EECS 478
Project 3 – Report

Gabrielle Guarino
gguarino

Topological Sorting:

I thought that the best way to go about topologically sorting a blif file would be to just always make sure that the nodes that fan in to another node come before it. I created a function that checks this by running through each sorted node of the circuit and checking the index of its fan in node and comparing it to the index of itself (by index I mean the index in my topological sort). I created a loop that continued looping until this check was true for all nodes. Originally I just looped through the nodes once and placed all their input nodes before the output node, but I ran across a problem where if a later node uses a previously sorted node, it could get a node placed before/after it that messed up its previously accurate sorting. That's when I decided to do the continuous loop.

To test this, I used my adder blifs, subtraction blifs, and multiplier blifs from the second project because I had a large number of them. I created a script that looped through all of these blifs from a 1 bit to a 32 bit and created the topological sort. I found that when running the program against some of my multiplier blifs I had an issue with the program seeming to stall. On further inspection, it didn't stall, it just was taking a long time to compute because some of my multiplier blifs reached sizes of 20,000+ lines.

Some changes that I made to the file system was in the circuit files I made a function that returns all the nodes in the circuit instead of just the primary inputs or primary outputs. I also created a file called topological.cpp with a corresponding header that contains all of my functions used to debug and run my topological sort.

Functional Simulation:

To simulate a blif I start by reading in the inputs and creating a deque of these nodes and their values. Later on in my debugging I came across an issue in regards to plain ONE_NODE and ZERO_NODE values, since they have a value initialized to them regardless of the input, and they are not listed as primary input. Therefore, after reading the input nodes, I add the 0 and 1 nodes to this input node list with their corresponding node names, and known values. After I have all of these nodes initialized, I loop through all the nodes in the circuit (except for the primary inputs) and check to see if all of their fan in nodes are located in my other node structure, the one that contains values. If all of the fan in nodes are there, then I am able to compute the value of the output/internal node. I do that and then add the new node with calculated value to the other node structure.

Something I wasn't sure about when making this simulator was the order that you wanted the output values to appear in, and if it even mattered at all. Originally I just outputted the nodes in whatever order they were added to the output deque. At the end, I changed it to match the order of the primary outputs given after you call circuit.getPOs().

To test this, I also used my files from the previous project. I created a cpp file that makes input files for different values. I also created another cpp file that takes in the input, and the output created from simulation, and checks that the value I computed with my simulation is the same value created when actually calculating the output given the two inputs. I originally created a script that ran the simulator and compared the outputs, but I found that the simulator output was too differently formatted, and I didn't feel like making another cpp file to morph this output into something comparable to my output. Also I wasn't on a CAEN computer for most of this project creation, so I lacked the ability to run simulator.

Some changes that I made to the file system was in the circuit files I made a function that returns all the nodes in the circuit besides the primary inputs. I didn't need the primary inputs because their values were already given. Another change I made was in the truth table files. I created a function that would return a vector of strings of the input values to the truth table. These strings were used when calculating a nodes value based on its input values. I have files simulation.cpp and a corresponding header that contain the functions used when simulating.

<u>Additional files I am submitting and their purpose:</u>

| | |
|---|---|
| check_sim.sh | - runs simulation for thousands of input files and blifs, then compares the output value to a calculated output (this will not work if you run because all my directory commands will be off) |
| check_topo.sh | - runs topological sort for all of my adder blifs, from 1 bit to 32 bits. (again, this won't work if you run of it because of the directories) |
| simulation.cpp<br>simulation.h | - my simulation functions used for debugging and running. |
| topological.cpp<br>topological.h | - my topological functions used for debugging and running. |
| check_sim.cpp | - the file that checks whether my outputted simulation values are equal to the calculated values (used in check_sim script) |