The python code here is used to generate the input to feed into the testbench in System Verilog and generates the output to compare with them generated by System Verilog. The code was done by two students separately, the python side will have a binary input sequence file, and the student who works on system Verilog side will take that input and copy past to the inputData file. The Pout7 and Pout2 file also copied the outputdata generated by the python code. Lastly, the student in system Verilog side will simulate the test bench in system Verilog side and then a file called outValues will be created and hold all the values generated by system Verilog. Eventually a diff -b command was used to check the correctness of the system. Runsynth.tcl file contains the information of the system which need to be synthesized.

The rando\_tb\_gen\_pt\_4\_1. Py file is the python file to work with part4a, and the rando\_tb\_gen\_pt4\_2. Py is the python file to work with part4b.

Additional notes, since there was no testbench created for the 3,4,5,6 stages pipeline multiplier, thus no random input generated for those design. Indeed, the inputdata files there were 40 lines of input which were 10 sets of input sequences to simply verify the correctness of the system design. Additionally, the top-level module source files in part4b3 , part4b4, part4b5, part4b6 were accidentally named the same : part4b3\_mac.sv, but the top-level module declaration and instantiation were different eg. part4b4\_mac , part4b5\_mac. These effects were adapted in the corresponding runsynth.tcl files.