Logic Gates Design Studio

Gordon Stein and Caitlin Snyder

Abstract

Logic gates are the foundation of all digital circuits. From these building blocks, a huge variety of possible circuits can be created. A model-based design studio allows for students to learn how to assemble these components in a comfortable environment. A meta-model was created encompassing the simple digital circuits domain and a design studio using the meta-model were created based on WebGME. The studio created integrates simulation in the browser and allows for export as SystemVerilog code to create more opportunities for learning through the system.

Introduction

In general, digital circuits performs their computations using components known as logic gates. These gates, such as the AND, OR, NOT, XOR, NAND, and NOR gates, represent some operation of Boolean algebra [3]. By combining these gates in various configurations, it is possible to create many practical applications, from simple adders and flip-flop latches to entire computers.

Model-integrated computing (MIC) “extends the scope and use of models” by developing models “to provide solutions for an entire class of problems” [5]. Models are combined with interpreters to automatically generate a usable output. The MIC process involves systems engineers creating a system of meta-models and domain-specific reusable components, which are given to domain engineers to build their models of their system under design. The models created by the domain engineers are then transformed into some other format to be used beyond the interface of the design studio. The tools available for MIC allow for domain-specific design studios to be created, customizing the design process for each domain involved.

A model-integrated approach follows logically from how digital circuits are designed and assembled by engineers. Circuits are often designed as smaller modules first, such as a “full adder” being created using multiple “half adder” modules, which are in turn composed of a collection of logic gates. The logic gates all have well-defined functions relating their inputs and outputs, allowing them to be easily modeled, and therefore the entire system can be modeled easily as well. In addition, the components of a digital circuit all follow similar rules, belonging to a single class of object, making a meta-model for the domain straightforward. The component model registry for the domain is then simply a collection of standard logic gates and other elementary circuit components.

The intended end users for this design studio are students learning the fundamentals of digital circuit design. The simple web-based interface provided enables rapid creation and simulation of circuits and the branching version control allows for convenient exploration of new ideas. While it cannot be expected to compete with existing commercial circuit design software, students could use this design studio to easily collaborate on class projects and both receive and submit assignments with their instructors.

Existing Design Studios

A design studio for the logic gates domain in WebGME has been created previously [6]. This existing design studio lacks a simulation option, but does allow for modelling of digital circuits. No code from this existing project was used in the creation of the new design studio in order to avoid any accidental plagiarism.

Another similar preexisting design studio is CircuitLab [8]. CircuitLab is a general circuit modelling tool, which includes logic gate components in addition to components such as resistors and capacitors. CircuitLab has many options available for non-interactive simulation. However, CircuitLab represents a different, but overlapping, domain from simple digital circuits. While the goal of the new design studio is to represent purely the domain of logic gates and simple digital circuits composed of them, CircuitLab is designed around comprehensive modelling of DC circuits, requiring significant additional knowledge to be used for the same purpose.

Methods

WebGME

WebGME [1][9] is a web-based toolsuite for MIC designed to replace the older Generic Modeling Environment toolsuite. It allows for collaborative online modelling and meta-modelling of a large range of possible domains. WebGME is open-source and designed to be modified and extended in order to create new design studios for desired domains.

Creating a new design studio required the creation of a “decorator” to display components properly, “interpreters” to transform the model into code, and a “visualizer” where the model can be shown to the end user after some transformation. In addition to these, in WebGME itself a meta-model and a component library needed to be created. The meta-model represents the different potential classes of objects, their hierarchy, and their constraints. The component library has concrete examples of different model components readily available to be added into a model under design. WebGME also includes cross-cut and set membership interfaces, which are not used in this design studio.

Decorator

In order to display the logic gates in the model of a circuit properly, a WebGME “decorator” had to be created. Without the decorator, each logic gate would be presented as a rectangle with the name of the gate superimposed on it. This view would not be easy to understand at a glance and would not match the diagrams engineers are familiar with. While there is a defined set of symbols using rectangular symbols for gates, which would reduce time required for development, it was decided upon to use the more commonly seen “distinctive-shape” symbols [7]. All gate symbols are included as SVG files with no wires. The wires for the gates are procedurally generated based on the input and output ports in the model itself. This allows the decorator to adapt to the possibility that a user desires to have additional inputs on a logic gate, for example to create a 3-input or 4-input AND gate, which are available commercially as integrated circuits. This capability is demonstrated in Figure 4 in the Discussion section of the paper.

Interpreter

Models created in the design studio have two interpreters available to generate usable code. The first transforms the model into a representation in the SystemVerilog language. SystemVerilog is “a unified hardware design, specification, and verification language” [2]. This creates a standardized output for use with other tools, and also enables testing systems to be created that interact with the model created.

Another code generator is available to transform the model into a JSON structure used with the visualizer. While this output is not necessarily very useful for students, it is available in the event that it is desired. A student who completes a circuit may wish to export it in the DigitalJS-compatible output in order to embed their creation on a webpage to share it more readily. Unfortunately, this format, unlike SystemVerilog, does not allow easy combinations of gates, so the interpreter must divide gates that were provided with additional inputs into a combination of the two-input versions of the gates.

Visualizer

Visualization and simulation were provided by integrating the DigitalJS library [4] into the design studio. This JavaScript library displays a graphical representation of a given circuit, and allows for simulation of the circuit directly in the browser. It has the capability of modelling all standard logic gates, Boolean inputs and outputs, numeric inputs and outputs, and memory components.

Due to conflicts between the client-side libraries used in DigitalJS and those used in WebGME, additional steps were required to integrate the simulation into the design studio interface. A custom router was created to serve an HTML page and the libraries required. Upon requesting the “CircuitSimulator” visualizer, the interpreter runs and its output is inserted into the HTML

page, which is then loaded into an iframe. The requested circuit will then be interactable in the browser.

Meta-Model

The system's metamodel contains components representing the necessary elements of a digital circuit including ports, connections, logic circuits and the different logic circuit components. The components of a logic circuit are the gates, inputs, and outputs. Each possible gate (AND, OR, NOT, XOR, NAND, and NOR) is represented as a component in the metamodel. The attributes in the logic gate components contain values necessary for the interpreter. A portion of the metamodel can be seen in Figure 1 below.

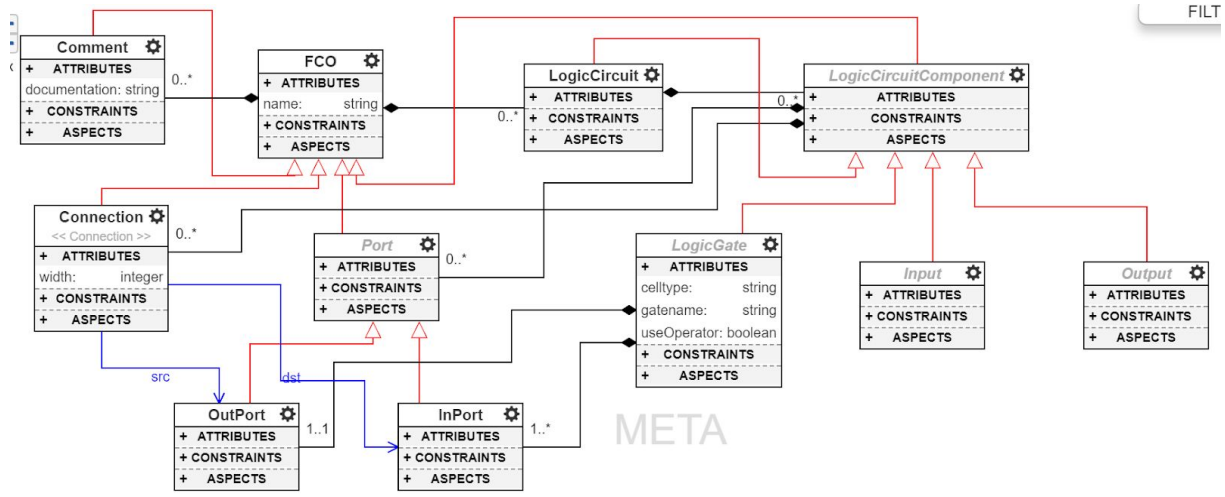

Figure 1: Metamodel

Discussion

The goal of this project was to create a new design studio for modelling digital circuits. We chose to evaluate this by comparing it to the existing design studio. The existing studio lacks a simulation component, which has been implemented in the new studio. However, the existing studio's decorator provides a slightly nicer output. Due to the open-source nature of both projects, it will be possible to combine these to obtain a "best of both worlds".

To both examine and demonstrate the use of this design studio, three example circuits were created: "Half Adder", "SR Latch" and "Big AND". The Half Adder example, as seen in Figure 2, shows the flexibility of the input components in the system by allowing any number of logic gate components to receive the same input.

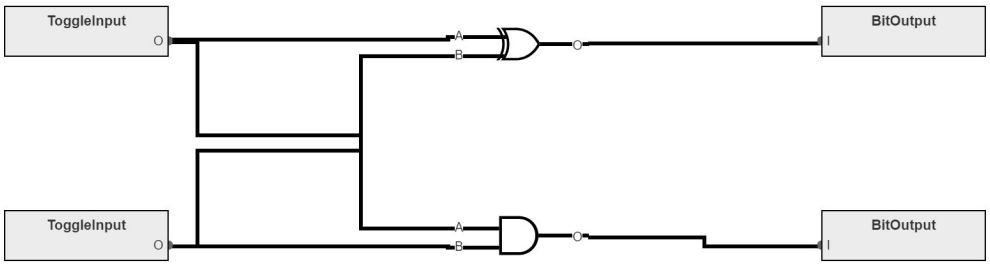

Figure 2: Half Adder

The SR Latch example, seen in Figure 3, illustrates the ability for users of the system to create circuits where the connections between gates includes a feedback loop.

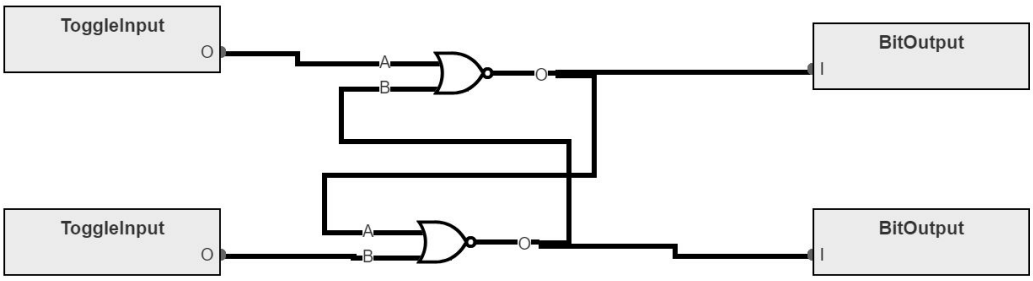

Figure 3: SR Latch

The Big AND example, shown in Figure 4, illustrates the versatility of the number of inputs the system allows for a given logic gate component along with the adaptation ability of the decorator to depict such a gate. Overall, the three examples depict the versatility of the system for designing a multitude of different logical circuits.

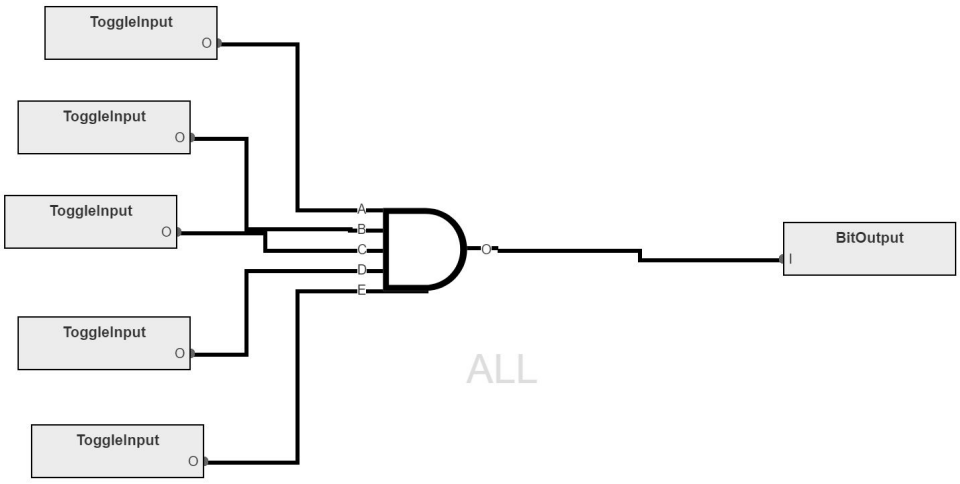

Figure 4: Big AND

Conclusion

Model-based design makes sense for digital circuit design. This domain is easily adapted into a meta-model and component library. By using model-integrated computing techniques and WebGME, it was possible to create a new design studio for potential educational use, where students would have a convenient interface for designing and simulating digital circuits.

Future Work

The current simulation package used has some limitations. It requires a specific code generator in order to work with it directly, and it is too incompatible with the existing WebGME packages to be easily integrated further. A future version of the design studio could use a custom-made digital circuit simulator in order to better incorporate the simulator into the design process. With proper integration into WebGME, design could be performed directly in the simulator instead of the model composition view, while the meta-model would continue to be used to describe the model's components.

Another expansion on the simulation would be to create an interface for better sharing of completed projects. Students who complete a project and would like to share it online currently are required to either export the code and create their own webpage to showcase the design, or to share their project in the design studio itself. A more ideal interface would allow simulations of submitted circuits to be viewed directly, with a search capability and the ability to view the original project in the design studio to inspect or modify it.

The SystemVerilog output has not been used to the fullest extent in this design studio. The existing toolset for this language could be leveraged to provide additional features to student users. For example, the code could be used to create hardware versions of their circuits, possibly using a field programmable gate array (FPGA). This would allow students to see their designs as more than simply images on a computer screen, while also demonstrating that the design studio can have practical use outside of an educational environment.

References

[1] Maróti, Miklós & Kereskényi, Róbert & Kecskes, Tamas & Völgyesi, Péter & Ledeczi, Akos. (2014). Online Collaborative Environment for Designing Complex Computational Systems. Procedia Computer Science. 29. 2432-2441. 10.1016/j.procs.2014.05.227.
[2] IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language," in IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012) , vol., no., pp.1-1315, 22 Feb. 2018. 10.1109/IEEESTD.2018.8299595
[3] Horowitz, Paul, and Winfield Hill. "The Art of Electronics." The Art of Electronics, by Paul Horowitz, Winfield Hill, Cambridge, UK: Cambridge University Press, 2015 (2015).

[4] Materzok, Marek. DigitalJS [Computer software]. (2018) Retrieved from https://github.com/tilk/digitaljs
[5] Sztipanovits, Janos, and Gabor Karsai. "Model-integrated computing." Computer 30.4 (1997): 110-111.
[6] Meijer, Patrik. Logic Gates [Computer software]. (2017) Retrieved from https://github.com/webgme/logic-gates
[7] IEEE Standard Graphic Symbols for Logic Functions (Including and incorporating IEEE Std 91a-1991, Supplement to IEEE Standard Graphic Symbols for Logic Functions)," in IEEE Std 91a-1991 & IEEE Std 91-1984 , vol., no., pp.1-160, 1984. 10.1109/IEEESTD.1984.7896954
[8] CircuitLab Team. CircuitLab [Computer software]. (2018) Retrieved from https://www.circuitlab.com
[9] M. Maróti, T. Kecskes, R. Kereskényi, B. Broll, P. Völgyesi, L. Juracz, T. Levendoszky, and A. Ledeczi, “Next generation (meta) modeling: Web-and cloud-based collaborative tool infrastructure,” Proceedings of MPM, p. 41, 2014.