ECSE318 Homework 5

Benjamin Scholar

CaseID: bbs27

Email: scholar@case.edu

Files (all cpp files have a corresponding header file in the include directory):

- src/sim.cpp – main file for simulator
- src/parser.cpp – main file for parser
- src/gate.cpp – holds gate class and other types
- src/Top.cpp – contains top sort algorithm implementation
- src/simulator.cpp – contains simulator logic
- include/lut.h - contains all look up tables
- CMakeList.txt – cmake configuration file
- Test.v – small input file
- Test2.v – large input file
- Test_gen.py – script for generating large input vectors

Build instructions (from root of project):

```
sudo apt-get install cmake libboost-all-dev
mkdir build
cd build
cmake ..
make
```

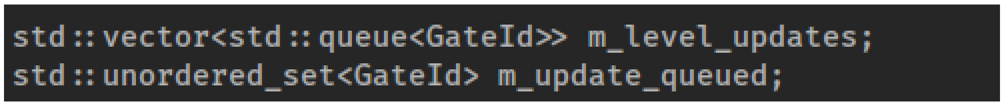
The binaries should be produced within the build directory.

Run instructions:

```
verilog_parser <input_file> <output_file>
verilog_sim  <gate_file> <input_vector_file> <output_file>
```

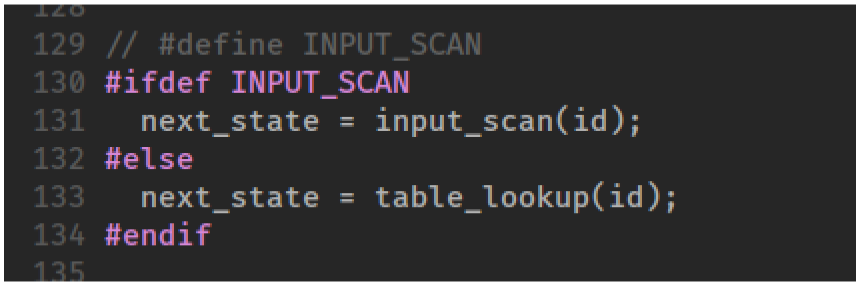
Approach:
To calculate the levels of the gates, I used a topological sort to get the correct order of the gates. This order is then used to loop through each gate in the correct order, setting the level of the current gate to one plus the max of the input gates' levels. This results in the correct level

categories being created.
Instead of the pointer method laid out by the homework document, I decided to use a FIFO queue for each level of the circuit as well as an unordered_set (from the c++ standard library) that keeps track of which gates have already been queued. This serves the same purpose as the method laid out in the homework document. These data structures can be seen here:



```
std::vector<std::queue<GateId>> m_level_updates;
std::unordered_set<GateId> m_update_queued;
```

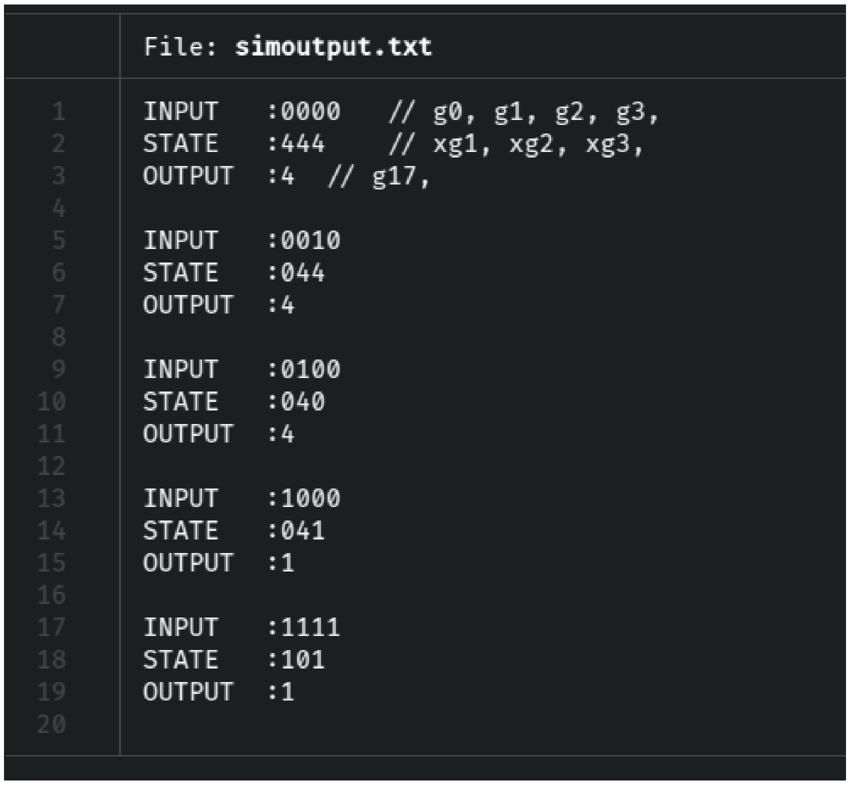
The two different evaluation methods are handled by the define statement of line 129 of the simulator.cpp file. Uncomment this line to use the input scanning evaluation method.



```
128
129 // #define INPUT_SCAN
130 #ifdef INPUT_SCAN
131   next_state = input_scan(id);
132 #else
133   next_state = table_lookup(id);
134 #endif
135
```

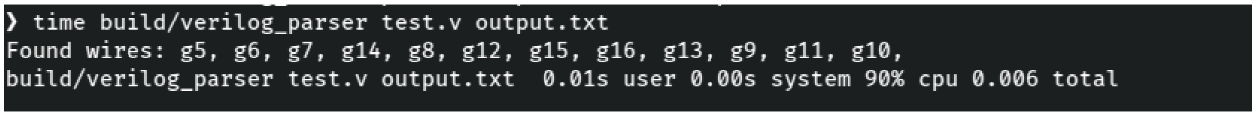
Results:

Correctness**:**
The simulated output from the provided, small input file matches the corresponding output from the homework document. Thus I am led to believe that the simulator is behaving correctly.

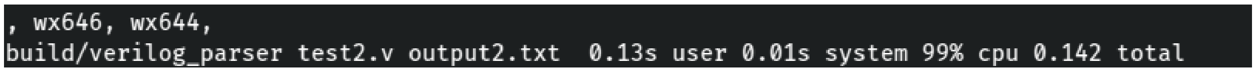


```
File: simoutput.txt

INPUT   :0000   // g0, g1, g2, g3,
STATE   :444    // xg1, xg2, xg3,
OUTPUT  :4  // g17,

INPUT   :0010
STATE   :044
OUTPUT  :4

INPUT   :0100
STATE   :040
OUTPUT  :4

INPUT   :1000
STATE   :041
OUTPUT  :1

INPUT   :1111
STATE   :101
OUTPUT  :1
```

Parser times:

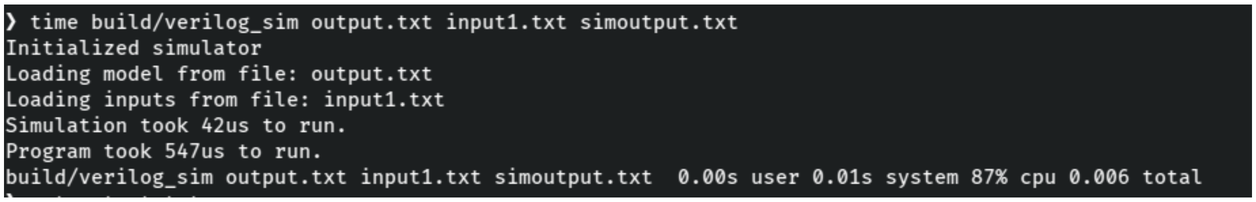


```
❯ time build/verilog_parser test.v output.txt
Found wires: g5, g6, g7, g14, g8, g12, g15, g16, g13, g9, g11, g10,
build/verilog_parser test.v output.txt  0.01s user 0.00s system 90% cpu 0.006 total
```

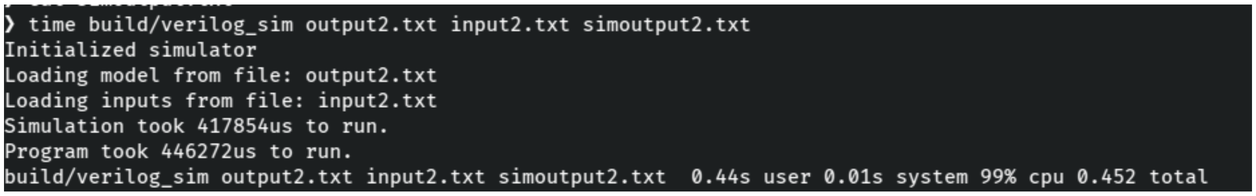


```
, wx646, wx644,
build/verilog_parser test2.v output2.txt  0.13s user 0.01s system 99% cpu 0.142 total
```

Simulator (input scan) times:

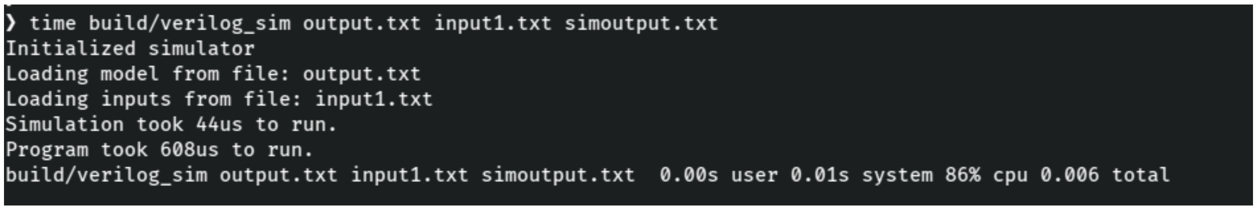


```
❯ time build/verilog_sim output.txt input1.txt simoutput.txt
Initialized simulator
Loading model from file: output.txt
Loading inputs from file: input1.txt
Simulation took 42us to run.
Program took 547us to run.
build/verilog_sim output.txt input1.txt simoutput.txt  0.00s user 0.01s system 87% cpu 0.006 total
```

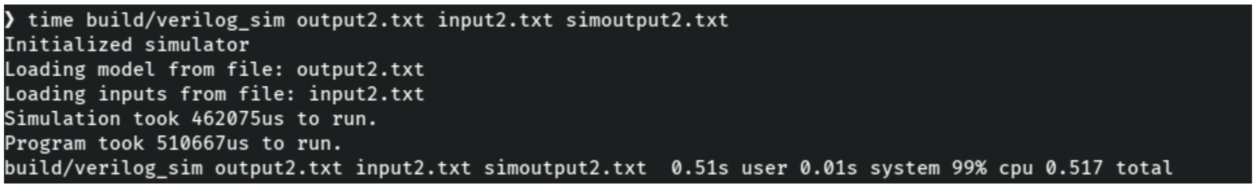


```
❯ time build/verilog_sim output2.txt input2.txt simoutput2.txt
Initialized simulator
Loading model from file: output2.txt
Loading inputs from file: input2.txt
Simulation took 417854us to run.
Program took 446272us to run.
build/verilog_sim output2.txt input2.txt simoutput2.txt  0.44s user 0.01s system 99% cpu 0.452 total
```

Simulator (table lookup) times:



```
❯ time build/verilog_sim output.txt input1.txt simoutput.txt
Initialized simulator
Loading model from file: output.txt
Loading inputs from file: input1.txt
Simulation took 44us to run.
Program took 608us to run.
build/verilog_sim output.txt input1.txt simoutput.txt  0.00s user 0.01s system 86% cpu 0.006 total
```



```
❯ time build/verilog_sim output2.txt input2.txt simoutput2.txt
Initialized simulator
Loading model from file: output2.txt
Loading inputs from file: input2.txt
Simulation took 462075us to run.
Program took 510667us to run.
build/verilog_sim output2.txt input2.txt simoutput2.txt  0.51s user 0.01s system 99% cpu 0.517 total
```

Here is a table of the recorded times of each program. Each entry is logged in seconds. These values were recorded using the *time* linux command.

Input file	Parser Time	Simulator Time (input scan)	Simulator Time (table lookup)
test.v (small input file)	0.006	0.006	0.006
test2.v (large input file)	0.142	0.452	0.517

It can be seen that the input scanning method of evaluation is faster in my implementation, and significantly so. I believe this is because the input scanning method returns early when it sees its controlling value, so less iterations are needed to get the final value. The table lookup method, on the other hand, has to do many lookups to solve 1 gate potentially and has to use all inputs. Surprisingly, the method laid out in the lab has very similar performance to the previous method that I used (mentioned in email).