Requirements and Analysis Document for Open Logic Gate Simulator

Jimmy Andersson
Jacob Eriksson
Martin Hilgendorf
Elias Sundqvist

1 Introduction

A huge benefit of choosing a software based career is that you can work from anywhere as long as you have a laptop and an internet connection. However, when it comes to hardware this is not always the case. In particular, when it comes to building computers it can get expensive real quick. For those students interested in learning how computers work, but do not want to invest in real hardware, there should be an application that helps understand how computers work.

Our approach to the problem is to develop a simulation software called Open Logic Gate Simulator. The purpose of this software is to simulate how low-level logic components work inside of a computer. It will try to emulate the reality close enough for the user to understand it on a logical level, without introducing the complexity of electronics theory to the user. It can be used by both students, teachers and hobbyists to enhance the learning experience, and to get a better intuitive understanding of how logic circuits work.

In practice, this is done by providing an interactive workspace where basic logic gates can be placed, moved and connected. Furthermore, more advanced components (such as clocks) can be placed, and the behaviour of the circuit can be simulated over time, allowing for the construction of more advanced circuits, such as small microprocessors.

The application will be a standalone desktop application with a graphical user interface for the Windows / Mac / Linux environment. It will be scalable and adapt to different screen sizes.

1.1 Definitions, acronyms, and abbreviations

- **Workspace** – The area from which you edit your currently open circuit
- **Circuit** – The set of components and wires shown; essentially the save file
- **Component** – A part of a circuit used to perform operations on a logic state
- **Wire** – A part of a circuit used for propagating a logic state between components
- **Port** – A part of a component that is used to input or output an logic state
- **GUI** – Graphical User Interface
- **MVC** – Model-View-Controller, a design pattern well suited for application with a GUI
- **Java** – The multi-paradigm programming language used to build this application
- **Checkstyle** – An automated tool that help developers stick to a harmonized style of coding
- **PMD** – An automated tool that help developers find suboptimal and unused code
- **SpotBugs** – An automated tool that help developers find potential bugs
- **JaCoCo** – Java Code Coverage, an automated tool that collects information on what code in a code base is covered by the current unit tests

2 Requirements

2.1 User Stories

As a User I want to design and simulate basic logic circuits to get an intuitive sense of what is happening at the lower level of computing.

---

1. As a User I want to see all components, so that I can get an overview of what's available.

- There should be a scrollable list containing all the components.
- Looking at the list, the User should be able to tell what the name of each component is.
- Looking at the list, the User should be able to tell how each component looks.
- The width of the list should be resizable so the User can better adjust it to their liking.
- The components should appear in a sensical order.

1.1. As a User I want to be able to place components on a workspace, so that I can use them.

- There should a big, initially empty, space that will act as the workspace.
- If the user drops the component onto the workspace, the component should stay there.
- If the user drops the component anywhere else, nothing should happen.
- While dragging around the component, the mouse should indicate if the current place is okay to drop the component on.

1.2. As a User I want to be able to get a sense of how big a component actually is and where exactly it will land before placing it, so that I can place components nicely and efficiently.

- If the User drags one of the components in the list towards the workspace, the component visual should follow along the mouse. And the component should be visualized in the same size, and at the same position as it will appear if dropped.

1.3. As a User I want to be able to see details about each available component, so that I can gain insight into what they're doing.

- The User should be able to hover over any component in the list and get a short description of what it does.

2. As a User I want to be able to connect components, so that I can create circuits.

- Each component's ports should be visible so the User can see what can be connected.
- Clicking on 2 ports of opposite types in sequence, should create a wire between them.
- When clicking on the first port in the sequence, all ports of the opposite types that are available to connect to, should highlight. This will make it easier for the User to quickly find ports that are available.
- Pressing ESC or clicking somewhere in the background should cancel the wire creation and thus remove the port highlighting.

2.1. As a User I want to be able to connect one source to several other components so that I can create more compact circuits and minimize unnecessary components.

- Connecting wires from a single output port to several input ports should be possible.

3. As a User I want to get visual feedback from connections that are active (1/on) and ones that are not (0/off), so that I get some live feedback on what is happening in my circuits.

- A connection currently in a HIGH state should be drawn in red.
- A connection currently in a LOW state should be drawn in black.
- Any connection that is in an UNDEFINED state should be marked by drawing it yellow.
- The colors should be updating live as the simulation is happening.

4. As a User I want to be able to remove components, in case I mess up.

- It should be possible to select components and wires by clicking on them, so that the User can choose what to remove.
- It should be possible to select/deselect multiple components/wires by SHIFT-clicking them.
- It should be possible to select an area of stuff to select by dragging the mouse across the background, so that the selection process can be sped up if there are a lot of stuff to select.
- Selecting one or several wires and components and clicking the BACKSPACE or DELETE key should remove them from the workspace.
- Removing a single component should also remove any and all wires that are dependent on the existence of that component.

5. **As a User I want to be able to move around components, so that I can reorganize my circuits as they grow.**

- It should be possible to move around components by dragging them, wires should follow.

6. As a User I want to have a component that gives a signal source, so I can power my circuits.

- A signal source component should be part of the basic components used to provide a circuit with a state to propagate.

6.1. As a User I want to be able to toggle these signal sources, so that I can also use it as a GND source, and interactively toggle it.

- Clicking on a signal source component should toggle its output between HIGH and LOW.

7. **As a User I want to have a pulse clock so that I can produce dynamic circuits.**

- There should be a pulse clock that produces a 1Hz HIGH/LOW signal as its output.

8. As a User I want to have a access to logic gates such as AND, OR, NOT, XOR, NOR, NAND so that I can perform common logic operations. This is essential for creating any complex and realistic circuit.

- There should be a component for each of these logic gates, ready to use from the component list.

9. **As a User I want to be able to save/load my circuits so that I can continue my work when I have time, and don't have to do everything in one sitting.**

- There should be a menu bar with the options save, save as and load.
- Save and save as should save the currently open workspace.
- Load should open a dialog that asks for a save file to open as a workspace.

10. **As a User I want to have a access to flip-flop latches so I can quickly setup circuits that store various states.**

- There should be SR, T, D and JK flip-flop latches, ready to use from the component list.

11. **As a User I want to have a basic diode component so that I can easily observe binary output states.**

- There should be a diode component that turns bright when it receives a HIGH input value, and should be dark otherwise.

12. **As a User I want to have easy access to multiple workspaces so that I can switch between projects quickly.**

- Each workspace should be contained in a tab at the top of the interface.
- Adding or opening a new workspace should also open a new tab in the editor area, allowing for quick and easy switching between circuits.

13. **As a User I want to be able to start and stop the simulation, so that I can see what is going on in the circuit at a specific point in time.**

- There should be a start/stop button in the graphical user interface, that will start and stop the dynamic components from updating when pressed.

14. **As a User I want to be able to pan across the workspace so that I can utilize more space for my circuits.**

- The workspace area should act as a sort of camera into a bigger workspace, and when the User drags along it while holding down the CTRL modifier, it should pan the position of the camera.

15. **As a User I want to be able to zoom in and out on the workspace to get a more detailed view or an overview of my circuits.**

- Scrolling the mouse wheel while hovering over the workspace area should zoom in/out the focus of the camera.

16. **As a User I want to be able to delay my signals to ensure that some signal reach some destination non-instantaneously.**

- There should be a delay component that delays all state changes passed into it by one second before it outputs that same value.

17. **As a User I want to have segment display components so I that can show numbers.**

- A 7-segment display will give the user the possibility to control the lighting of individual segments on the display by sending a state to each of them.
- A hex display should give the user the possibility to display a hexadecimal number by sending a 4 bit binary number to it.

18. **As a User I want to have access to a basic counter component so that I can conveniently use the segment display.**

- There should be a counter component that increments its count on high flanks and outputs a 4 bit binary mod 16 number.
- The output should overflow and reset to 0b0000 when the counter hits multiples of 16.

19**. As a User I want to have a note component so I that can trigger sound queues.**

- There should be a note component that plays a sound when its single input receives a high flank (I.e. goes from UNDEFINED/LOW -> HIGH).

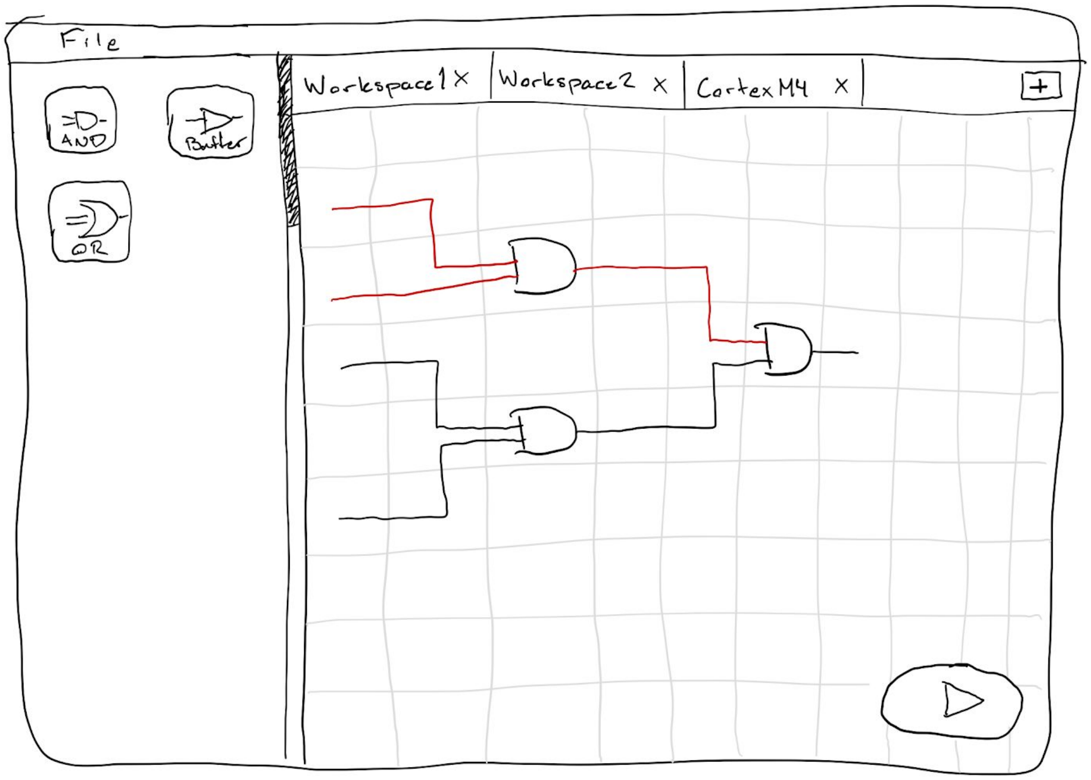
20. **As a User I want the power in my circuits to flow realistically I.e. nearby components and wires should be affected before those far away.**

- The simulation should perform logic state updates in a BFS approach. In layman's term this roughly means that the power needs to expand through circuit, similar to how waves expand from the water surface where a water drop lands.

21. **As a Team we want a proper CI workflow so that all code in the main development branch is ensured to build and follow our desired standards.**

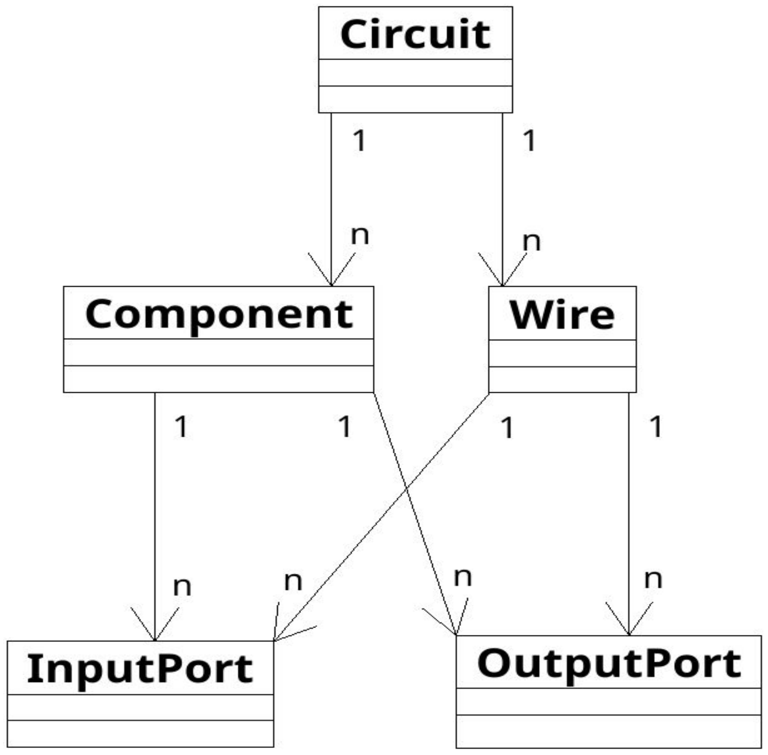
- Setup the main development branch as a protected branch. This way all code added to this branch have to go through PR/CI which means it must follow our requirements.
- Add a Travis setup that builds the project and runs the tests.
- Add Checkstyle to the build.
- Add SpotBugs to the build.
- Add PMD to the build.

2.2 User Interface



This is the main and only view of the application.
At the top we have a menu bar which allows for saving/loading circuit files and closing the program.
On the left-hand side we have a list of components that can be dragged and dropped onto the circuit.
In the center stage we have the actual circuits (each tab at the top represents a single circuit, and it is possible to switch between). Circuits can be panned/zoomed and you can move components around, as well as connect them together with wires.

3 Domain model



Judging from the domain model it isn't totally obvious how the component and wire actually communicate. We discussed this with our supervisor (after the presentation feedback) and ended up keeping it this way. The options would have been to exclude the ports all together and draw double-arrows between Component and Wire, but this gives false information as the Wire and Component don't actually know about each other. We could have abstracted it to Port but any domain expert would very likely have disagreed as input port and output port serve very different purposes in a real life circuit.

3.1 Class responsibilities

- Circuit: Simulates the power flow between wires and components. Power flow in this context is synonymous with logical state.
- Component: Modifies the logic state of output ports, often based on input ports' logical states.
- Wire: Transmits logic states from all connected output ports to all connected input ports.
- InputPort: Reflects the logic state of the connected wire, without allowing modification to the wire's logic state.
- OutputPort: Affects the logic state of the connected wire.