MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF REPUBLIC OF MOLDOVA
TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

Computer Architecture

Laboratory work 3 : Exercises in Logisim

Elaborated:
st.gr. FAF-211 Grama Alexandru
Verified:
asist.univ. Vladislav Voitcovschi

Chișinău, 2023

Content

Introduction

Logisim is ”An educational tool for designing and simulating digital logic circuits, featuring a simple-to-learn interface, hierarchical circuits, wire bundles, and a large component library. As a Java application, it can run on many platforms.” When learning computer architecture and logic circuits, you will need a real-world, graphical example of what you are studying. Text and diagrams only go so far. A helpful tool for designing and simulating logic circuits is Logisim.

Because the tool lets you create large circuits from smaller circuits, you can design entire CPUs using Logisim. Further, the tool will run on any computer!

The interface itself is very intuitive and the use of color-coding of wires and elements allows for easy analysis and testing of circuits. You can also save the completed file as an image, or as a .circ file (core to Logisim).

The main window consists of the following items:

- Toolbar: contains short cuts to several commonly used items
  - The simulation tool: shaped like a hand, is used in simulation mode to alter input pins.
  - The design tool: is used while designing the circuit.
  - The input pin: green circle surrounded by a square box, is used to send a signal through a wire. When placing the input on the canvas it initializes the input to logic 1 or 0. The number of bits can be increased in the Attribute Table.
  - The output pin: green circle surrounded by a circular shape, is used to observe the output from a gate or a block. The output pin toggles in real time as long as the simulation is enabled from the menu bar: Simulate ¿ Simulation Enabled
- Explorer Pane: This pane contains the list of wiring, gates, multiplexers and other components that are available for digital design in Logisim.
- Attribute Table: Gives detailed attributes of digital design components (e.g., AND, OR, XOR gates). The attribute table allows you to alter the number of inputs/outputs that a digital component may have.
- Canvas: The canvas is the area for you to create your digital circuits. In this area you may xsimulate your circuits while designing in real time.

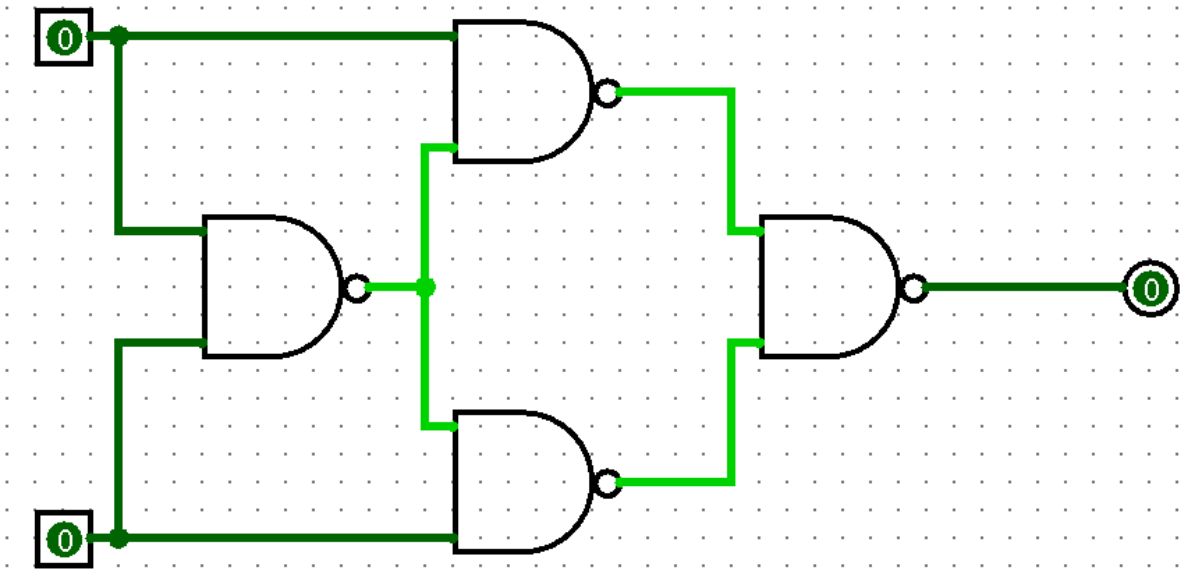
Objectives

1. Familiarize yourself with the basic components of digital circuits such as logic gates, flip-flops, and multiplexers.
2. Develop skills in designing and testing simple and complex digital circuits using Logisim.
3. Learn how to use Logisim's simulation tools to analyze circuit behavior and identify potential problems or errors.
4. Gain proficiency in creating custom components in Logisim to support the design and simulation of more advanced digital circuits.
5. Build experience in using Logisim to design and simulate real-world digital systems such as computer CPUs, memory units, and control units.
6. Practice using Logisim in a team setting to collaborate on digital circuit design projects and ensure that designs meet performance, timing, and other requirements.
7. Develop an understanding of how digital circuits are used in different applications such as telecommunications, automation, and control systems, and use Logisim to design circuits that meet specific requirements in these fields.
8. Experiment with advanced Logisim features such as hierarchical design, subcircuits, and Verilog import/export to improve your digital circuit design skills.
9. Participate in online communities or forums related to Logisim to learn from other users, share knowledge, and collaborate on digital circuit design projects.
10. Continuously evaluate and improve your Logisim skills by experimenting with new design techniques, exploring new features, and taking on increasingly challenging digital circuit design projects.

Exercise 1E

Implement a two-input XOR gate using NAND gate.

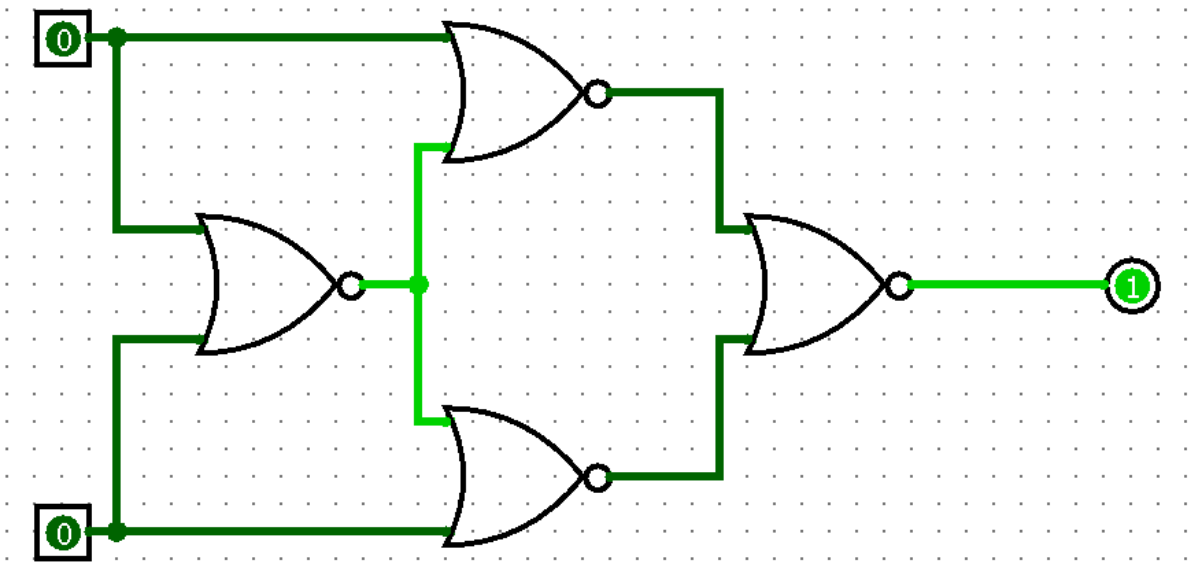
When both inputs are low (logic 0), both NAND gates produce high (logic 1) outputs. When one input is low and the other is high, one NAND gate produces a high (logic 1) output, while the other produces a low (logic 0) output. When both inputs are high, both NAND gates produce low (logic 0) outputs.



Exercise 2E

Implement a two-input XNOR gate using the NOR gate.

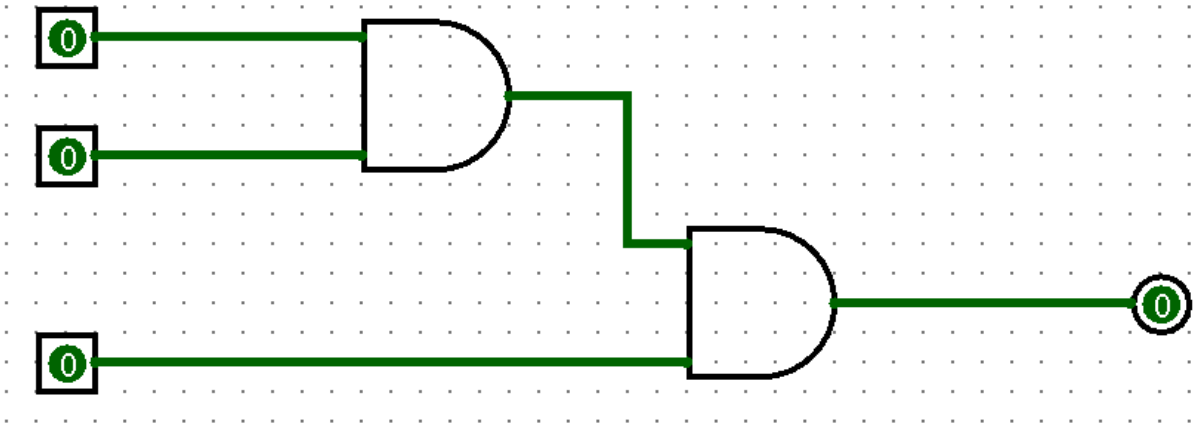
When both inputs are the same, the first inverter inverts one of the inputs, while the other input passes through the inverter unchanged. Thus, one input to the NOR gate will be low (logic 0) while the other input is high (logic 1), causing the output of the NOR gate to be high (logic 1). The second inverter then inverts the output of the NOR gate, giving a final output of high (logic 1).



Exercise 3E

Implement a three-input AND gate using the AND gate.

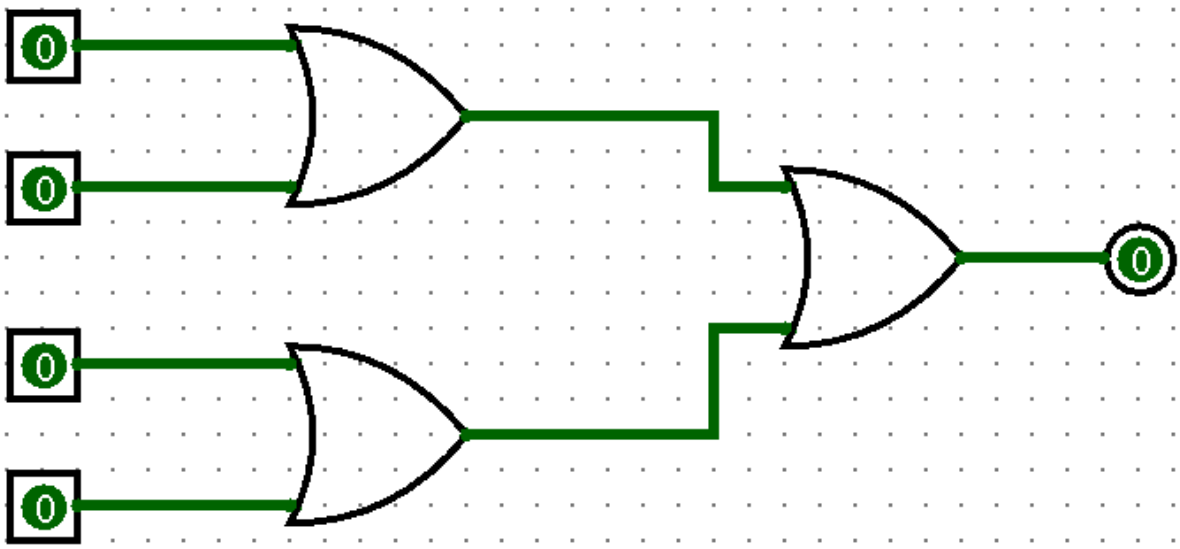
When both A and B are 1, the first AND gate produces a 1 at C. Then, when D is also 1, the second AND gate produces a 1 at E, indicating that all three inputs are 1. If any one of the inputs A, B, or D is 0, then the output of the corresponding AND gate will be 0, and the final output E will be 0.



Exercise 4E

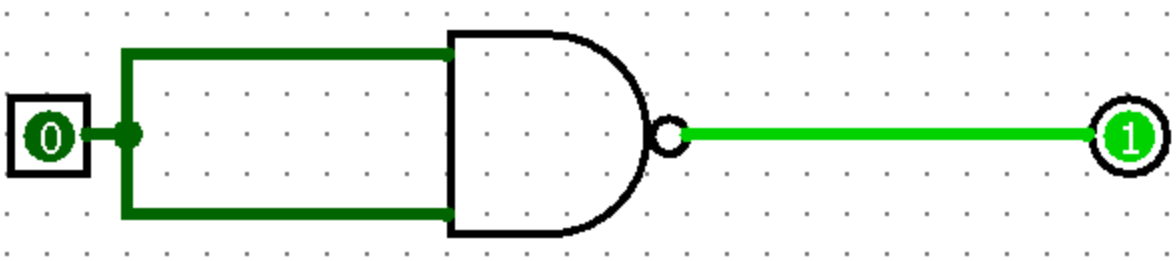
Implement a four-input OR gate using OR gate.

When any of the inputs A, B, D, or F is 1, the corresponding OR gate will produce a 1. If none of the inputs are 1, all three OR gates will produce a 0, and the final output G will also be 0. This implementation uses three OR gates, but it is also possible to implement a four-input OR gate using fewer gates by cascading multiple two-input OR gates together.



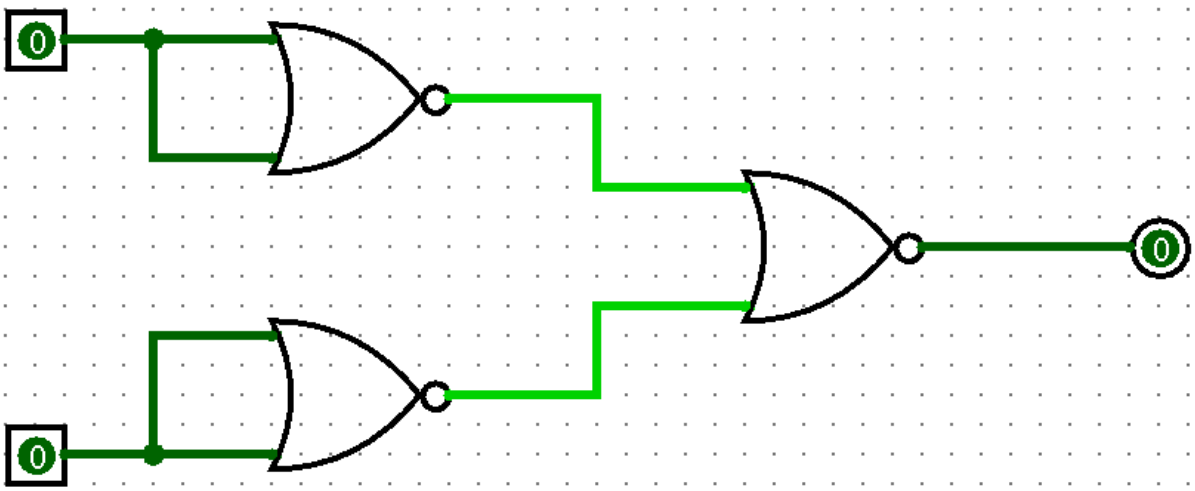
Exercise 5E

Implement a two-input NOT gate using NAND gate. In this circuit, both inputs (A and B) of the first NAND gate are connected together, so the first NAND gate will output a low signal (0) only when both inputs are high (1), which corresponds to the second row of the truth table. The output of the first NAND gate is then connected to the input of the second NAND gate, which acts as an inverter and outputs the opposite of the first NAND gate, which corresponds to the first row of the truth table. Therefore, this circuit implements a two-input NOT gate using a NAND gate.



Exercise 6E

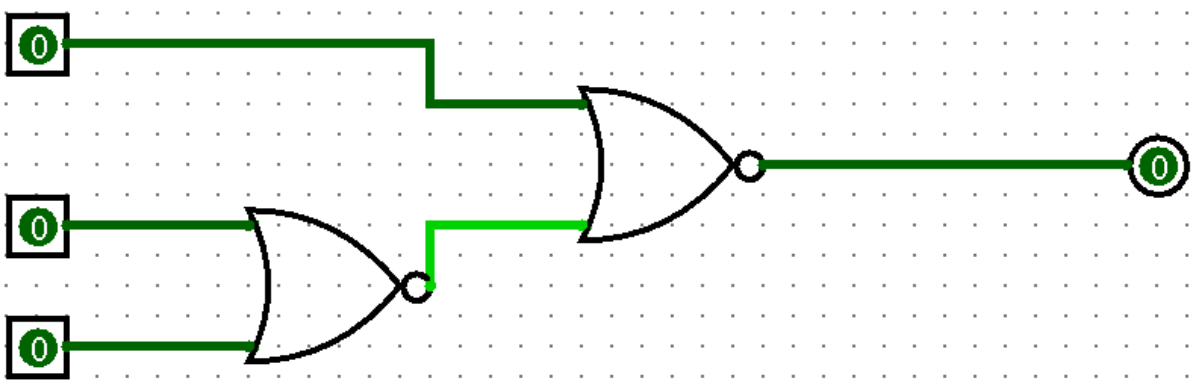
Implement a two-input AND gate using the NOR gate. In this circuit, both inputs (A and B) of the NOR gate are connected together, so the NOR gate will output a high signal (1) only when both inputs are low (0), which corresponds to the fourth row of the truth table. The output of the NOR gate is then connected to an inverter, which will output the opposite of the NOR gate, which corresponds to the first row of the truth table. Therefore, this circuit implements a two-input AND gate using a NOR gate.



Exercise 7E

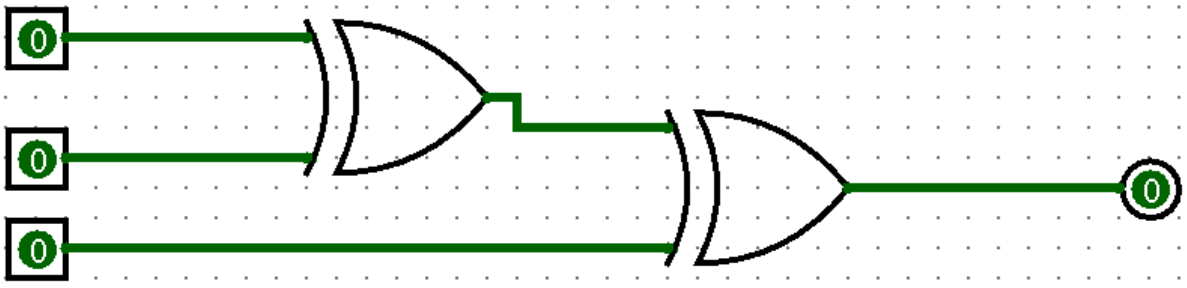
Implement a three-input OR gate using the NOR gate. In this circuit, each input (A, B, and C) is connected to an inverter, which will invert the input signal. The outputs of the inverters are then connected to the inputs of a NOR gate. The NOR gate will output a high signal (1) only when all inputs are low (0), which corresponds to the first row of the truth table. The output of the NOR gate is then the output of the

three-input OR gate, which corresponds to the truth table for the three-input OR gate. Therefore, this circuit implements a three-input OR gate using a NOR gate.



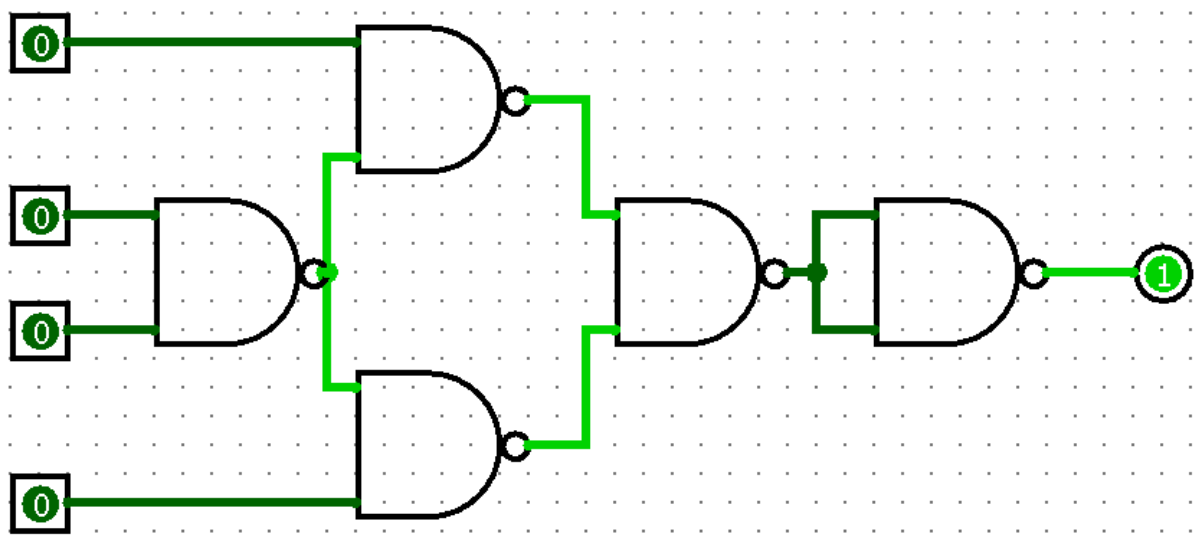
Exercise 8E

Implement a three-input XOR gate using XOR gate. In this circuit, the first two inputs (A and B) are connected to a two-input XOR gate, and the last two inputs (B and C) are connected to another two-input XOR gate. The outputs of these two XOR gates are then connected to an AND gate. The AND gate will output a high signal (1) only when both inputs are high (1), which corresponds to the fourth and eighth rows of the truth table. Therefore, the output of the AND gate will be high (1) only when the two XOR gates output different signals, which corresponds to the first, second, third, and seventh rows of the truth table. The output of the AND gate is then the output of the three-input XOR gate, which corresponds to the truth table for the three-input XOR gate. Therefore, this circuit implements a three-input XOR gate using XOR gates.



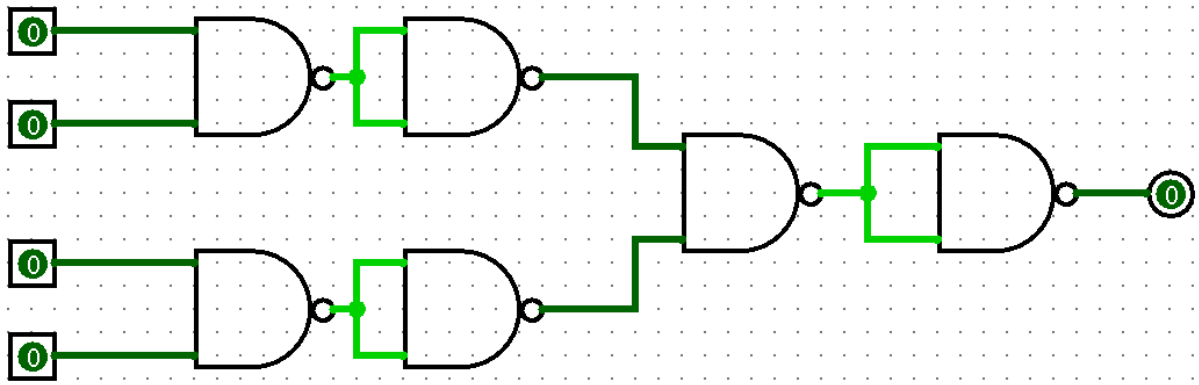
Exercise 9E

Implement a four-input XNOR gate using NAND gate. In this circuit, the inputs A and B are connected to a NAND gate, and the inputs C and D are connected to another NAND gate. The outputs of these two NAND gates are then connected to another NAND gate. The output of this NAND gate is then the output of the four-input XNOR gate, which corresponds to the truth table for the four-input XNOR gate. Therefore, this circuit implements a four-input XNOR gate using NAND gates.



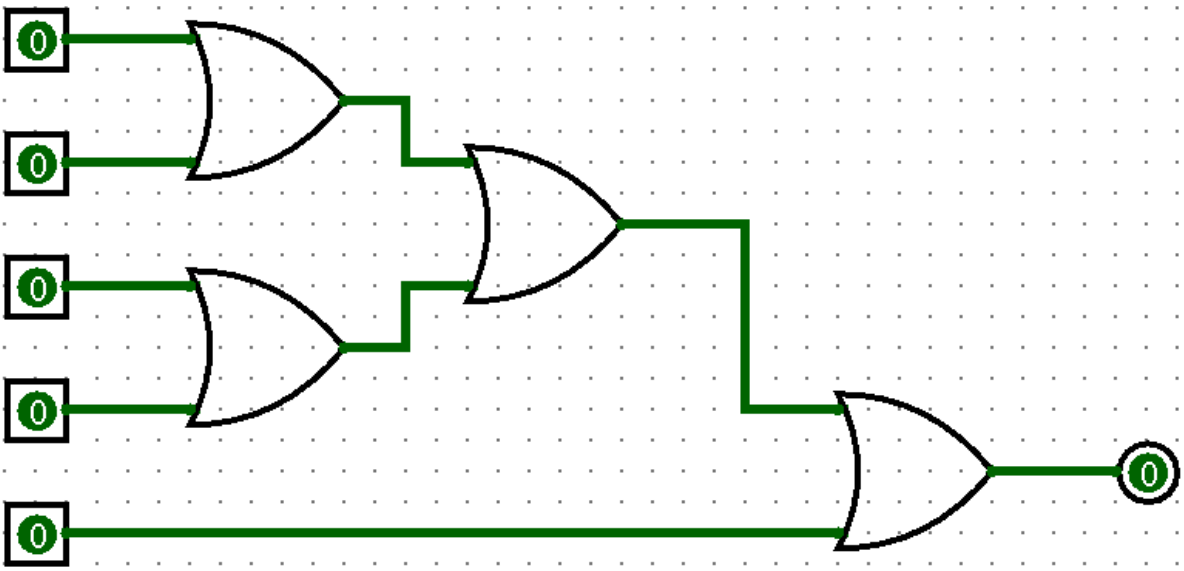
Exercise 10E

Implement a four-input AND gate using NAND gate. In this circuit, the inputs A and B are connected to a NAND gate, and the inputs C and D are connected to another NAND gate. The outputs of these two NAND gates are then connected to another NAND gate. The output of this NAND gate is then the output of the four-input AND gate, which corresponds to the truth table for the four-input AND gate. Therefore, this circuit implements a four-input AND gate using NAND gates.



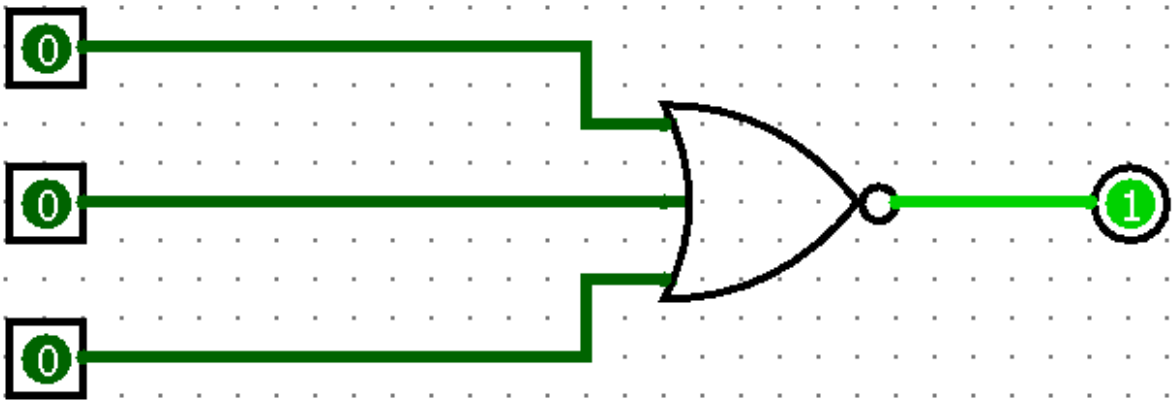
Exercise 11E

Implement a five-input OR gate using OR gate. To implement a five-input OR gate using OR gates, we can use a tree-like structure. We can start by grouping the inputs into pairs and using OR gates to combine them. Then, we can combine the resulting outputs in pairs using additional OR gates, until we have a single output.



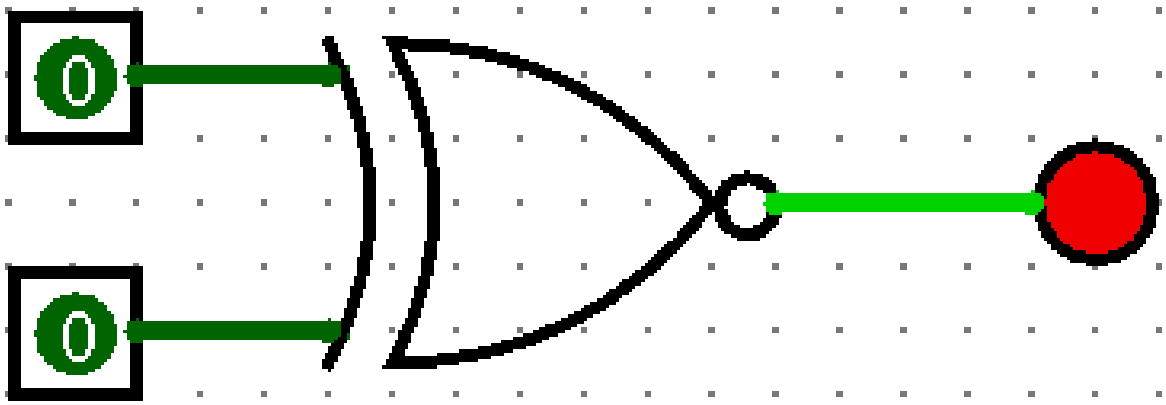
Exercise 12E

Implement a three-input NOT gate using the NOR gate. A NOT gate is a logic gate that inverts its input signal, i.e., if its input is high (1), its output will be low (0), and vice versa. We can implement a three-input NOT gate using the NOR gate by connecting all three inputs to one input of a NOR gate and its output to the other input of another NOR gate. The output of the second NOR gate will be the inverted output of the three-input NOT gate.



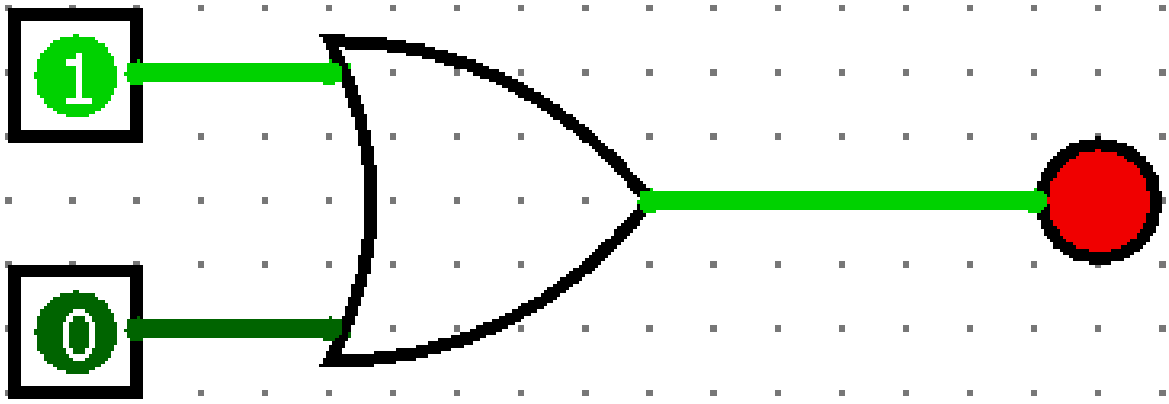
Exercise 1M

Implement a logic circuit that will activate the output if two inputs are equal. To implement a logic circuit that will activate the output if two inputs are equal, we can use an XOR gate. An XOR gate produces an output of 1 only when its two inputs are different. Thus, we can connect the two inputs to the two inputs of an XOR gate, and the output of the XOR gate will be 1 only when the inputs are different, i.e., when they are not equal. To obtain the desired behavior, we can connect the output of the XOR gate to a NOT gate, which will invert the output, producing a 1 only when the inputs are equal.



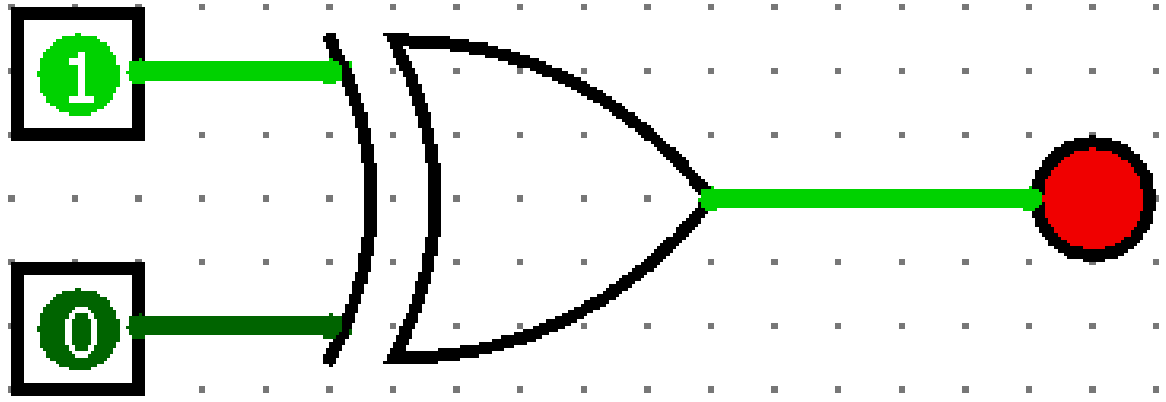
Exercise 2M

Implement a logic circuit that will activate the output if at least one of two inputs is 1. To implement a logic circuit that will activate the output if at least one of two inputs is 1, we can use an OR gate. An OR gate produces an output of 1 when at least one of its inputs is 1. Thus, we can connect the two inputs to the two inputs of an OR gate, and the output of the OR gate will be 1 if at least one of the inputs is 1.



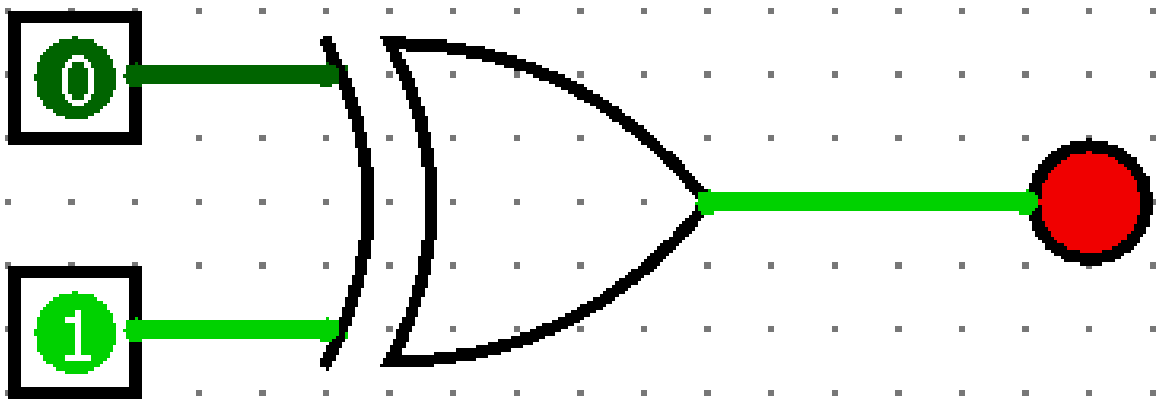
Exercise 3M

Implement a logic circuit that will activate the output if two inputs are different. To implement a logic circuit that will activate the output if two inputs are different, we can use an XOR gate. An XOR gate produces an output of 1 only when its two inputs are different. Thus, we can connect the two inputs to the two inputs of an XOR gate, and the output of the XOR gate will be 1 only when the inputs are different.



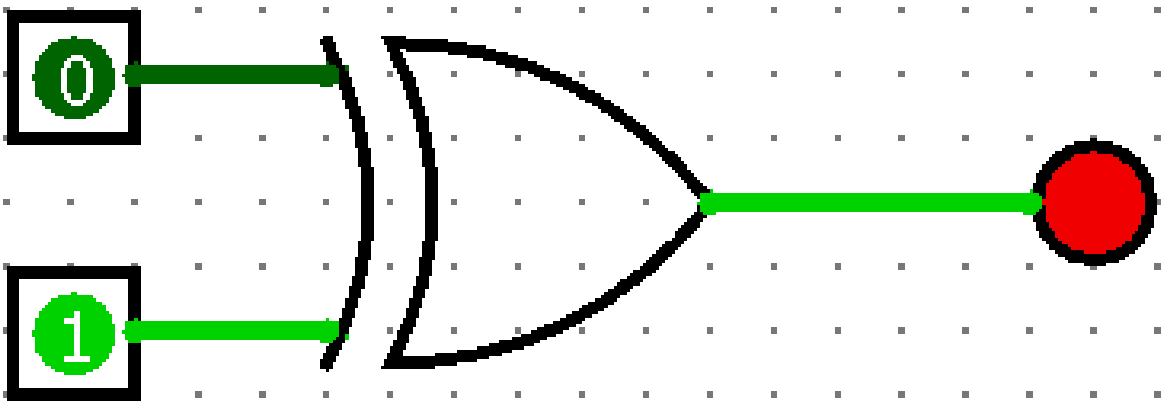
Exercise 4M

Implement a logic circuit that will activate the output if one of two inputs is 1 and the other is 0. To implement a logic circuit that will activate the output if one of two inputs is 1 and the other is 0, we can use an XNOR gate followed by a NOT gate. An XNOR gate produces an output of 1 when its two inputs are equal, and a NOT gate inverts the output. Thus, we can connect the two inputs to the two inputs of an XNOR gate, and the output of the XNOR gate will be 1 only when the inputs are equal, i.e., when they are both 0 or both 1. By applying a NOT gate to the output of the XNOR gate, we will get a 1 output only when the inputs are different, i.e., one is 0 and the other is 1.



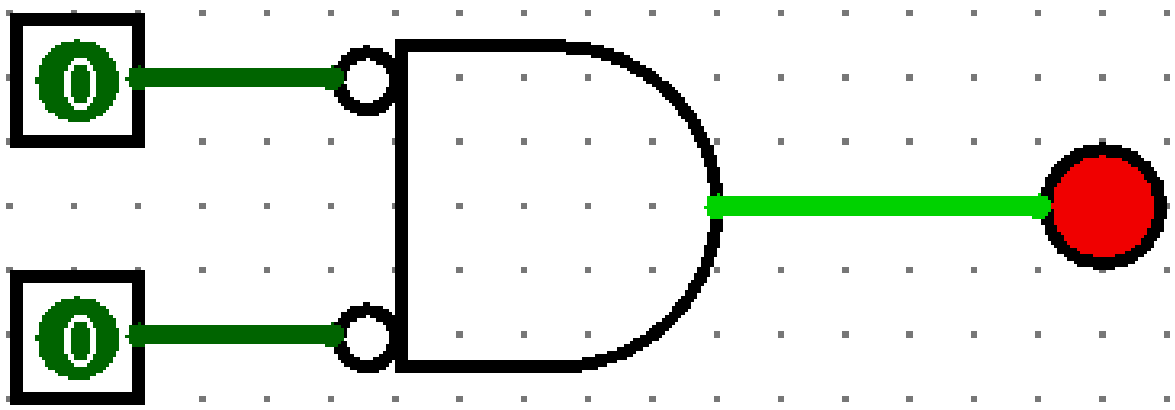
Exercise 5M

Implement a logic circuit that will activate the output if one of two inputs is 0 and the other is 1. To implement a logic circuit that will activate the output if one of two inputs is 0 and the other is 1, we can use an AND gate followed by a NOT gate. An AND gate produces an output of 1 only when both its inputs are 1, and a NOT gate inverts the output. Thus, we can connect the two inputs to the two inputs of an AND gate, and the output of the AND gate will be 1 only when both inputs are 1. By applying a NOT gate to the output of the AND gate, we will get a 1 output only when both inputs are 0, i.e., when one input is 0 and the other is 1.



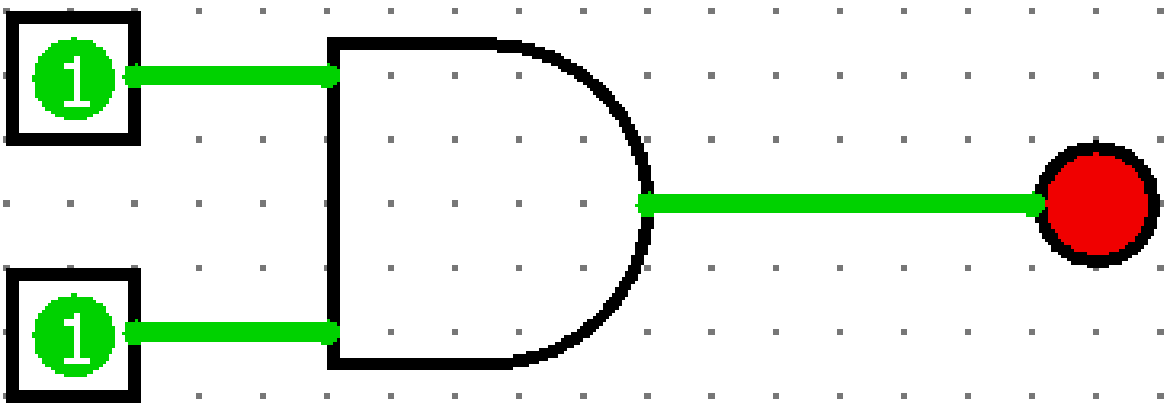
Exercise 6M

Implement a logic circuit that will enable the output if both inputs are 0. To implement a logic circuit that will enable the output if both inputs are 0, we can use an AND gate with inverted inputs. By inverting both inputs of an AND gate, the output of the AND gate will be 1 only when both inputs are 0.



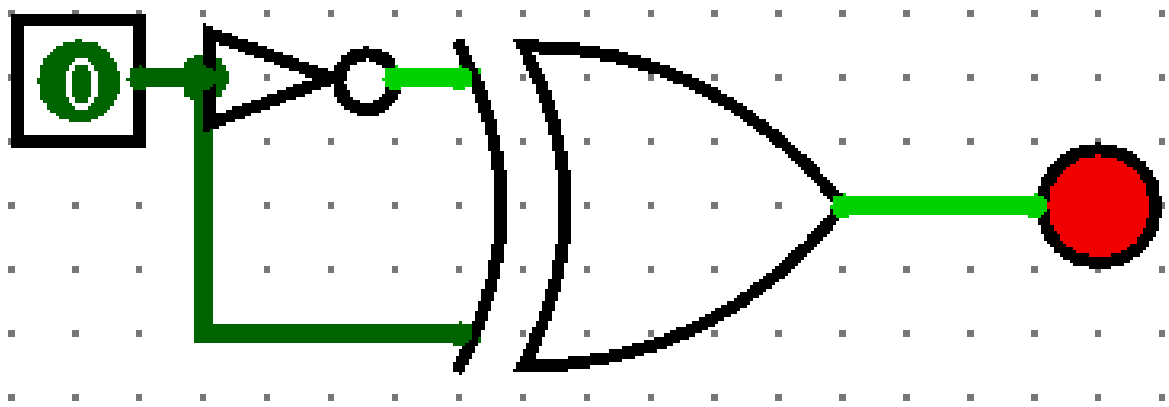
Exercise 7M

Implement a logic circuit that will activate the output if both inputs are 1. To implement a logic circuit that will activate the output if both inputs are 1, we can use an AND gate. An AND gate produces an output of 1 only when both its inputs are 1.



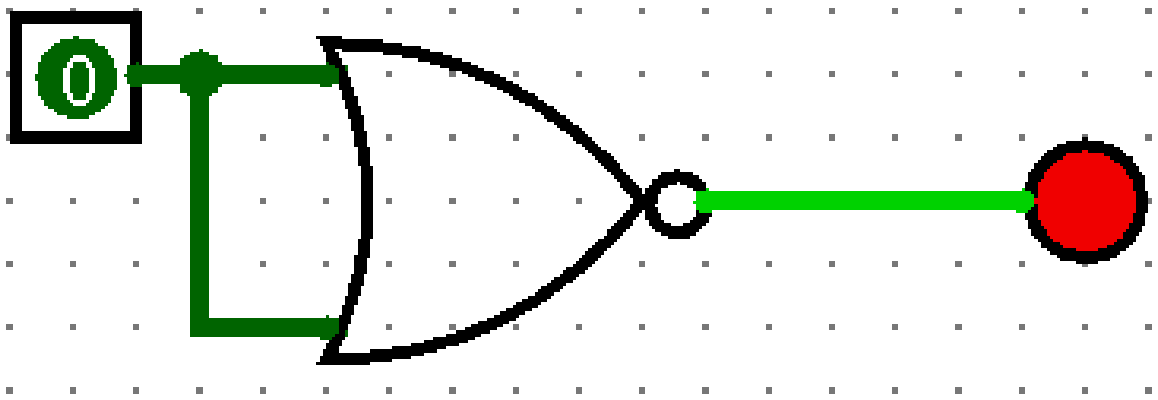
Exercise 8M

Implement a logic circuit that will enable the output if the input is negated. o implement a logic circuit that will enable the output if the input is negated, we can simply use a NOT gate. A NOT gate inverts its input, producing an output of 1 when the input is 0, and an output of 0 when the input is 1.



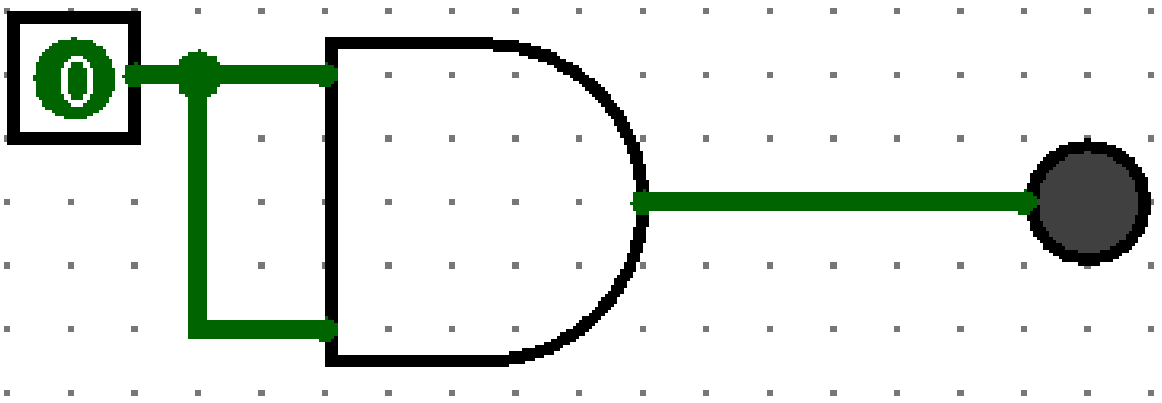
Exercise 9M

Implement a logic circuit that will activate the output if the input is null. To implement a logic circuit that will activate the output if the input is null (i.e., equal to 0), we can use an inverter to negate the input, and then use an AND gate with one input connected to the inverted input and the other input connected to a constant value of 1. This will cause the output of the AND gate to be 1 only when the input is 0.



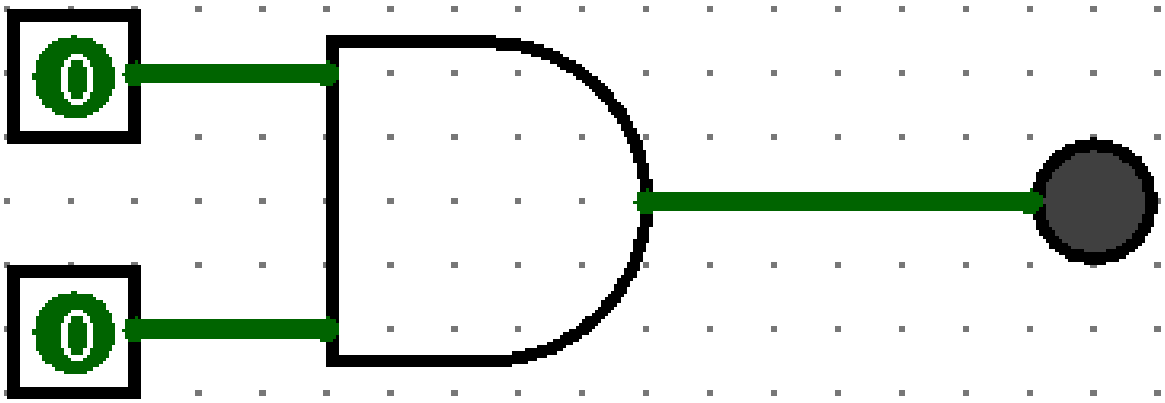
Exercise 10M

Implement a logic circuit that will activate the output if the input is non-zero. To implement a logic circuit that will activate the output if the input is non-zero, we can use a series of logic gates. First, we can use a NOT gate to invert the input. Then, we can use an AND gate with one input connected to the inverted input and the other input connected to the original input. This will cause the output of the AND gate to be 0 only when the input is 0. Finally, we can use another NOT gate to invert the output of the AND gate, producing an output of 1 when the input is non-zero.



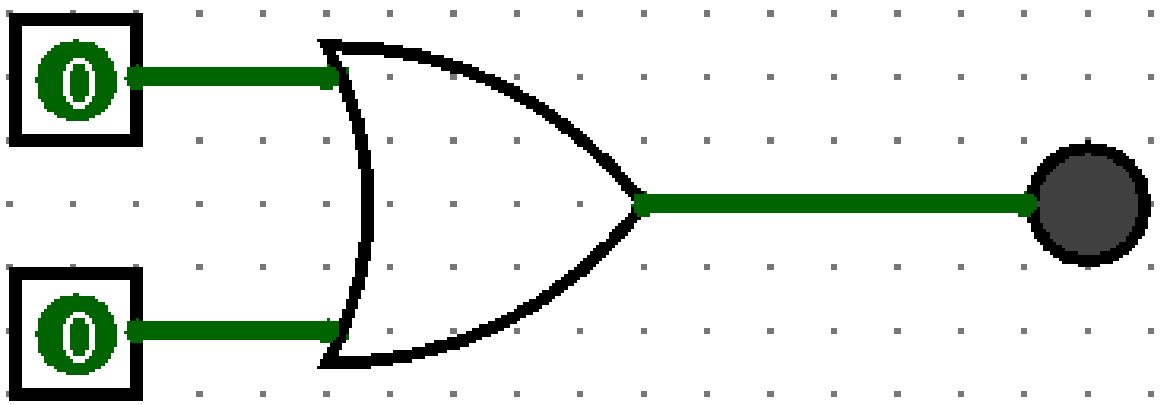
Exercise 11M

Implement a logic circuit that will activate the output if both inputs are 1, but not if both are 0. To implement a logic circuit that will activate the output if both inputs are 1, but not if both are 0, we can use an XOR gate and an AND gate. We can connect the two inputs to the XOR gate, and then connect the output of the XOR gate to one input of the AND gate. We can then connect both inputs to the other input of the AND gate. This will cause the output of the AND gate to be 1 only when both inputs are 1 and not when both inputs are 0.



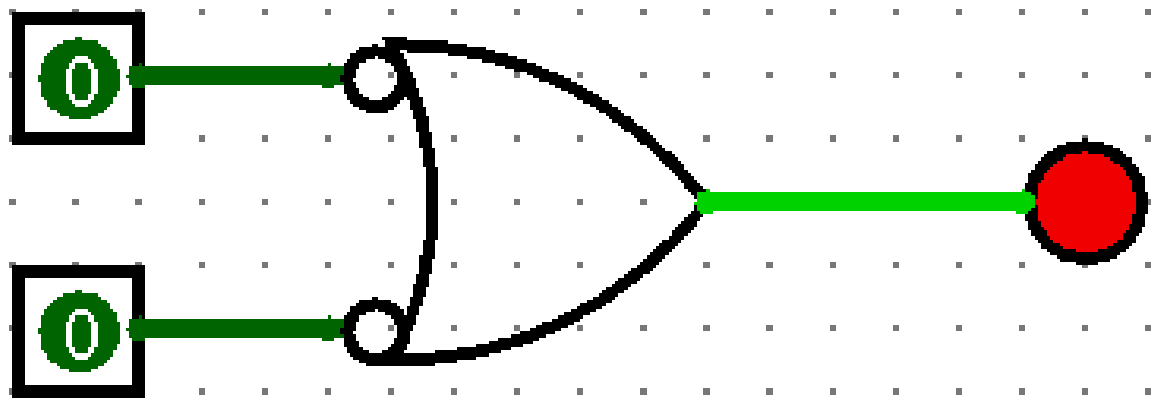
Exercise 12M

Implement a logic circuit that will activate the output if at least one of two inputs is 1, but not if both are 0. To implement a logic circuit that will activate the output if at least one of two inputs is 1, but not if both are 0, we can use an XOR gate and an AND gate. We can connect the two inputs to the XOR gate, and then connect the output of the XOR gate to one input of the AND gate. We can then connect both inputs to the other input of the AND gate, but also invert both inputs before connecting them. This will cause the output of the AND gate to be 0 only when both inputs are 0, but to be 1 in all other cases.



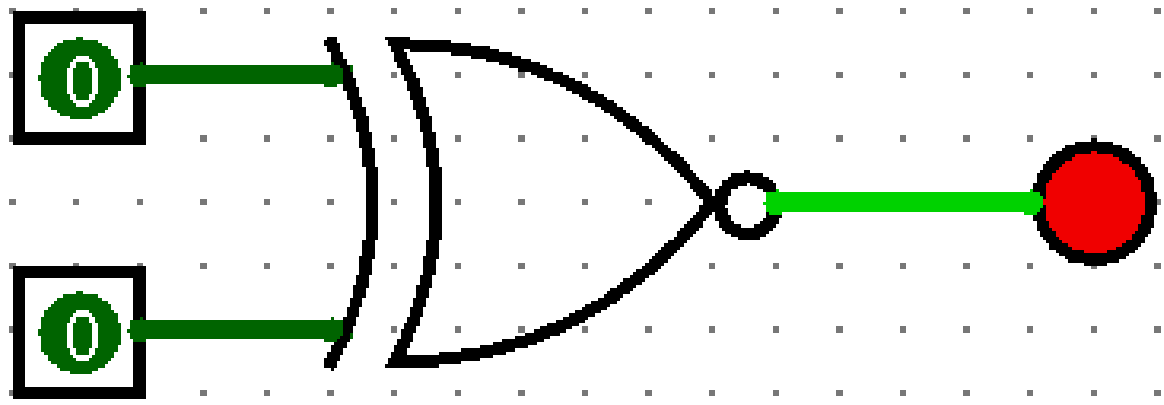
Exercise 13M

Implement a logic circuit that will activate the output if at least one of two inputs is 0, but not if both are 1. To implement a logic circuit that will activate the output if at least one of two inputs is 0, but not if both are 1, we can use an XOR gate and a NAND gate. We can connect the two inputs to the XOR gate, and then connect the output of the XOR gate to one input of the NAND gate. We can then connect both inputs to the other input of the NAND gate. This will cause the output of the NAND gate to be 1 only when at least one input is 0, but not when both inputs are 1.



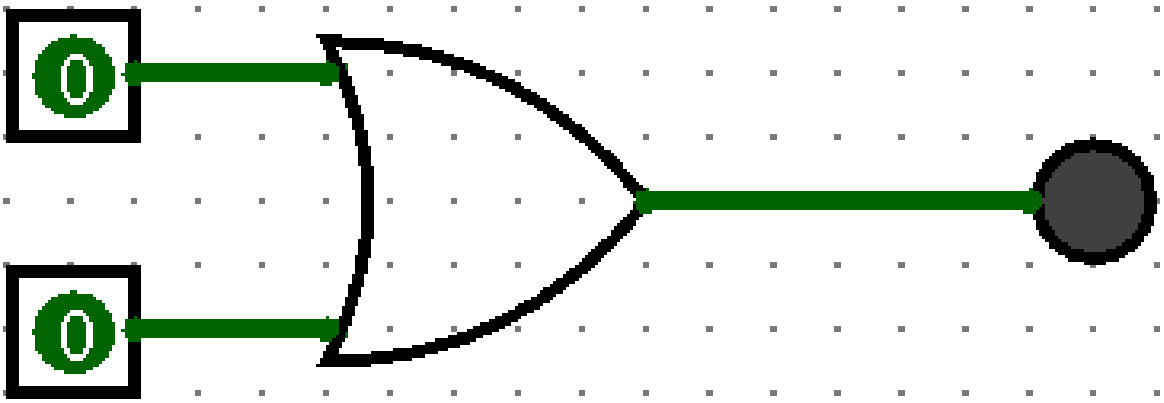
Exercise 14M

Implement a logic circuit that will activate the output if both inputs are 0 or both are 1, but not if one of them is 0 and the other is 1. To implement a logic circuit that will activate the output if both inputs are 0 or both are 1, but not if one of them is 0 and the other is 1, we can use an XOR gate and a NOT gate. We can connect the two inputs to the XOR gate, and then connect the output of the XOR gate to the input of the NOT gate. This will cause the output of the NOT gate to be 1 only when both inputs are the same (either both 0 or both 1), but not when one of the inputs is 0 and the other is 1.



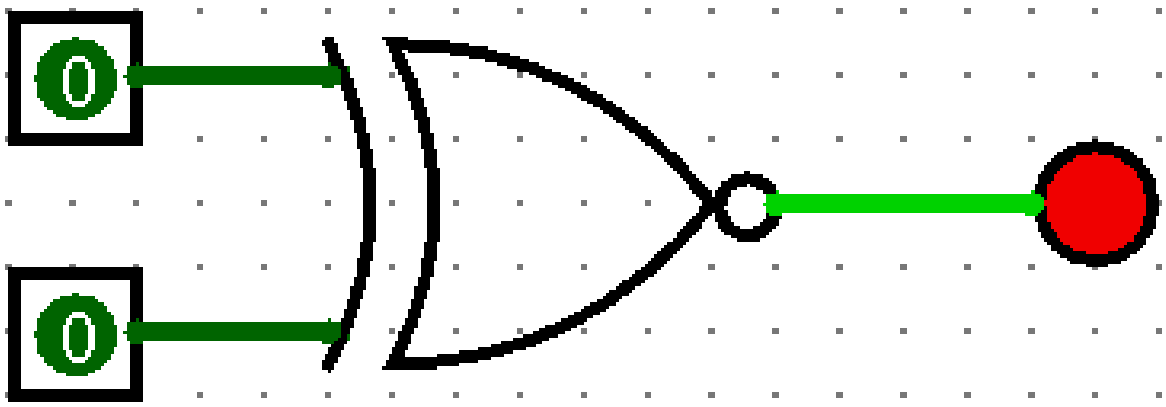
Exercise 15M

Implement a logic circuit that will activate the output if both inputs are 1 and at least one of them is 0. To implement a logic circuit that will activate the output if both inputs are 1 and at least one of them is 0, we can use an AND gate and a NOR gate. We can connect the two inputs to both the AND gate and the NOR gate, and then connect the outputs of both gates to the inputs of another AND gate. This will cause the output of the final AND gate to be 1 only when both inputs are 1 and at least one of them is 0.



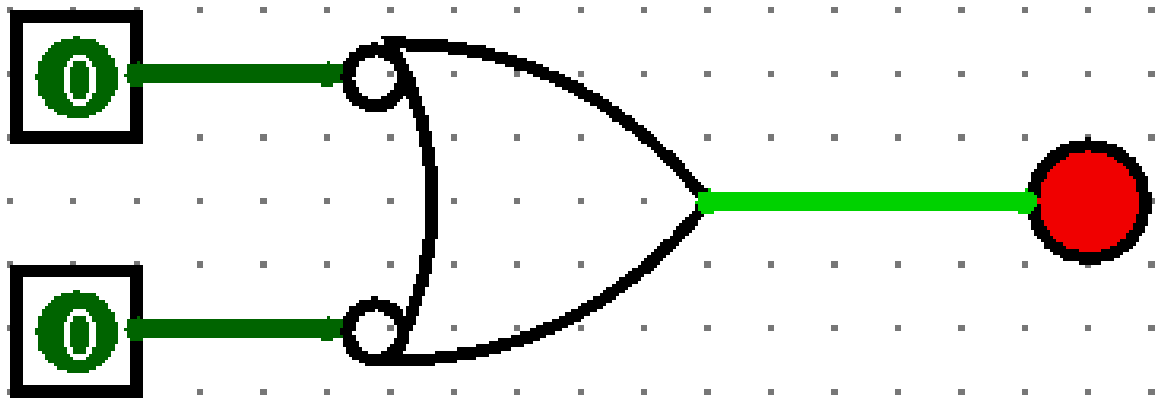
Exercise 16M

Implement a logic circuit that will activate the output if both inputs are 0 or both are 1, but not if one of them is 1 and the other is 0. To implement a logic circuit that will activate the output if both inputs are 0 or both are 1, but not if one of them is 1 and the other is 0, we can use an XOR gate and an inverter. We can connect both inputs to the XOR gate and then connect the output of the XOR gate to the input of the inverter. The output of the inverter will be the final output of the circuit.



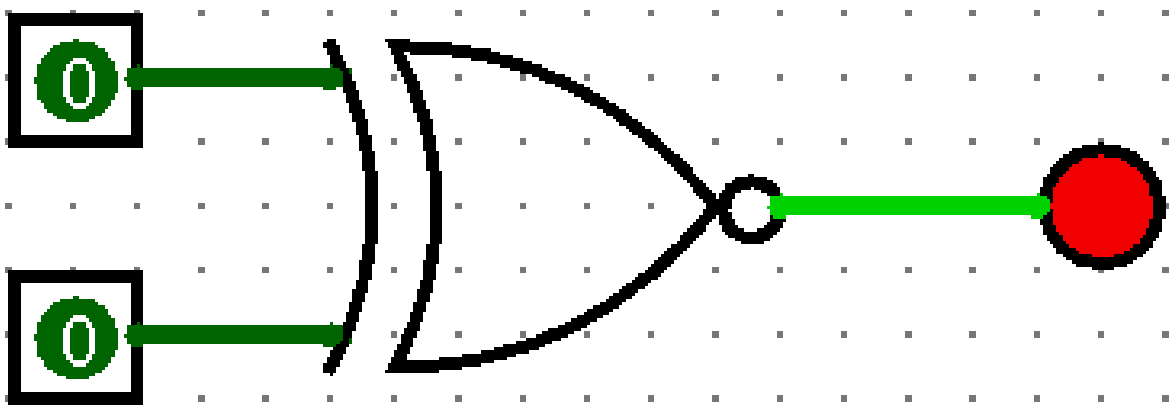
Exercise 17M

Implement a logic circuit that will activate the output if both inputs are 0 and at least one of them is 1. To implement a logic circuit that will activate the output if both inputs are 0 or at least one of them is 1, we can use an OR gate. We can connect both inputs to the OR gate and the output of the OR gate will be the final output of the circuit.



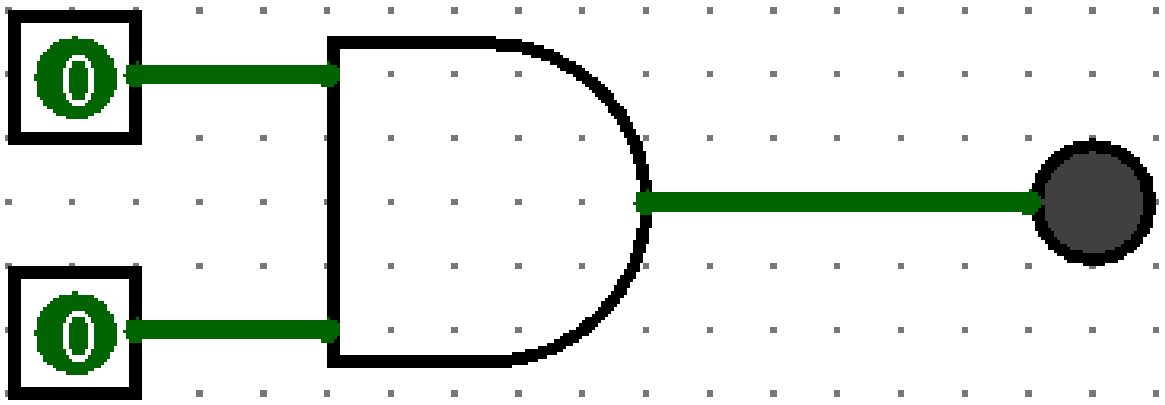
Exercise 18M

Implement a logic circuit that will activate the output if both inputs are 1 or both are 0, but not if one of them is 0 and the other is 1. To implement a logic circuit that will activate the output if both inputs are 1 or both are 0, but not if one of them is 0 and the other is 1, we can use an XOR gate followed by an AND gate. We can connect both inputs to the XOR gate and the output of the XOR gate will be connected to one input of the AND gate. The other input of the AND gate will be connected to the complement (i.e., NOT) of one of the inputs.



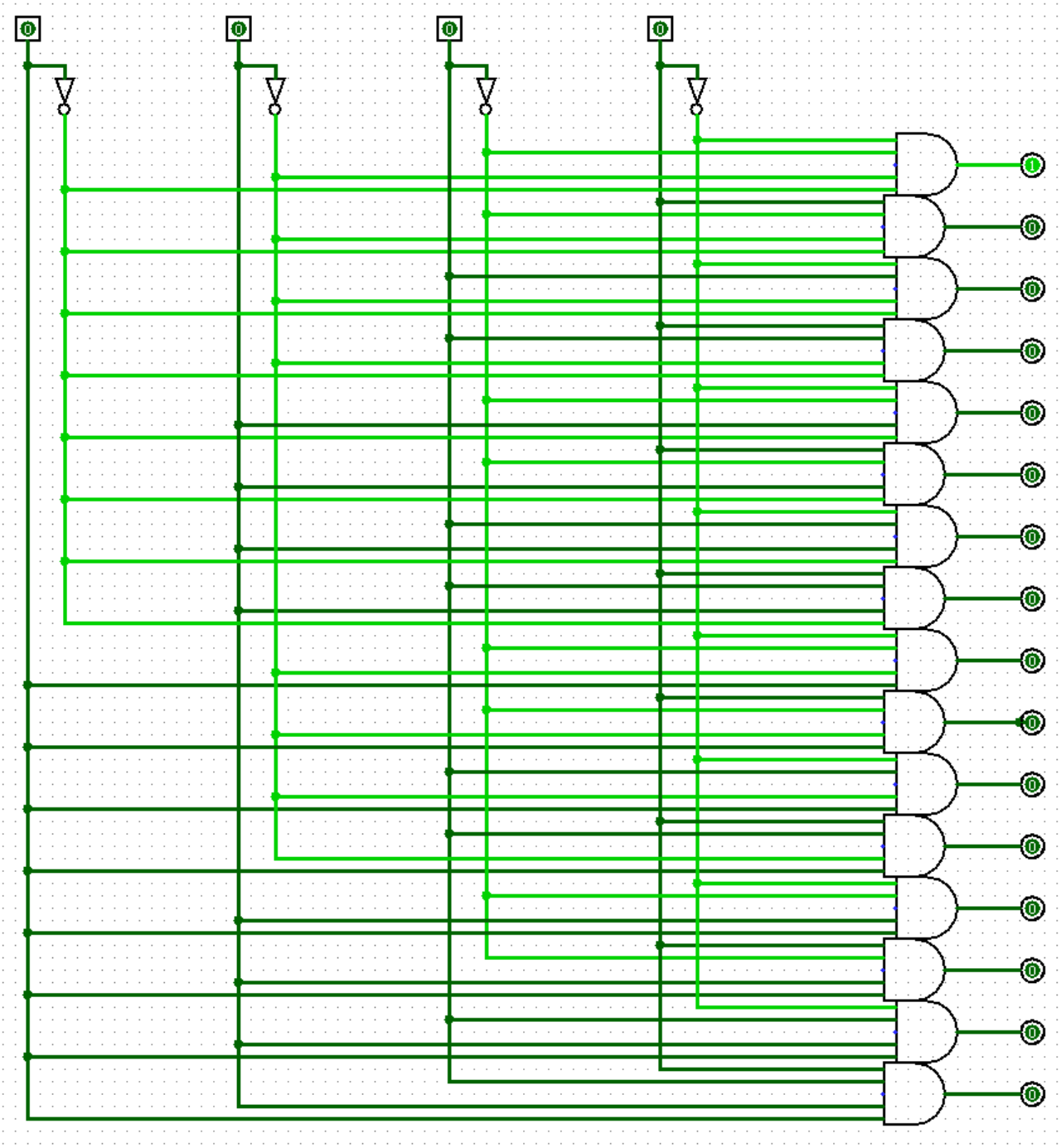
Exercise 19M

Implement a logic circuit that will activate the output if both inputs are 1 and there are no 0s between them. In this circuit, the two inputs are connected to two separate inputs of an AND gate. The AND gate will only output a 1 if both inputs are 1. The two inputs are also connected to a series of NOT gates that invert the signals. The outputs of the NOT gates are connected to the inputs of another AND gate, which will only output a 1 if both inputs are 0. Finally, the output of the two AND gates are connected to the inputs of an OR gate. The OR gate will output a 1 if either the first AND gate or the second AND gate output a 1.



Exercise 2H

A 2 to 4 decoder can be used to implement a 4 to 16 decoder by using two of them. The inputs of the 4 to 16 decoder are divided into two groups of two, and each group is connected to one of the 2 to 4 decoders. The inputs D0 and D1 are connected to the first 2 to 4 decoder, and inputs D2 and D3 are connected to the second 2 to 4 decoder. The outputs Y0 to Y3 from both decoders are combined to generate the 16 output lines Y0 to Y15 of the 4 to 16 decoder.



Conclusions

Logisim's intuitive interface and comprehensive component library make it easy for users to design and simulate digital circuits. The software's drag-and-drop functionality and customizable toolbars allow designers to quickly select and place components, while its versatile wiring tool makes it easy to connect components and define signal paths. Logisim's library includes various components such as logic gates, adders, multiplexers, registers, and memory, enabling designers to create complex digital circuits with ease.

Another strength of Logisim is its advanced simulation capabilities. The software provides a range of simulation tools that enable designers to verify circuit behavior, detect errors, and optimize circuit performance. For example, the software allows designers to simulate clock cycles, set input signals, and monitor output signals to evaluate circuit performance. Logisim also provides debugging tools that allow designers to isolate and fix errors in the circuit.

Furthermore, Logisim supports collaboration and teamwork by allowing designers to share circuits with colleagues and peers. The software enables users to export and import circuits in various formats, including VHDL, Verilog, and circuits diagrams. This feature is particularly useful for teams working on a shared project, as it allows them to work on different aspects of the circuit simultaneously and merge their changes seamlessly.

In conclusion, Logisim is a powerful and versatile digital circuit design and simulation tool that empowers users to create, edit, and simulate complex digital circuits. The software's intuitive interface, comprehensive component library, advanced simulation capabilities, and collaboration features make it an indispensable tool for designers in various fields. Whether you're a student learning digital circuit design or a professional working on complex digital systems, Logisim provides the tools and resources you need to succeed.