table:**



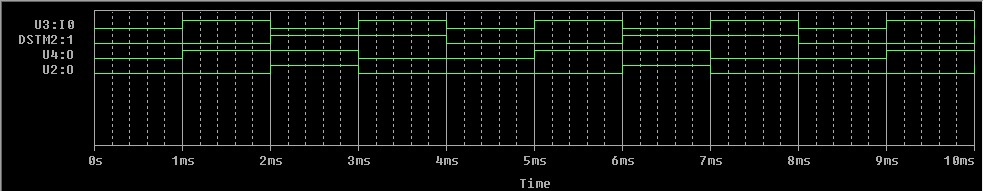
**Schematic Diagram**

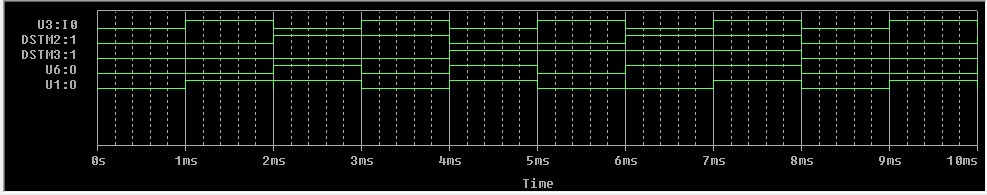
**Half Subtractor:**

****

**Full Subtractor:**

**Waveform**

**Half Subtractor:**

**Full Subtractor:**

**Conclusion:** All circuits are defined and verified using OR-CAD.

**Experiment 4**

**Objective:** Introduction to 8085 Microprocessor.

**Theory:**

8085 Microprocessor — Functional Units

8085 consists of the following functional units —

**Accumulator**

It Is an 8-bit register used to perform arithmetic, logical, I/O & LOAD/STORE operations. It is connected to internal data bus & ALU.

**Arithmetic and Logic Unit**

As the name suggests, it performs arithmetic and logical operations like Addition, Subtraction, AND, OR, etc. on 8-bit data.

**General Purpose Register**

There are 6 general purpose registers in 8085 processor, i.e., B, C, D, E, H & L. Each register can hold 8-bit data.

These registers can work in pair to hold 16-bit data and their pairing combination is like BC, D-E & H-L.

**Program Counter**

It is a 16-bit register used to store the memory address location of the next instruction to be executed. Microprocessor increments the program whenever an instruction is being executed, so that the program counter points to the memory address of the next instruction that is going to be executed.

**Stack Pointer**

It is also a 16-bit register works like stack, which is always incremented/decremented by 2 during push & pop operations.

**Temporary Register**

It is an 8-bit register, which holds the temporary data of arithmetic and logical operations.

**Flag Register**

It is an 8-bit register having five 1-bit flip-flops, which holds either 0 or 1 depending upon the result stored in the accumulator.

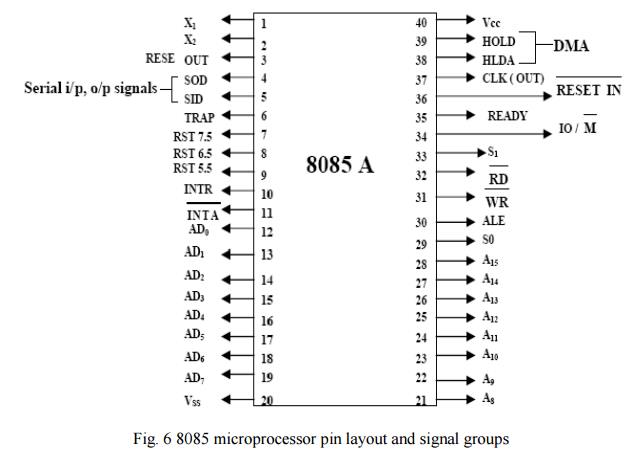
These are the set of 5 flip-flops —

* Sign (S)
* Zero (Z)
* Auxiliary Carry (AC)
* Parity (P)
* Carry

**Address bus and data bus**

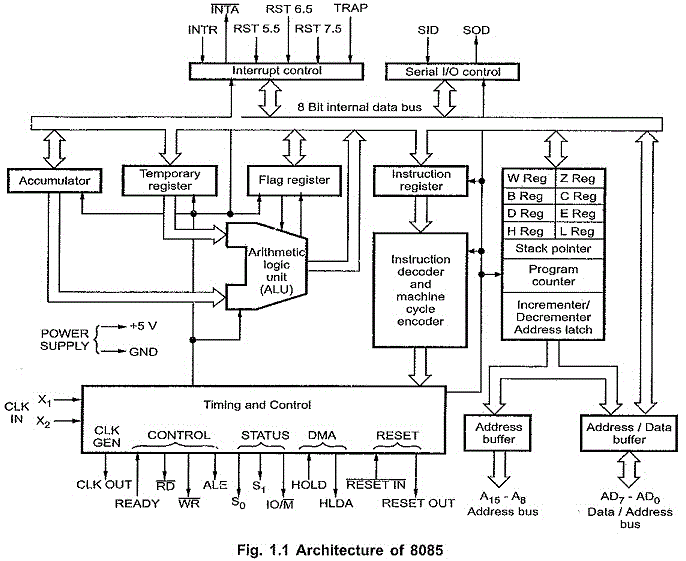
Data bus carries the data to be stored. It is bidirectional, whereas address bus carries the location to where it should be stored and it is unidirectional. It is used to transfer Address I/O devices.

**PIN DIAGRAM:**

****

**8085 Architecture:**

* It is a 40 pin LC. package fabricated on a single LSI chip.
* The Intel 8085 uses a single +5Vd.c. supply for its operation.
* Intel 8085’s clock speed is about 3 MHz; the clock cycle is of 320ns.
* 8-bit data bus. Address bus is of 16-bit, which can address up to 64KB
* 16-bit stack pointer
* 16-bit PC (Program Counter)
* Six 8-bit registers are arranged in pairs: BC, DE, HL



**Experiment 5**

**Objective:** Program to add and subtract using 8085 microprocessors.

**Theory:**

**Program to Add**

**Algorithm –**

* Load the first number from memory location 2050 to accumulator.
* Move the content of accumulator to register H.
* Load the second number from memory location 2051 to accumulator.
* Then add the content of register H and accumulator using “ADD” instruction and Storing result at 3050
* The carry generated is recovered using “ADC” command and is stored at memory location 3051

Input Data

Memory Address

|  |  |
| --- | --- |
| F9 | 3B |
| 2051 | 2050 |

Carry

Output Data

Memory Address

|  |  |
| --- | --- |
| 01 | 34 |
| 3051 | 3050 |

**Program –**

|  |  |  |  |
| --- | --- | --- | --- |
| **Memory** | **Address** | **Mnemonics** | **Comment** |
| 2000 | LDA | 2050 | A <- [2050] |
| 2003 | MOV | H, A | H <- A |
| 2004 | LDA | 2051 | A <- [2051] |
| 2007 | ADD | H | A <- A+H |
| 2008 | MOV | L, A | L <- A |
| 2009 | MVI | A 00 | A <- 00 |

**Explanation –**

* LDA 2050 moves the contents of 2050 memory location to the accumulator.
* MOV H, A copies contents of Accumulator to register H to A
* LDA 2051 moves the contents of 2051 memory location to the accumulator.
* ADD H adds contents of A (Accumulator) and H register (F9). The result is stored in itself. For all arithmetic instructions A is by default an operand and A stores the result as well.
* MOV L, A copies the contents of A (34) to L.
* MVI A 00 moves immediate data (i.e., 00) to A.
* ADC A adds contents of A (00), contents of register specified (i.e., A) and carry (1). As ADC is also an arithmetic operation, A is by default an operand and A stores the result as well.
* MOV H, A copies contents of A (01) to H.
* SHLD 3050 moves the contents of L register (34) in 3050 memory location and contents of H register (01) in 3051 memory location.
* HLT stops executing the program and halts any further execution.

**Program to Subtract**

Algorithm –

* Load 00 in a register C (for borrow)
* Load two 8-bit number from memory into registers
* Move one number to accumulator Subtract the second number with accumulator
* If borrow is not equal to 1, go to step 7
* Increment register for borrow by 1
* Store accumulator content in memory
* Move content of register into accumulator
* Store content of accumulator in other memory location
* Stop

H-L

Input Data

Memory Address

|  |  |
| --- | --- |
| 03 | 04 |
| 2501 | 2500 |

Output Data

Memory Address

Borrow

Result

|  |  |
| --- | --- |
| 01 | 01 |
| 2503 | 2502 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Memory** | **Mnemonics** | **Operands** | **Comment** |
| 2000 | MVI | C, 00 | [C] <- 00 |
| 2002 | LHLD | 2500 | [H-L] <- [2500] |
| 2005 | MOV | A, H | [A] <- [H] |
| 2006 | SUB | L | [A] <- [A] – [L] |
| 2007 | JNC | 200B | Jump if no borrow |
| 200A | INR | C | [C] <- [C] + 1 |
| 200B | STA | 2502 | [A] -> [2502], Result |
| 200E  MOV | A, C | [A] <- [C] |  |
| 2010 | STA | 2503 | [A] -> [2503],  Borrow |
| 2013 | HLT |  | Stop |

**Explanation** — Registers A, H, L, C are used for general purpose:

* MOV is used to transfer the data from memory to accumulator (1 Byte)
* LHLD is used to load register pair directly using 16-bit address (3 Byte instruction)
* MVI is used to move data immediately into any of registers (2 Byte)
* STA is used to store the content of accumulator into memory (3 Byte instruction)
* INR is used to increase register by 1 (1 Byte instruction)
* JNC is used to jump if no borrow (3 Byte instruction)
* SUB is used to subtract two numbers where one number is in accumulator (1 Byte)
* HLT is used to halt the program

**Result**: Verified Using the program.

**Experiment 6**

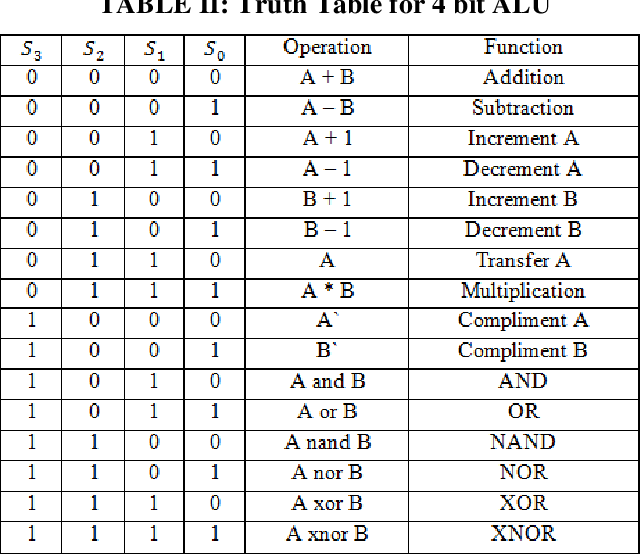
**Objective:** To simulate the logical part of the arithmetic logic unit

**Software Used:** ORCAD, Pspice.

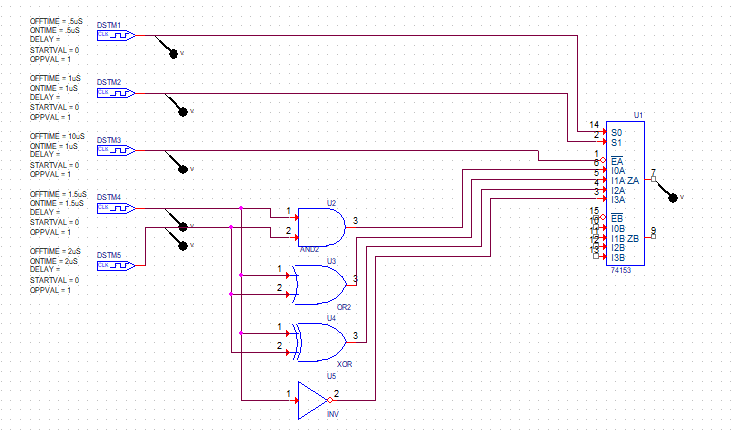
**Theory:**

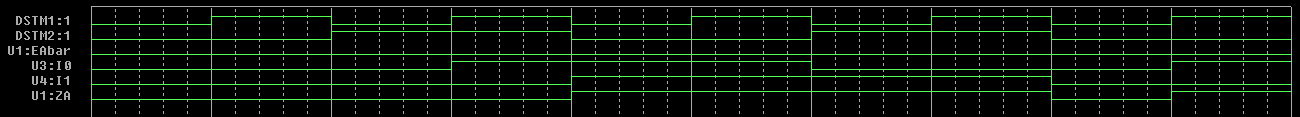
In computing, an arithmetic logic unit (ALU) is a combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers. This is in contrast to a floating-point unit (FPU), which operates on floating point numbers. It is a fundamental building block of many types of computing circuits, including the central processing unit (CPU) of computers, FPUs, and graphics processing units (GPUs).

The inputs to an ALU are the data to be operated on, called operands, and a code indicating the operation to be performed; the ALU's output is the result of the performed operation. In many designs, the ALU also has status inputs or outputs, or both, which convey information about a previous operation or the current operation, respectively, between the ALU and external status registers.



**Schematic Diagram:**

****

**Waveform:**

**Result:** All circuits are defined and verified using the ORCAD

**Experiment 7**

**Objective:** To simulate a 4-bit adder/subtractor circuit using ORCAD

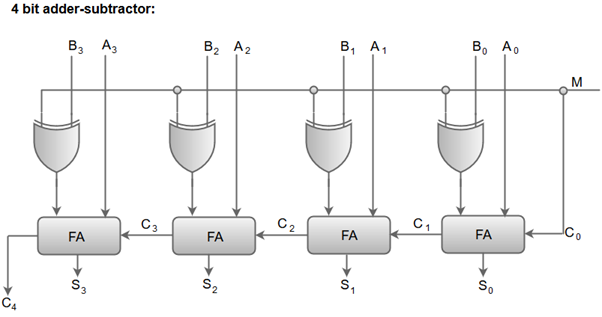
**Software Used:** ORCAD and Pspice.

**Theory:**

IC 7483 performs the addition of two 4-bit binary numbers.

A3 A2 A1 A0 and B3 B2 B1 B0 and carry input to give the output S3 S2 S1 S0 and carry out. So, for adding the two numbers A3 A2 A1 A0 and B3 B2 B1 B0, the two numbers are given to the input terminals 1, 3, 8, 10 and 16, 4, 7, 11 of the IC 7483 and carry in the terminal 13 is set to zero. To subtract two numbers by two’s compliment method, we are adding the two’s compliment of the second number to each of the four bits of the first numbers. The final carry is neglected and the difference is taken from S3 S2 S1 S0.

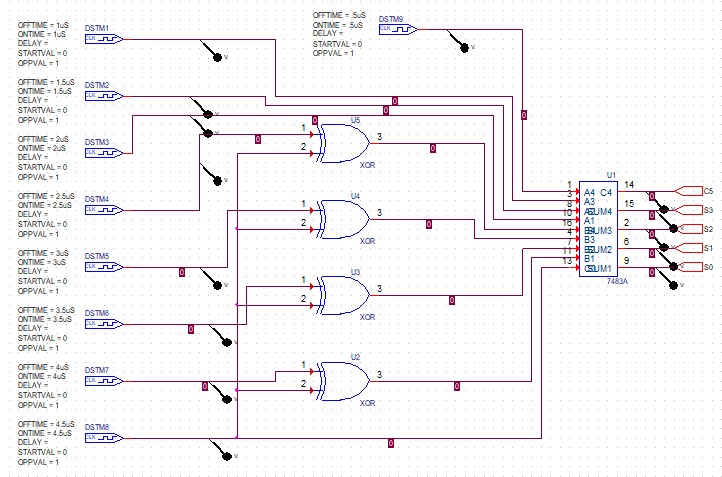
In the circuit, we set mode control such that when the mode control is zero, addition is performed and subtraction is performed when the mode control is one. We use XOR gate to feed the input so that when mode control is one, the complement of each of the four biases are fed and when mode control is zero, the input as such is fed.



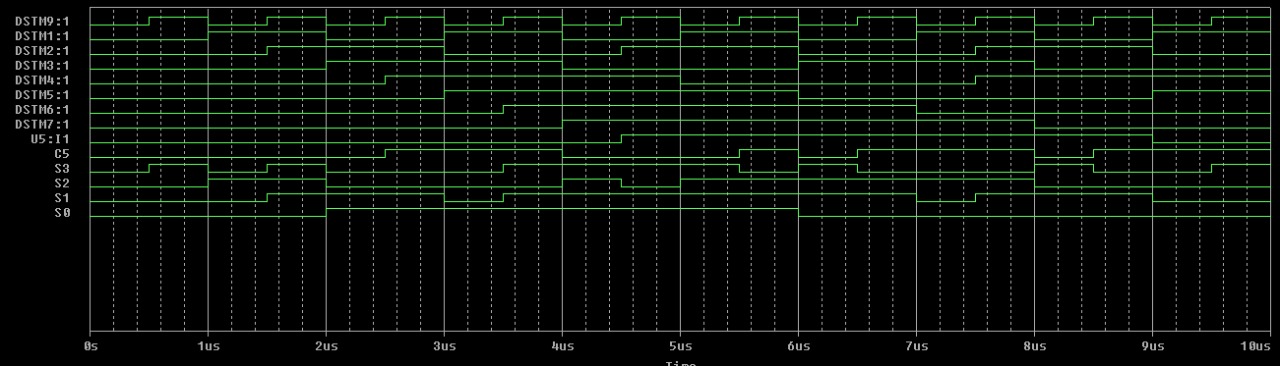
In 1’s complement subtraction, the complement of the subtraction is taken and added with the other number. The final carry is then added to the LSB of the result. In case there no carry, the complement of the result is taken and this will be a negative number. This indicates that, subtraction is performed from a smaller number.

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **M** | **S** |
| 0000 | 0000 | 0 | 00000 |
| 0001 | 0000 | 1 | 00001 |
| 0011 | 0001 | 0 | 00100 |
| 0011 | 0001 | 1 | 00010 |
| 0100 | 0001 | 0 | 00101 |
| 0100 | 0001 | 1 | 00011 |
| 1000 | 0101 | 0 | 01101 |
| 1000 | 0101 | 1 | 00011 |

**Schematic Diagram:**

****

**Waveform:**

****

**Result:** All circuits are defined and verified using the OR-CAD.

**Experiment 8**

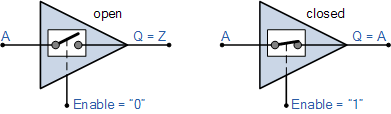
**Objective:** To simulate and study tri-state buffer.

**Software Used:** ORCAD and Pspice.

**Theory:**

A Tri-state Buffer can be thought of as an input-controlled switch with an output that can be electronically turned “ON” or “OFF” by means of an external “Control” or “Enable” (EN) signal input. This control signal can be either a logic “0” or a logic “1” type signal resulting in the Tri-state Buffer being in one State allowing its Output to operate normally producing the required output or in another state where its output is blocked or disconnected.

**Tri-state Buffer Switch Equivalent**

****

When activated into its third state it disables or turns “OFF” its output producing an open circuit condition that is neither at a logic “HIGH” or “LOW”, but instead gives an output state of very high impedance, High-Z, or more commonly Hi-Z. Then this type of device has two logic state inputs, “0” or a “1” but can produce three different output states, “0”, “1” or” Hi-Z” which is why it is called a “Tri” or “3-state” device.

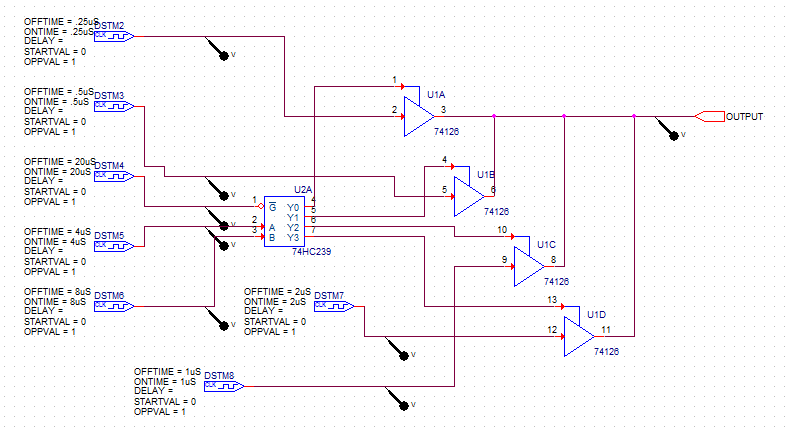
|  |  |  |
| --- | --- | --- |
| **C** | **A** | **F** |
| 0 | 0 | Z |
| 0 | 1 | Z |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Note that this third state is NOT equal to a logic level “0” or “1”, but is a highly impedance state in which the buffers output is electrically disconnected from the rest of the circuit. As a result, no current is drawn from the supply.

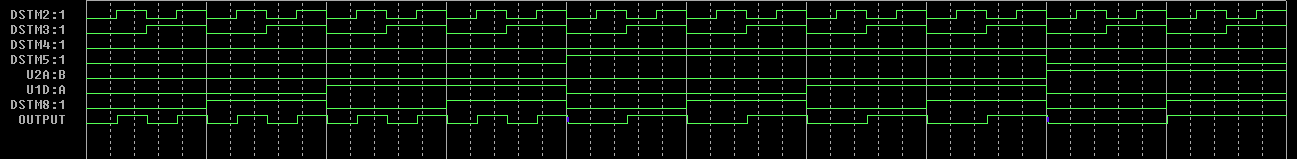
|  |  |  |
| --- | --- | --- |
| **S1** | **S0** | **Register Selected** |
| 0 | 0 | A |
| 0 | 1 | B |
| 1 | 0 | C |
| 1 | 1 | D |

The connected buffers must be controlled so that only one three-state buffer has access to the bus line while others are in high impedance. Decoder can be used to control the active input at any given time. When enable input of decoder is 0 all the outputs are zero and the lines are in high impedance. When enable is 1 then one of the four outputs of the decoder is active depending upon binary values of the selection lines.

**Schematic Diagram:**

****

**Waveform:**

****

**Result:** All circuits are defined and verified using the ORCAD.

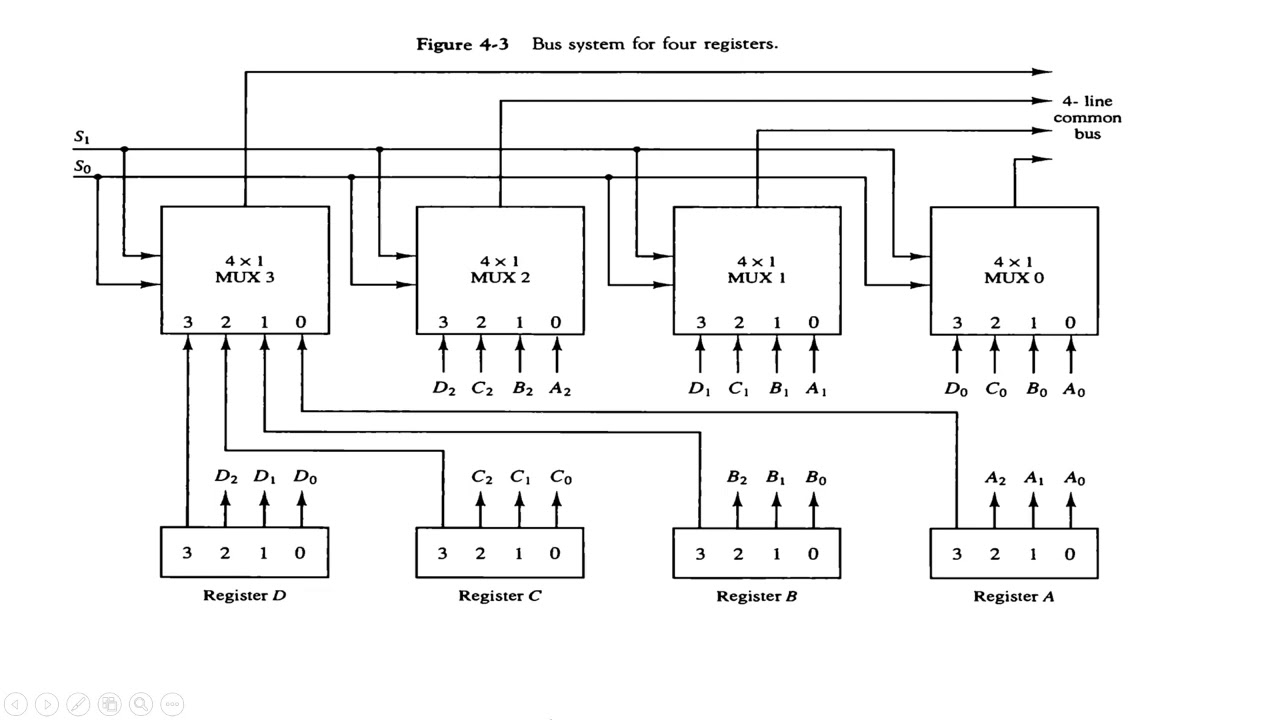
**Experiment 9**

**Objective:** To simulate common bus with multiplexers.

**Software Used:** ORCAD and Pspice.

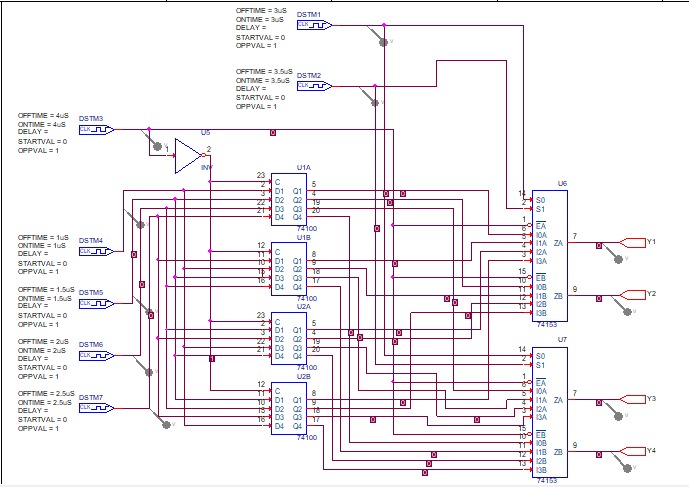
**Theory:**

A typical computer has many registers and we need to transfer the information between these registers. A way to transfer the information is using the common bus system. In this article we shall discuss the common bus system using multiplexers.

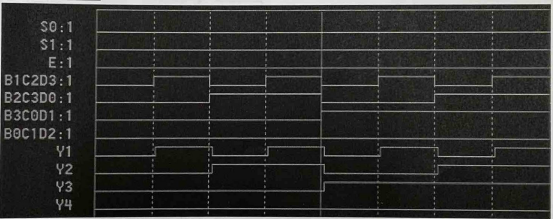
****

The construction of this bus system for 4 registers is shown above. The bus consists of 4x1 multiplexers with 4 inputs and 1 output and 4 registers with bits numbered 0 to 3. There are 2 select inputs SO and S1 which are connected to the select inputs of the multiplexers.

The output 1 of register A is connected to input 0 of MUX 1 and similarly other connections are made as shown in the diagram. The data transferred to the bus depends upon the select lines. A table for the various combinations of select lines is shown below.

**Schematic Diagram:**

**Waveform:**

****

**Result:** All circuits are defined and verified using the ORCAD.

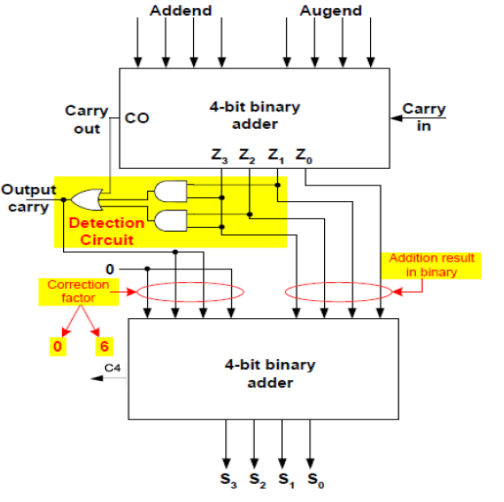
**Experiment 10**

**Objective:** To design BCD adder/subtractor and to be familiar with its functionality.

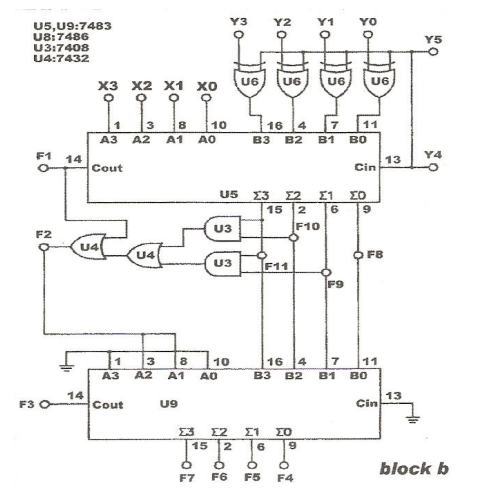
**Software Used:** ORCAD and Pspice.

**Theory:**

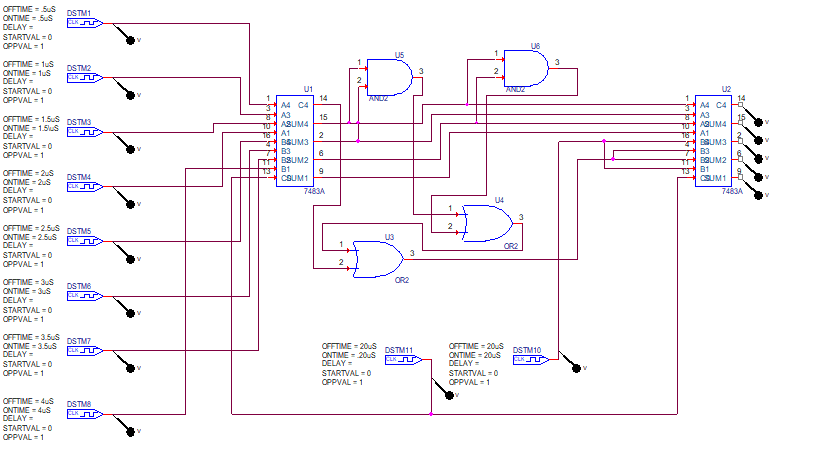
Binary adders can be converted into BCD adders. Since BCD has 4 bits with the largest number being 9; and the largest 4-bit binary number is equivalent to 15, there is a difference of 6 between the binary and the BCD adder. Under the following conditions 6 (0110) must be added when binary adders are used to add BCD codes:

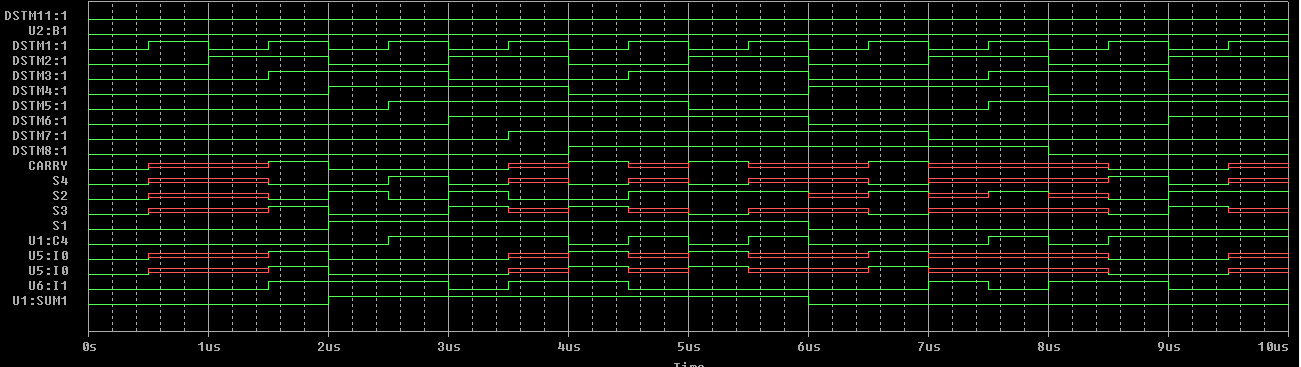
1. When there is any carry.
2. When the sum is larger than 9

BCD adder/subtractor circuit is below where the subtraction process is performed throughout adding the 2’s complement of the number to be subtracted. The circuit can work as adder when the input Y5 (the same as Y4) equals zero and as a subtractor when the input Y5 equals one.



**Schematic Diagram:**



**Waveform:**

**Result:** All the circuit is checked and verified using ORCAD and Pspice.

**Open Ended Program 1**

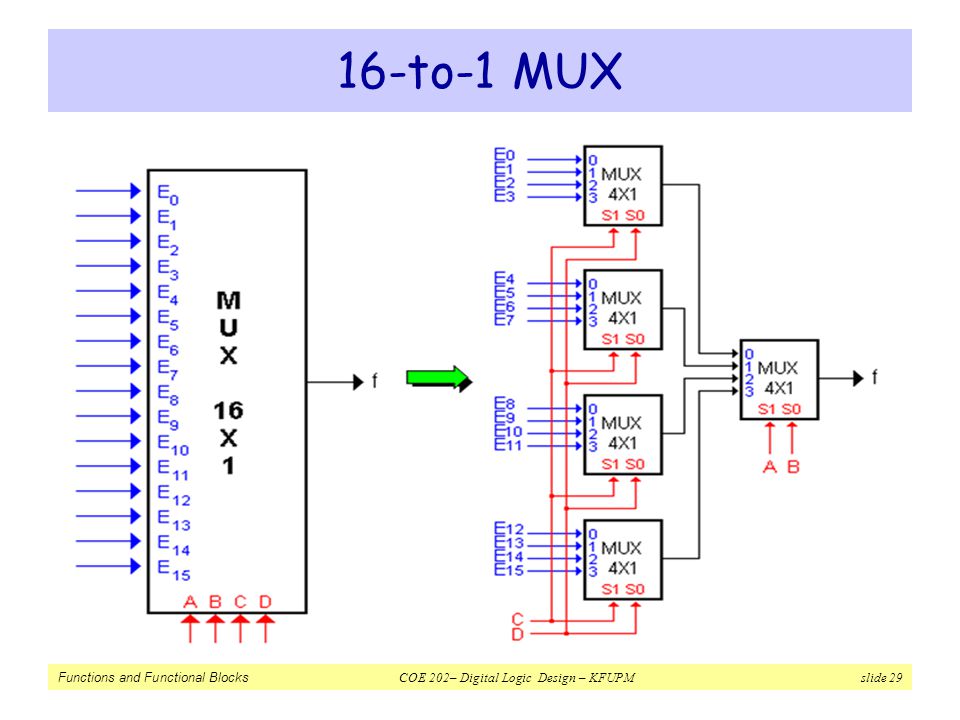
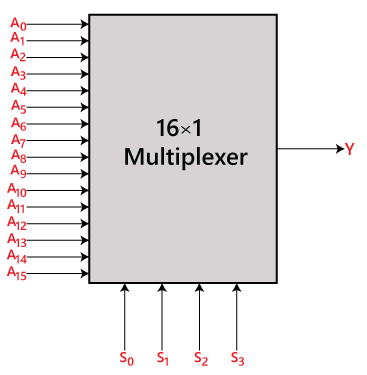
**Aim:** To make 16:1 MUX using 4:1 MUX

**Software Used:** ORCAD and Pspice.

**Theory:**

**Multiplexer –**

It is a combinational circuit which have many data inputs and single output depending on control or select inputs. For N input lines, log n (base2) selection lines, or we can say that for 2n input lines, n selection lines are required. Multiplexers are also known as “Data n selector, parallel to serial convertor, many to one circuit, universal logic circuit”. Multiplexers are mainly used to increase amount of the data that can be sent over the network within certain amount of time and bandwidth.



(a)

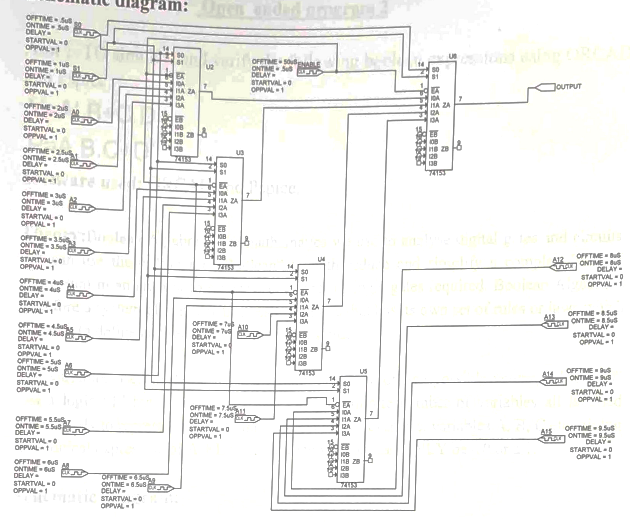
(b)

**Fig**: 16:1 MUX (a) Black-Box representation. (b) Architecture using 4:1 MUX

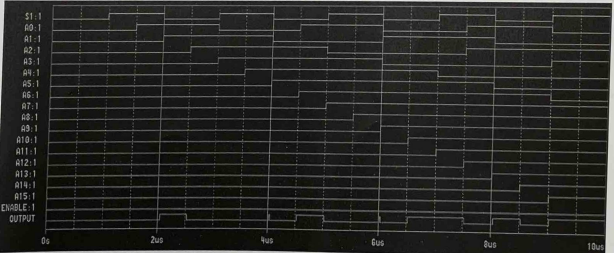
**Truth Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **EN** | **S3** | **S2** | **S1** | **S0** | **Y** |
| 1 | X | X | X | X | Z |
| 0 | 0 | 0 | 0 | 0 | IN0 |
| 0 | 0 | 0 | 0 | 1 | IN1 |
| 0 | 0 | 0 | 1 | 0 | IN2 |
| 0 | 0 | 0 | 1 | 1 | IN3 |
| 0 | 0 | 1 | 0 | 0 | IN4 |
| 0 | 0 | 1 | 0 | 1 | IN5 |
| 0 | 0 | 1 | 1 | 0 | IN6 |
| 0 | 0 | 1 | 1 | 1 | IN7 |
| 0 | 1 | 0 | 0 | 0 | IN8 |
| 0 | 1 | 0 | 0 | 1 | IN9 |
| 0 | 1 | 0 | 1 | 0 | IN10 |
| 0 | 1 | 0 | 1 | 1 | IN11 |
| 0 | 1 | 1 | 0 | 0 | IN12 |
| 0 | 1 | 1 | 0 | 1 | IN13 |
| 0 | 1 | 1 | 1 | 0 | IN14 |
| 0 | 1 | 1 | 1 | 1 | IN15 |

**Schematic Diagram:**



**Waveform:**



**Result**: All the circuits are checked and verified using ORCAD and Pspice.

**Open Ended Program 2**

**Aim:** To stimulate and verify the following Boolean expression using ORCAD.

**Y = A’.B + C.D**

**P = A.B.C + D’**

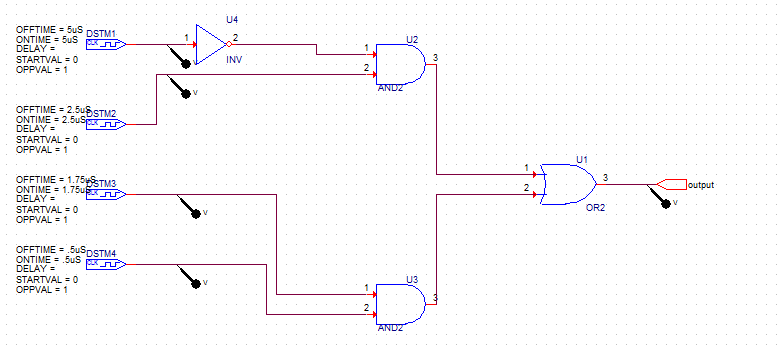
**Software Used:** ORCAD and Pspice.

**Theory:**

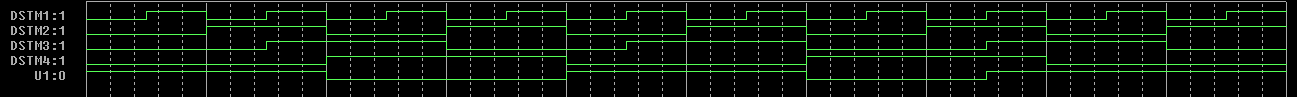
Boolean Algebra is the mathematics we use to analyze digital gates and circuits. We can use these “Laws of Boolean” to both reduce and simplify a complex Boolean expression in an attempt to reduce the number of logic gates required. Boolean Algebra is therefore a system of mathematics based on logic that has its own set of rules or laws which used to define and reduce Boolean expressions.

The variables used in Boolean Algebra only have one of two possible values, a logic “0” and a logic “1” but an expression can have an infinite number of variables all labelled individually to represent inputs to the expression, for example, variables A, B, C etc., giving us a logical expression of A + B =C, but each variable can ONLY be 0 and 1.

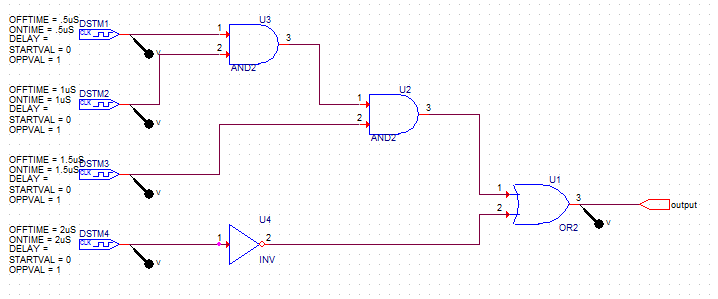
**Schematic Diagram:**



**Waveform:**

****

**Schematic Diagram:**



**Waveform:**

