The report

Data Structures:

Unordered maps were used as well as a vector of vectors in unordered maps to store the necessary data within the code. Unordered maps were used to store the inputs and outputs of the circuit, where the key is the input and the vector of strings are used to store the outputs where the last element in the vector is the value of the input. The other unordered_map is used to store the output as the key and the vector of strings is used such that the first element is the name of the gate from which the output is produced and the rest of the vector elements store the inputs names and the last element stores the delay. Finally, the vector of vectors is used to store the events that are currently in progress and are currently processed by the simulator.

Algorithms:

All in all, 6 distinct functions were used for different purposes. For instance, the trim function was used to start the string from the non-empty character and end at a non-empty character and all the spaces from the string are removed. The split function is also used to divide and slice the string up depending on the delimiter fed into the function. The readVerilogFile function is used to read the verilog file containing the module description in Verilog. Also, the inputs and outputs as well as gate delays are read from the verilog file and entered and stored into their corresponding locations in the ordered maps. For instance, the gate name is entered in the first element of the vector in the unordered_map and the outputs are stored inside the vector of strings in the unordered_map of inputs and finally the delay is stored in the last element of the output map. This is done by looping over all the indices until the end of the string is reached and this is done for each gate with respect to the delays of each gate.

As for the value in the last element of the vector of outputs in the inputs map, the boolean variable validinputs is first checked to ensure that the inputs are valid so that a value could be calculated and stored inside the allocated space. Then, for each gate the value is calculated using addition and multiplication and is stored in its respectful space at the end. As for the readstimulfile function the stimuli file is read line by line and for each line the delay and input name as well as input value are extracted and trimmed as well as extracted and split and stored in their corresponding variables then a vector of events is used where the values inside the variables are pushed onto the vector of events to store the necessary data about each event. The function writingstimulifile checks if the vector of output corresponding to an event is empty, if this is the case then the output is not another intermediate event but is rather the final output which will be outputted to the screen. If this is not the case then it creates a new event and adds up the delay of the gate and the delay of the inputs and stores it in its corresponding place as well as the rest of event-related data.

Testing:

Testing was conducted on multiple circuits including the 5 test circuits as well as other circuits that were created to test for certain cases that apply in specific scenarios, such as when inputs consist of two letters or delays are two digits etc.

Challenges:

There were some challenges that faced us while working on the code that included adapting the code to make it accept multiple letters of the alphabet as inputs as well as accepting multi-digit delays.However, such errors that resulted from these cases were resolved.

Member Contributions:

Me, Sondos and Roaa worked on our own drafts of the code initially and later on after each one of us finished the code, we decided that Sondos' code was the one most functioning and closer to a fully functioning code. Then, Roaa worked on the cases that were not functioning in Sondos' code and added them to her code. Finally, I added the Github Actions bonus and wrote the report as well as the comments for the code.