Project 1: Lexical Analysis

Adam Sumner
ECE 449

Initial Release Due: 9/11/2015
Final Release Due: 9/18/2015

1 Initial Release

The features implemented in the initial release included:

- Line Parser Loop
- Character Parser Loop
- Comment Recognition
- Space Recognition
- **SINGLE** token Recognition
- **NAME** token Recognition
- File Output
- Error Handling

These features were mostly chosen because they are the basis of the Lexical Analysis functionality in the simulator. The two main loops involved in the parser provided a solid skeletal structure so that detail specific recognition, such as recognizing comments, name tokens, etc., could be included later. Subsequently, once the basis was created, the **SINGLE** and **NAME** tokens were easily implemented using if statements within the character parser loop. As stated previously, these features provided the backbone of the lexical analysis functionality, so it was crucial to design this portion of the simulator for the initial release.

2 Test Cases

When designing the test cases, it was critical that the `*.evl` test files accurately represented proper examples of real circuits. This means that in all test cases, **NAME** tokens, **SINGLE** tokens, and comments were a must to include in each

file. Furthermore, the structure of each `*.evl` test file had to differ from each other to make sure the lexical analysis functionality could handle general case circuits, and not only be able to tokenize certain circuits. A variety of structures were used in the `*.evl` test files to ensure that tokenization was being properly processed.

3 Final Release

The final release involved implementing the recognition of **NUMBER** tokens. Because so much structure was designed and implemented during the initial release, this feature was easily able to be added to the character parser loop and file output functionality. Using the `cctype` library, the function `isdigit(int c)` was used to check for numbers in the file. Because of the modularity of the `if` statements within the character parser loop, the required code to recognize **NUMBER** tokens was easily injected into the previously developed initial release code, and passed all 10 EasyVL test cases, along with the 5 custom test cases designed for the initial release.