I worked on building the DAG, Gate, and Wire structures. The next step is to be able to go from a circuit representation to the DAG and then to be able to update the gates in the DAG with values from the library. Then from there we will want to convert the DAG values to the Json file format so that we can open the file from the Dynamic Circuit modeler.

I was able to make functions that go from DAG to Json file. I still need to figure out if the output if the input should be a buffer/or. I need to write a function that can convert a circuit string to a DAG. Once I finish these two small tasks, the full path will be complete.

I went back to the minGateFinder from java to fix a couple of lingering problems.

I made a mechanism that allows for swapping in gates. So if you go from a string representation for a circuit to a DAG, you can make a gate out of something in a library and just set the wires to the gates fanIn and fanOut and it will automatically remove it from the fanIn and fanOut of the gates made from the string representation. Note all functions must be done from gates not from wires. If you want to attach a wire to a gate, you call the gate’s set fanIn or fanOut function not the wire’s setTo or setFrom function.

You can also just simply remove an attachment.

I tested going from a DAG to a Jsonfile to a graph, and after some debugging, got it to work.

What’s left is being able to go from a string representation to a DAG. Then go from a library of Inputs/Repressors/Outputs, to Gates and then start swapping in things to test them out.

I made a wrapper function that incorporates everything together. I just need to specify the location of the libraries, the circuit string, and where to save the intermediate json file, and graphs will be generated. The function that creates the DAG from the string representation of the circuit and the function that gets the replacement gates needs to written still.

I tried to do a bit of work with making DAG from a string of a circuit. I used "(((((a.0).b).(a.0)).0).(c.0))" as my sample circuit. I am aiming to first turn all inputs into gates and then pair them with fan out wires. Then make repressor gates and pair them with their fan in. Also pair them in another dictionary with their fan outs. Finally I will make the final Gate an output instead of a repressor. The output will look like this:

Input fan out

{'0': ['W1', 'W2', 'W3', 'W4'], 'IN1': ['W5', 'W6'], 'IN2': ['W7'], 'IN3': ['W8']}

Repressor fan in

{'G6': '(W8.W4)', 'G5': '(W12.W3)', 'G4': '(W10.W11)', 'G3': '(W6.W2)', 'G2': '(W9.W7)', 'G1': '(W5.W1)'}

Repressor fan out

{'G6': ['W15'], 'G5': ['W14'], 'G4': ['W13'], 'G3': ['W12'], 'G2': ['W11'], 'G1': ['W10']}

Output fan in

{'Y': '(W13.W14)'}

Wires from the same place

{'W15': ['W15'], 'W14': ['W14'], 'W13': ['W13'], 'W12': ['W12'], 'W11': ['W11'], 'W7': ['W7'],

'W5': ['W5', 'W6'], 'W1': ['W1', 'W2', 'W3', 'W4'], 'W9': ['W9', 'W10'], 'W8': ['W8']}

The output looks a bit weird because the same wire is a fan in for multiple gates, but I would go through and replace the duplicates with their similar wire using the dictionary of wires from the same place. Then I would pair the wires with to and from gates with these dictionaries.

The problem is that W4 is a ghost wire. Using the plan specified I would be left with no ToGate for W4. I would need to make some way of pairing the wires and then remove any ghost wires like this. Then I would shift the number portion of the wire names appropriately.

Once I have the wires paired with their to and from, I can make the connections between gates and wires more easily.

I need to figure out how to split a string by multiple things rather than just by “.”.

I got the wires to be uniquely paired with gates to have unique fanIn and fanOut wires and removed the ghost wires. The wire numbering isn’t in order as I had hoped because W12 would be placed above W2 by sorting, so I didn’t bother sorting.

I think I was able to successfully convert a string of a circuit to a DAG. But this only works for circuits with nor. It will not work for circuits with ~&+^ etc.

The next step would be to make gates from libraries. Then they will be swapped in where necessary. And the graphs will be produced.