Overview

BooLeX originated in CPSC 439: Software Engineering, Spring 2014 as a project to address the difficulties new computer science and electrical engineering students face in first learning the laws of digital circuitry and Boolean logic. BooLeX is an educational tool that allows students to assemble virtual circuits and observe their performance on various input signals. By offering a visual aid to students learning about digital logic and circuitry, BooLeX enables students to test their own circuit designs and solidify the fundamental concepts of Boolean algebra.

Why BooLeX?

The tools currently available to students studying digital logic are unfortunately lacking. Some tools such as Logic.ly suffer unexpected behavior in certain sequential circuit designs, while others such as CircuitLab operate on a lower level than the abstracted principles of digital logic, and though offer more detailed simulations, can be overwhelming for new students struggling with the concepts of Boolean algebra. Most simulators such as Logic.ly and CircuitLab further suffer from the slow rate at which the simulations must be built, as the circuits must be built using a drag-and-drop graphical interface rather than using a text-based interface that allows students to simply write a description for the circuit they would like to simulate.

BooLeX uses a novel evaluative model by translating circuits created in its drag-and-drop interface into a simple domain-specific language (DSL) that is evaluated on the back end. This linguistic evaluation allows circuits to be simulated visually in a similar way to the other simulators, but offers the additional benefit of allowing users to directly express circuits in the BooLeX DSL so they can express more complex circuits more rapidly and can build more intricate designs.

Recognizing the advantage of directly expressing circuit designs over using a drag-and-drop simulator, some electrical engineering courses rely on hardware description languages such as VHDL to teach students to construct more complex
circuits. However, many students encountering code for the first time find the fully featured VHDL overwhelming and its pedagogical utility is limited. The BooLeX DSL, in contrast, is designed to be simple and intuitive, and is stripped down to the most basic essentials necessary for constructing and analyzing circuits in an educational context.

Previous Work

The first version of BooLeX was produced as a team project for CPSC 439 by Abhishek Chandra, Daniel Dickstein, Graham Kaemmer, and Alexander Reinking. We were able to successfully simulate combinatorial and sequential circuits, including half and full adders, SR and D latches, and a D flip-flop. The front-end interface is crude and has a few minor bugs, but is largely functional. The back end seems to work in most cases, but some of the more ambitious circuits we had attempted failed, and further investigation into both the front and back ends is required. When CPSC 439 drew to a close, our project was not yet ready for the classroom, but had come a long way. My goal in the deliverables for CPSC 490 below is to bring this project to a presentable conclusion.

Project Goals

In continuing work on BooLeX, the primary goal is to build an indispensable learning tool. To that end, it must be simple, intuitive, and easy to use, but also must be equipped with the features that would make it a dependable and worthwhile tool for teachers to introduce in the classroom. For electrical engineering departments that rely on hardware labs, this means that work done on BooLeX should be able to be ported to VHDL simulators such as Quartus II. For computer science departments that focus on theory and Boolean algebra, complex logic must be easily expressed and evaluated, ideally in a familiar REPL environment, without requiring a deep understanding of hardware. Finally, to be most useful for collaborating students, BooLeX must allow designs to be easily shared among peers. To balance all of these project goals and to make BooLeX a tool that can be easily used in courses such as CPSC 112, CPSC 201, and EENG 201, I propose the following deliverables.
Deliverables

The deliverables listed here are noted in general, overall terms. For more specific goals and progress over the course of the term, please refer to the BooLeX Trello board.

1. Address lingering bugs. There are a number of bugs that we identified but did not have the time to pursue and eliminate when the project drew to a close, so before adding features I want to be sure the core product is working properly.

2. Improve the front-end user experience. Aside from bugs, the front end also lacks several features without which the interface is cumbersome and unintuitive. Since I was not the original front-end designer, this will require me to learn CoffeeScript and the various other JavaScript frameworks employed by the front end in order to successfully make the required alterations.

3. Build a user interface to allow users to directly write BooLeX code and execute it in a REPL environment.

4. Allow users to create integrated circuits by “black-boxing” parts of their circuit designs, and allow the integrated circuits to be used in multiple locations. Some functionality has been built out for this feature, but it is currently broken.

5. Allow users to import / export integrated circuits from the drag-and-drop user interface.

6. Transition the back end from Java to Scala. This will allow me to take advantage of the functional features Scala offers such as pattern matching, which will simplify the evaluation of the abstract syntax tree generated for a given circuit and should enable greater back-end modularity.

7. Write a BooLeX to VHDL compiler by building a different evaluator for the abstract syntax tree than the one used by the live simulator. The language features made available by transitioning to Scala should simplify this task, but it will likely still be the most time-consuming.

8. To maximize its accessibility, the BooLeX tutorial should be expanded so that it covers all the basics of digital logic assuming no prior knowledge or background.