Software Design Document

Interactive Logic

6.1. Introduction

When learning the basics of electrical engineering, it can often be difficult to format information in a way that is easy to teach and learn.

From taking electrical engineering courses at LMU, there were a number of things I found frustrating with the digital logic software we used as a class.

Knowing that other students were frustrated by these same shortcomings, I decided to rebuild the basic functionality of this software from scratch, with simple intuitive design as the primary goal.

6.1.1 System Objectives

My project is to create a standalone desktop application that combines:

* Intuitive design
* Straightforward interactions
* A suite of basic features and functionality for teachers and students to visualize digital logic circuits effectively.

6.1.2 Hardware, Software, and Human Interfaces

Standalone application built in Python 2.7 using the PyGame graphics library.

Digital logic simulation from the seven basic gates

Interactive input components (switches)

Clocks

Simple output lights

Truth table generation

Timing diagram generation.

Save / Load circuits

6.2 Architectural Design

6.2.1 Major Software Components

Interactive Logic is a standalone application for Windows, Mac & Linux

Dependencies are an install of Python 2.7

The application runs on the Pygame graphics library

6.2.2 Major Software Interactions

6.2.3 Architectural Design Diagrams

6.3. CSC and CSU Descriptions

6.3.1 Class Descriptions

InteractiveLogic.py

gates.py

generator.py

saveload.py

6.3.1.1 InteractiveLogic.py

This is the brains of the program. It controls all the GUI interaction, buttons, and visual representations of circuits

6.3.1.2 gates.py

This file runs the digital logic simulation real-time as the circuit is built. It contains the logic for each individual gate and runs the distributed network. It also does tree traversal for the purpose of truth table data generation.

6.3.1.3 generator.py

This file controls the creation of the truth table and timing diagram windows. It uses python multiprocessing. The timing diagram window listens for state changes from InteractiveLogic.py and updates the diagram.

6.3.1.4 saveload.py

This file handles the saving and loading of circuits. It accomplishes this by pickling the data into a text file. Pickling doesn’t support Pygame surface objects though, so I have to iterate through everything and remove them on save and restore them on load.

6.3.2 Detailed Interface Descriptions

6.3.3 Detailed Data Structure Descriptions

6.3.4 Detailed Design Diagrams