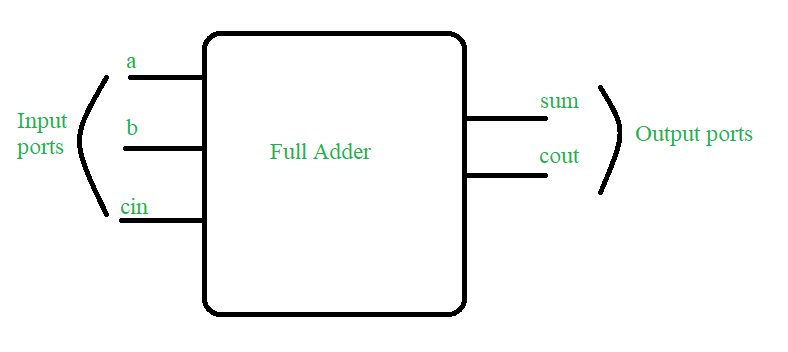
**Assignment-2**

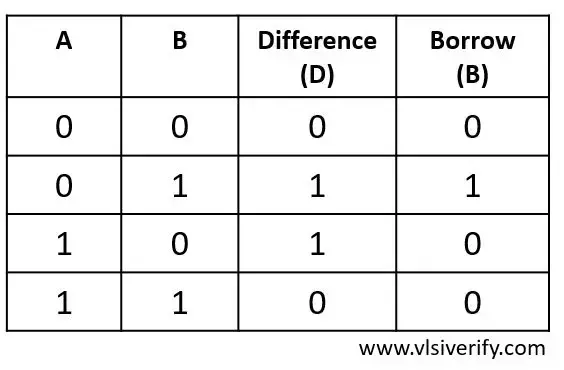
Code the following on verilog and test them using the testbench file. You have to submit both the module (**m**) file and the testbench (**tb**) file. Follow up:

1. Save each file as **RollNo\_A2\_Q{x}\_y** where (1<=x<=4) and y = {m, tb}
2. **Attach all the files in a drive link and submit it along with view access in** [**https://forms.gle/7uScS7X4ZkeZiDJD6**](https://forms.gle/7uScS7X4ZkeZiDJD6)
3. Implement a code for a digital circuit that adds three **binary** numbers. Given two inputs **A** and **B** and another Carry-In input, **Cin** from a previous stage. The outputs should be Sum, **S** and Carry-Out, **Cout**.

(Hint : Use Two stage half adders to achieve this!)



1. Implement a **behavioral** level code for the following truth table with input A, B and outputs Difference, D and Borrow, B:



1. A **ternary operator** follows the structure:

condition ? value\_if\_true : value\_if\_false;

Ex : Value = 0 ? chill : guy;

Since the condition is **false (0)**, the operator chooses the **false branch**, which is guy. So, the value of the expression is **guy**.

Write a Verilog code for the following problem statement:

Design a **signal routing mechanism** that controls the flow of a **100-bit** data stream (data\_in) to one of two outputs, based on a control signal. The system should ensure that the data is routed to only one of the outputs at a time, based on the value of the control signal. The mechanism works as follows:

* When the control signal **control = 0**, the signal (data\_in) is routed to Output 1 (out1).
* When the control signal **control = 1**, the signal (data\_in) is routed to Output 2 (out2).

1. You are a treasure hunter navigating a vault filled with **64 secret compartments**. Each compartment contains an **8-bit treasure** (a secret binary code), and these treasures are packed into a **single 512-bit vault** (input vector).

Your map gives you a **x-bit selector** (sel), which points to the specific treasure chest you need to unlock.

* If sel=0, you open the first chest and claim in [7:0] (bits 7 through 0).
* If sel=1, you unlock the second chest, containing in [15:8].
* If sel=2, you move to the third chest, revealing in [23:16].
* ... and so on, all the way up to sel=63, which holds in [511:504].

Your goal: Find “**x**” and build a **digital multiplexer** that fetches the **correct 8-bit treasure** based on the selector!

