1) In this assignment you are going to learn about multipliers and you are tasked with creating one to be later used as an integrated Multiplier unit in your RISC-V core. Any RISC-V processor implementation must support a base integer ISA (In our case, RV32I). In addition, an implementation may support one or more extensions. The standard instruction-set extension for multiplication, which is named “M", adds multiplication and division instructions that operate on two integer registers.

Instruction `MUL` performs an `XLEN-bit × XLEN-bit` multiplication and places the lower `XLEN` bits in the destination register. We are not going to deal with difficulties that arise with signed × signed, unsigned × unsigned and signed × signed and what happens with the higher XLEN bits, we make sure that our operands are 16-bit inputs and the complete multiplication product would fit into a 32-bit register. RV32M standard extension instruction that is needed to be implemented in your RISC-V core:

MUL

31-25	24-20	19-15	14-12	11-7	6-0	
0000001	rs2	rs1	000	rd	0110011	MUL

Multipliers are one of the most important circuits in processor design and there are multiple ways of implementing them:

1. Binary multiplier
2. Array multiplier
3. Multiplier using Shift-Register + Adder
4. Booth multiplier

Choose any **one of the options** above, research about it and then implement that design in Verilog.

Your implementation must support two 16-bit inputs. (The multiplication will result in a 32-bit value). Write a *testbench* for your module and check the results.

2 - Bonus point) Booth algorithm is a multiplication algorithm that multiplies two signed binary numbers in 2’s complement notation and is generally considered one of the fastest multiplication algorithms for hardware implementation. As you know multipliers are widely used in arithmetic units of microprocessors. Considering the condition of complete result accuracy, it is becoming impossible to further develop this unit. However, this requirement may not be needed for many applications such as multimedia signal processing and machine learning. Numerous error-tolerant applications can be found in computing by relaxing this requirement. This design principle is generally known as *approximate* or inexact computing. An approximate Booth multiplier can be created by altering the Booth encoding process, resulting in a faster computation.

Based on Booth algorithm, design an approximate multiplier that supports two 16-bit inputs using approximate computing methodology. [Hint: Refer to the paper below]

Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing IEEE Transactions on computers, 2017

Notes:

Send all your assignment related files (*Top module* and *Testbench* [.v] and [.vvp] and [.vcd] files) along with a detailed report in [.pdf] format all in one zip file to the email address of the class.

For this assignment you can work in groups of two, each participant must submit the assignment individually with their respective name and student number.

If you have any questions regarding this assignment, feel free to contact us.

Please submit your homework, simulations and projects in the following format:

Name_StudentNumber_Verilog_HW1 (BillGates_12345678_Verilog_HW1)

Good Luck!