**PROJECT DOCUMENT**



**CSE-206L Electronic Circuits Lab**

**Group members:**

**NAME REG.NO**

**Wajid Ullah 19PWCSE1759**

**Muhammad Ali 19PWCSE1801**

Class Section**: A**

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Submitted to:

**Engr. Abdullah Hamid**

Data:(15,08,2021)

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar

**Digital adder: -**

An adder is a [digital circuit](https://en.wikipedia.org/wiki/Digital_circuit) that performs [addition](https://en.wikipedia.org/wiki/Addition) of numbers. In many [computers](https://en.wikipedia.org/wiki/Computer) and other kinds of [processors](https://en.wikipedia.org/wiki/Microprocessor) adders are used in the [arithmetic logic units](https://en.wikipedia.org/wiki/Arithmetic_logic_unit) or ALU. They are also used in other parts of the processor, where they are used to calculate [addresses](https://en.wikipedia.org/wiki/Address_(computing)), table indices, [increment and decrement operators](https://en.wikipedia.org/wiki/Increment_and_decrement_operators) and similar operations.

**Half adder Work: -**

Our project based on half adder, so the half adder adds two single binary digits A and B. It has two outputs, sum (S) and carry (C). The carry signal represents an [overflow](https://en.wikipedia.org/wiki/Integer_overflow) into the next digit of a multi-digit addition. The value of the sum is 2C + S. The simplest half-adder design, pictured on the right, incorporates an [XOR gate](https://en.wikipedia.org/wiki/XOR_gate) for S and an [AND gate](https://en.wikipedia.org/wiki/AND_gate) for C. The Boolean logic for the sum (in this case S) will be A′B + AB′ whereas for the carry (C) will be AB. With the addition of an [OR gate](https://en.wikipedia.org/wiki/OR_gate) to combine their carry outputs, two half adders can be combined to make a full adder.[[1]](https://en.wikipedia.org/wiki/Adder_(electronics)#cite_note-Lancaster_2004-1) The half adder adds two input bits and generates a carry and sum, which are the two outputs of a half adder. The input [variables](https://en.wikipedia.org/wiki/Variable_(computer_science)) of a half adder are called the augend and addend bits. The output variables are the sum and carry.

**Full adder: -**

A full adder is a digital circuit that performs addition. Full adders are implemented with logic gates in hardware. A full adder adds three one-bit binary numbers, two operands and a carry bit. The adder outputs two numbers, a sum and a carry bit. The term is contrasted with a half adder, which adds two binary digits.

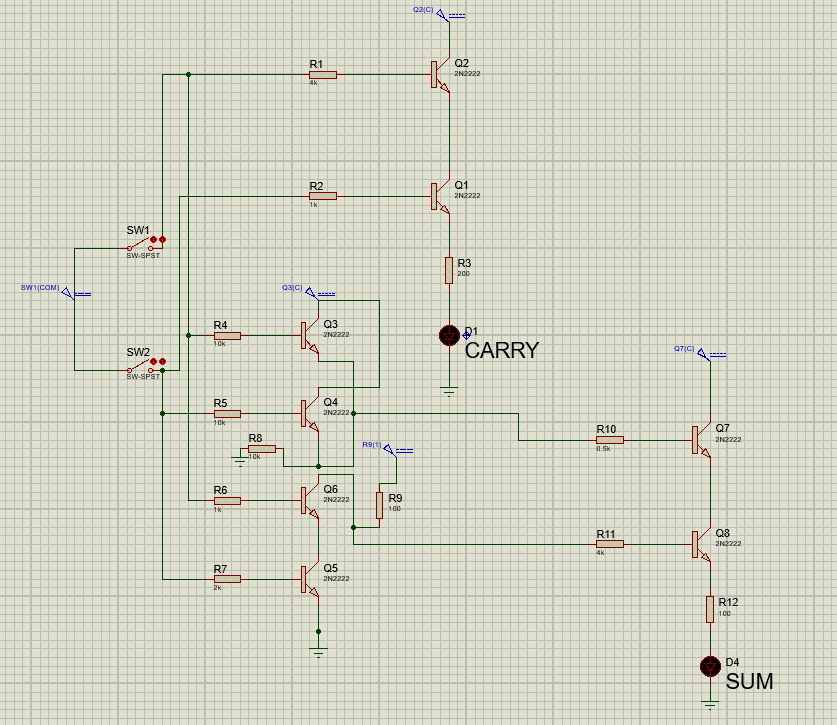
**Application: -**

Digital adders are mostly used in computer's ALU (Arithmetic logic unit) to compute addition. Digital calculators use adders for athematic addition. Micro controllers use adders in arithmetic additions,PC (program counter) and timers etc.

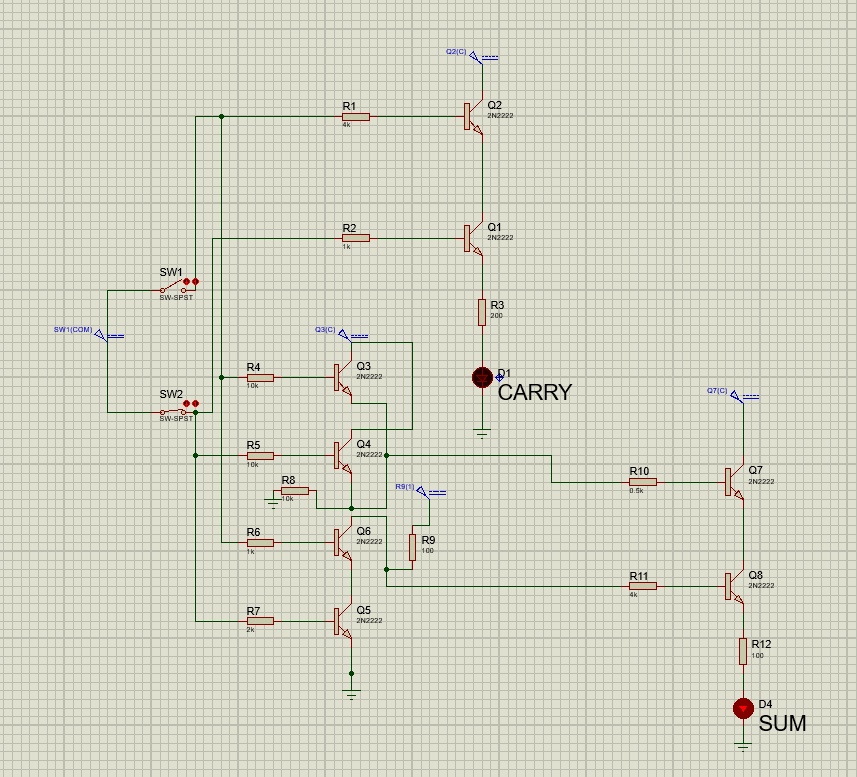
**Project constructions: -**

1. First we draw internal structure of half adder which consist of one AND and one XOR gate.
2. AND is for CARRY output and XOR for SUM output.
3. We simplified XOR gate to one OR, NAND gates and then its AND again so that we can easily construct each gate with NPN BJP transistor.
4. We made each and every gate with transistors in proteus as they are structure internally.
5. We give DC input to both AND and XOR gate, and variate input through two switches (open=0, close=1) of each gate.
6. And lastly examine the outputs by LEDS views (ON=1, OFF=0).

**Proteus schematic (A=0, B=0): -**

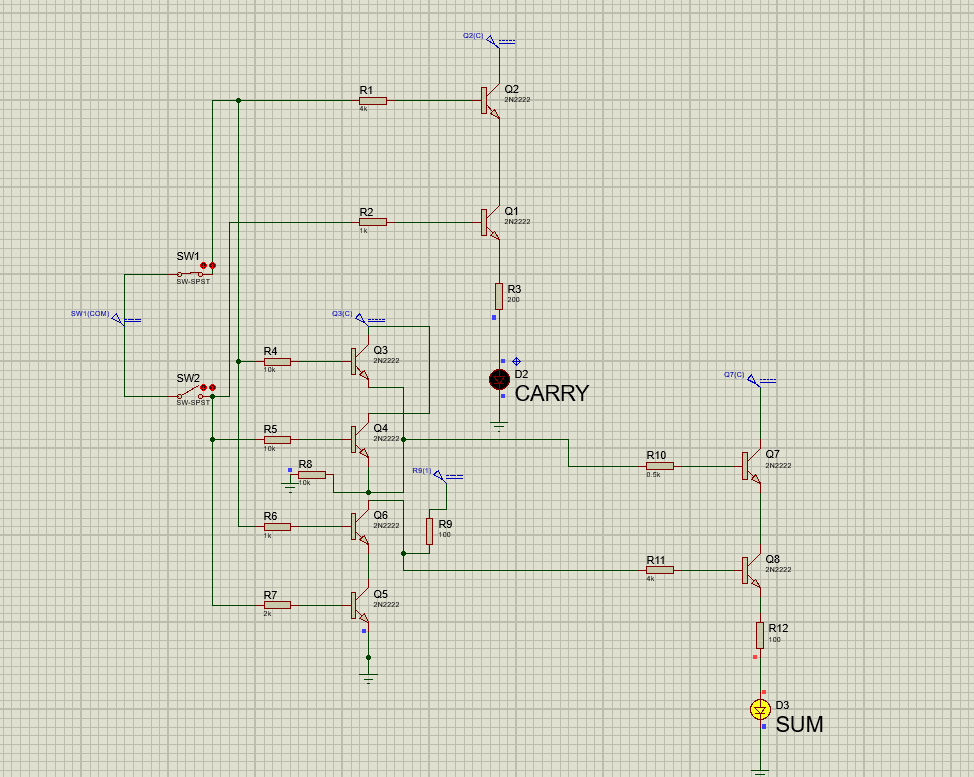
****

**Proteus schematic (A=0, B=1): -**

****

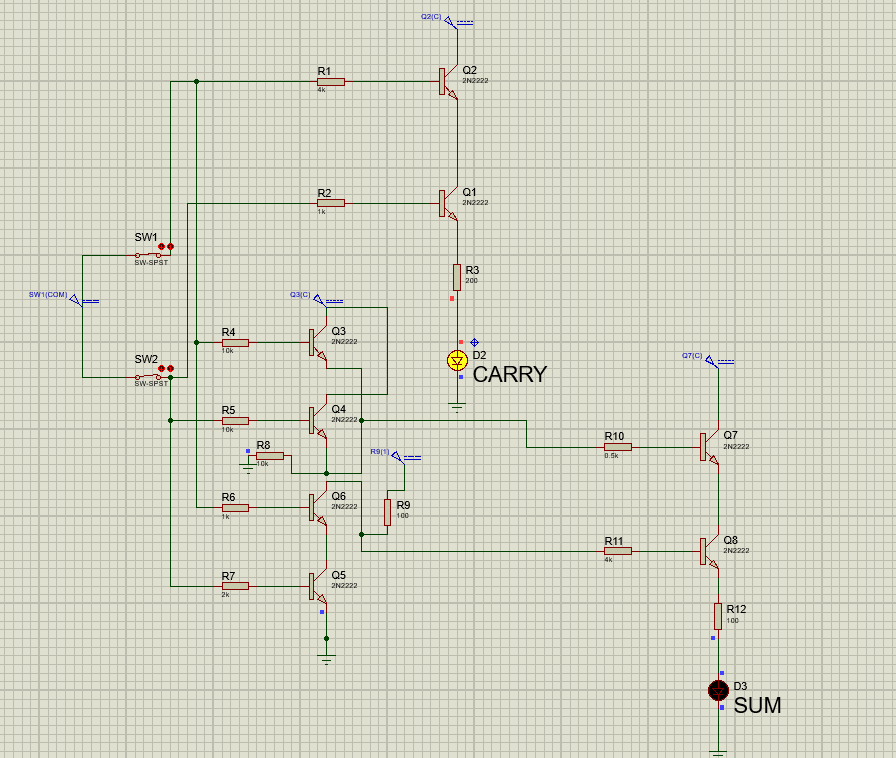
**Note:** Red carry is off and Red sum is on.

**Proteus schematic (A=1, B=0): -**

****

**Note:** yellow carry is off and yellow sum is on.

**Proteus schematic (A=1, B=1): -**

****

**Note:** yellow carry is on and yellow sum is off.

**Truth table of Half adder: -**

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **CARRY** | **SUM** |
| 0 (OPEN SWICTH) | 0 (OPEN SWITH) | 0 (LED OFF) | 0 (LED OFF) |
| 0 (OPEN SWICTH) | 1 (CLOSED SWITH) | 0 (LED OFF) | 1 (LED ON) |
| 1 (CLOSED SWICTH) | 0 (OPEN SWITH) | 0 (LED OFF) | 1 (LED ON) |
| 1 (CLOSED SWITH) | 1 (CLOSED SWITH) | 1 (LED ON) | 0 (LED OFF) |