LogiSketch: A FREE-SKETCH DIGITAL CIRCUIT DESIGN AND SIMULATION SYSTEM

Emerging Technology Research Strand

Christine Alvarado[1,2], Andy Kearney[1], Alexa Keizur[1], Calvin Loncaric[1], Miranda Parker[1], Jessi Peck[1], Kiley Sobel[1], Fiona Tay[3]
[1]*Harvey Mudd College,* [2]*University of California, San Diego,* [3]*Pivotal Labs*
alvarado@cs.ucsd.edu

1. Abstract

This paper presents LogiSketch, a system that recognizes hand-drawn digital logic diagrams and then allows students to simulate those diagrams. LogiSketch is one of few complete sketch recognition systems (and the first in its domain) that allows the student to draw freely, without drawing style constraints. LogiSketch employs novel recognition feedback and active support for error correction. Additionally, LogiSketch incorporates behind-the-scenes user-targeted learning that improves recognition that requires no additional effort from the student. A pilot study reveals that LogiSketch succeeds in engaging students, even though it is not yet a suitable replacement for menu-based tools. Study results also reveal what is most important in the interface and functionality of a sketch recognition tool for education.

2. Problem Statement and Context

Even in today's technology-enabled classrooms, drawing remains a central activity for students in design-oriented classes. Drawing allows students to think about their designs and to focus on learning the discipline rather than on learning to use a potentially complex tool.

Of course, drawing on paper does not allow students to see the behavior of their designs, so students usually transfer their designs to a computer tool after drawing them on paper. This two-stage process is cumbersome and does not allow the computer tool to provide assistance to the student as they are initially creating their designs, when they are likely the most confused. On the other hand, having students work directly with point-and-click menu-based tools removes drawing from the learning process, which could have negative consequences, as drawing has been found to be a critical part of the design process [4, 11].

Recently deployed pen-based computer systems in the fields of statics [8, 12] and discrete math [3] have shown learning improvements by integrating drawing and interaction into a single tool. Following in the vein of these tools' successes, we developed LogiSketch, a system that allows students to freely draw and then simulate digital logic circuit diagrams. LogiSketch makes three central contributions to the field of pen-based educational tools. First, it is the first pen-based simulation tool for the domain of digital circuit design. This complex domain presents many new challenges: sketches can be quite large, many of the symbols in the domain are visually similar, and drawing styles vary significantly between users. Second, LogiSketch implements delayed recognition of unconstrained sketches in a complex domain, and provides novel recognition feedback and adaptive error correction. Third, LogiSketch incorporates behind-the-scenes user-targeted learning that improves recognition as a student uses the system.

We deployed LogiSketch in a pilot study in an introductory computer science course at Harvey Mudd College during a lab on digital circuit design. Yet we categorize LogiSketch as an emerging technology rather than deployed system because of the scope of the tool and the complexity of the domain. Our pilot study upheld our central premise—that students would find value in being able to freely draw and then simulate their circuits—but it also showed that LogiSketch is not yet a suitable replacement for point-and-click menu-based simulation tools. This paper reports on this initial student feedback to help guide the further development of LogiSketch and similar sketch-based tools for complex domains.

2.1 Related Work

LogiSketch is one of many sketch-based simulation tools for education targeting a variety of domains. These previous systems are distinct from one another and from LogiSketch not only in their domain of focus, but also in the interaction techniques they employ to aid recognition. Some systems place constraints on the way users must draw, for example requiring the user to draw each shape with an individual stroke, or to pause between symbols [2, 7, 8]. These restrictions are appropriate for some domains but do not match the way students draw digital logic diagrams [1]. Other tools, such as Mechanix [11] and ChemInk [9], allow users to draw more freely, placing few, if any, restrictions on the user. LogiSketch takes a similar approach, but unlike these previous systems, we focus not only on recognition algorithms, but on recognition feedback and error correction mechanisms that are essential in an interactive tool. Finally, there are a growing number of systems to support mathematical equation recognition and exploration [3, 6]. Because of the domain, these systems tend to be fairly different from tools that support two-dimensional drawing and simulation in both their interface and their recognition algorithms.

3. Method Employed

We had three central goals in the design of LogiSketch. First, we wanted to allow students to sketch freely, just as they would on paper, and then simulate these sketches directly, without transforming them into "clean" diagrams. We found in previous work that students prefer not to be interrupted by recognition feedback while they are designing a circuit because this feedback is distracting [13]. Informally we have also observed that the experience of having a drawing literally come to life can be very engaging for a user. Second, we wanted to provide a seamless pen-based interface to support drawing, error correction, editing and simulation. Third, we wanted to allow the creation of circuits up to the complexity students would see in the first three or four weeks of a digital circuit design course. After this time, circuits typically become complex enough to require alternate tools such as scripting languages.

Figure 1 shows the interaction process with the LogiSketch interface, which runs on a Tablet PC. On the left, the student draws her circuit diagram. When the student is finished drawing, she presses the red "Recognize" button in the upper right corner of the window. LogiSketch then interprets her diagram and gives her recognition feedback, which is shown in the middle pane in Figure 1 and in more detail in Figure 2. When the student is satisfied that the system has recognized her sketch correctly, she presses the green "Simulate" button, which appears underneath the "Recognize" button once the drawing has been recognized as a legal circuit. Once the student begins simulating the circuit (Figure 1 right, and Figure 4), she can interact with the circuit in a variety of ways, as described below.

3.1 Recognition Feedback and Error Correction

LogiSketch gives feedback through multiple channels at both the symbol level and the circuit level to help students understand how their drawings have been interpreted and how they can correct recognition errors. The system provides symbol level feedback by coloring the strokes with a unique color for each symbol and by overlaying recognized shape temples on top of the user's strokes in the case of gate recognition (Figure 2, left). These "ghost gates" not only help the user understand what symbol was recognized, but also help the user see when a gate's orientation has been recognized incorrectly, as in Figure 2, left. Additionally, hovering the pen over any recognized object will display the name of the recognized shape (Figure 3, left).

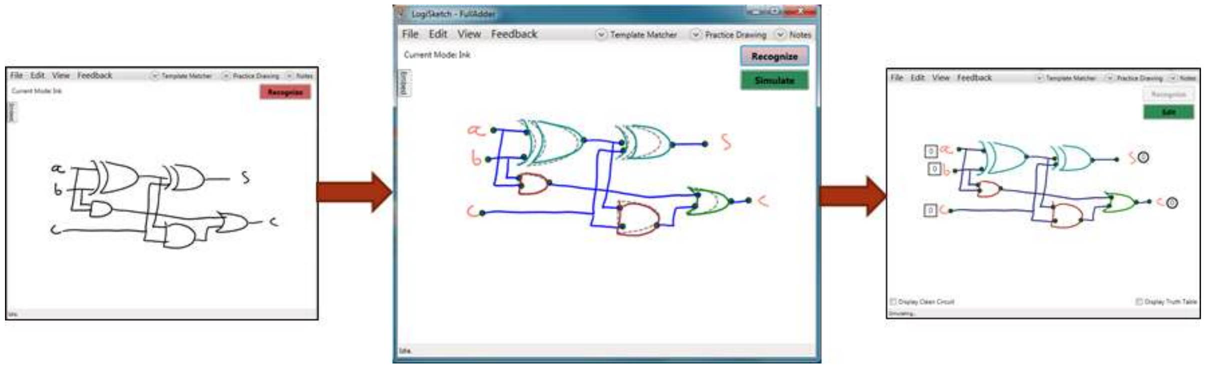

Figure 1: An overview of the LogiSketch interaction process: drawing (left), recognizing (middle) and simulating (right) a circuit.

LogiSketch also provides feedback about the recognition of the circuit at a higher level. Wire endpoints are highlighted as either small green circles, meaning that they are connected to another component in the sketch, or red x's, meaning that they are not connected to any other component in the sketch. When LogiSketch detects circuit errors that prevent it from simulating a user's sketch, it highlights these errors and displays a message about a potential way to fix the errors (Figure 2, right). Finally, LogiSketch provides wire *mesh highlighting*: when the user hovers her pen over either a wire or a gate, LogiSketch highlights the wires and gates that are directly connected to that component by thickening the strokes that comprise these objects (e.g., Figure 2 (right) where the wire and AND-gate are thickened as the user hovers over the wire).

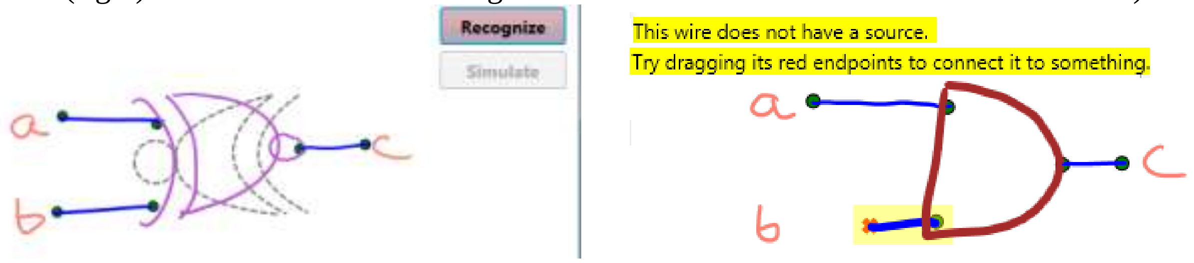
This wire does not have a source.
Try dragging its red endpoints to connect it to something.

Figure 2: Two different examples of the recognition feedback provided by LogiSketch.

LogiSketch supports efficient error correction both by providing a suite of pen-based interaction mechanisms for correcting errors as well as by leveraging the user's input to perform additional corrections automatically. As in previous systems (e.g., [12]), our modeless editing interface comprises a ring of buttons that appear when the user hovers her pen over the tablet screen for a few seconds (Figure 3). If the user is not intending to edit, she can ignore these buttons and continue to draw. If she clicks on one of the buttons, she enters a temporary editing mode where she can perform the selected action with the next stroke and then she automatically returns to drawing mode. These hover menus are context sensitive, depending on which shape the user is hovering over and which strokes (if any) are highlighted.

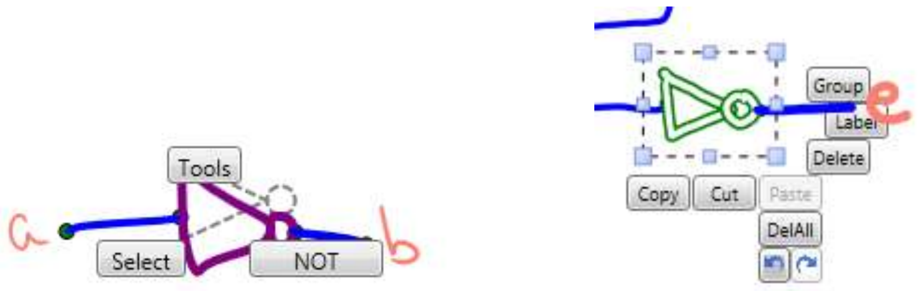

Figure 3: The edit menus that appear when the user hovers the pen over a shape while nothing is selected (left) and while a set of strokes is selected (right).

To help the user efficiently correct recognition errors, the system performs eager re-recognition in response to the user's explicit input. For example, if the system has misrecognized both a gate and the wire connected to that gate, and the user corrects the recognition of the gate, the system (usually) will automatically correct the recognition of the wire using the new context supplied by the correct interpretation of the gate. Additionally, to correct recognition errors, often it is enough for a user simply to indicate which strokes are supposed to belong to the same symbol using the "Group" menu option shown in Figure 3 (right). Once the strokes are correctly grouped, LogiSketch is usually able to determine the correct interpretation for the symbol. Simply clicking "Group" is much faster than selecting the correct interpretation from a list.

LogiSketch also allows users to correct errors by directly manipulating feedback objects in the sketch. The user can drag the red "x"s that appear at the ends of unconnected wires to a shape in the sketch to create the connection between the wire and that shape. The user can also directly rotate the ghost gates to indicate the proper orientation of a symbol.

3.2 Simulating Circuits

Once a circuit is recognized correctly, a user can interact with it in a number of ways. When she presses "Simulate" values appear next to the inputs and outputs of the circuit (Figure 4). She can toggle input values by clicking on them with the pen, and see the output values change. LogiSketch also displays data flowing through the circuit by coloring wires carrying a 1 light blue and wires carrying a 0 dark blue. The user can also open the truth table that corresponds to the circuit. Highlighting a row in the truth table shows the corresponding data in simulation.

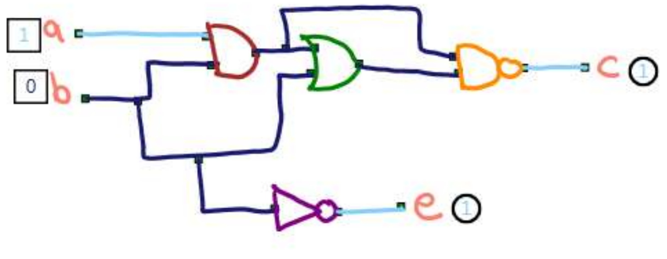

Figure 4: The simulation interface in LogiSketch

3.3 Personalized Recognition

LogiSketch uses a multi-stage recognition process that first groups strokes into individual shapes and then recognizes those shapes as individual symbols. Finally, recognized symbols are linked together into a complete circuit diagram. As part of the initial grouping process, each stroke group is categorized as either a wire, text or a gate (details can be found in [10]). Wire strokes are simply passed on to the circuit recognition algorithm. Text strokes are recognized using the Microsoft Ink Analyzer.

For gate recognition, we needed an algorithm that allows the user to understand the underlying recognition model and that is easy to adapt to a particular user's style using few examples. To meet these goals, we use the image-based nearest-neighbor classifier described in [5]. In other words, we compare each user-drawn symbol visually to a set of pre-populated examples and choose the label of the example that looks the most similar to what the user drew. We initially train (populate) the system with five relatively clean examples from each type of gate, but we add examples as the user draws, as described below.

We expose the recognition model to the user in the hope that understanding the underlying model will make users less frustrated when they encounter recognition errors and better able to subtly adapt their drawing style to reduce errors. We expose the model in two ways. First, the ghost-gates are created by drawing the gate in a way that matches the templates that the system is initially trained on. Second, we provide an interface that allows users to hover over a shape and see the template that it was matched to during the recognition process.

LogiSketch also adapts to individual users' styles without requiring users to explicitly train the recognizer. When the user re-labels a gate, the recognizer adapts by adding the drawing to its database of templates. To prune the database of templates, we keep track of how often each template in the database was matched to the user's strokes to yield both correct and incorrect recognition results. We remove the templates that are least likely to yield correct recognition and most likely to result in recognition errors.

4. Results and Evaluation

We deployed LogiSketch in an introductory computer science class at Harvey Mudd College during the lab on circuit design. During this lab students use the Logisim circuit simulation tool to design and implement circuits of increasing complexity, starting with a simple XOR circuit up to a ripple carry adder. We recruited 21 volunteers to use LogiSketch instead of Logisim to complete their lab and then to complete a short survey about their experience.

Our results showed promise for our tool. 73% of participants stated that they enjoyed using LogiSketch either "somewhat" or "quite a bit." Only 5.6% said they did not enjoy using it at all, despite LogiSketch's limitations in the context of this lab, described below. When asked what one thing they enjoyed most about LogiSketch, almost half of the students cited the intuitiveness or ease of sketching, and about a fourth of the students reported that using the interface was fun.

On the other hand, our study also revealed that LogiSketch is not yet ready to replace menu-based tools. While most students (73%) successfully completed the XOR circuit, only 33% were able to complete the ripple-carry adder.

Our study revealed two central limitations to LogiSketch that must be addressed. First, the (perceived) recognition rate is simply too low to be practical. Only 29% of users reported that LogiSketch correctly interpreted their sketch most of the time (90-100%). 56% of users felt LogiSketch understood their sketch only about half the time or less. While most users (84%) felt it was relatively easy to understand the recognition feedback LogiSketch provided, over half (55%) of users felt it was at least somewhat difficult to correct recognition errors. Many users (56%) felt that LogiSketch's recognition improved its recognition as they used it, but given the low starting point, this improvement was not enough to be useful. The second main problem users had with LogiSketch is that their drawings would become too messy as their circuits got larger. While it's nice to allow users to sketch freely and to simulate these sketches directly, we must better leverage the computer's ability to help organize the user's own strokes as she draws.

5. Future Work

Our pilot study showed the promise of LogiSketch, and we are actively working to improve both the system's recognition and its interface, particularly for large circuits. One of the central challenges to recognition is that digital logic symbols all look visually very similar. We are looking into algorithms that could pinpoint salient features in the shape (e.g., the curved back of an OR gate vs. an AND gate) to and weight these features more prominently in the recognition process. On the interface side, we are exploring methods for organizing the user's strokes while still leaving them "rough" so that the user still has the experience of their drawing coming to life.

6. Additional Resources

For videos and downloads see `http://www.cs.hmc.edu/~alvarado/research/sketch.html`.

7. Acknowledgements

This work is supported by NSF grant number 0546809.

8. References

[1] Alvarado, C., Lazzareschi, M. Properties of Real World Digital Logic Diagrams. In *Proc of 1st International Workshop on Pen-based Learning Technologies.* 2007.

[2] Buchanan, S., Ochs, B., and LaViola, J. "CSTutor: A Pen-Based Tutor for Data Structure Visualization", *Proc. of the 43rd Technical Symposium on Computer Science Education (SIGCSE 2012),* 565-570, February 2012.

[3] Cossairt, T. and LaViola J., SetPad: A Sketch-Based Tool For Exploring Discrete Math Set Problems, *Proceedings of the Ninth Eurographics/ACM Symposium on Sketch-Based Interfaces and Modeling (SBIM 12),* 47-56, June 2012.

[4] Do, E., Gross, M. D. Drawing as a means to design reasoning. In *Proc. of Artificial Intelligence in Design '96 Workshop on Visual Representation, Reasoning and Interaction in Design*, Palo Alto, CA. 1996

[5] Kara, L.B., Stahovich, T. F. An Image-Based, Trainable Symbol Recognizer for Hand-drawn Sketches. *Computers & Graphics* 29(4): 501-517 2005.

[6] Labahn, G., E. Lank, S. MacLean, M. Marzouk, and D. Tausky, MathBrush: A System for Doing Math on Pen-Based Devices, In *Proc of the Eighth IAPR International Workshop on Document Analysis Systems,* Washington, DC, USA, IEEE Computer Society, 2008.

[7] LaViola, J., Zeleznik, R. MathPad2: A system for the creation and exploration of mathematical sketches. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 23(3), 2004.

[8] Lee, W., de Silva, R., Peterson, E. J., Calfee, R. C., Stahovich, T. F. Newton's Pen: A pen-based tutoring system for statics. *Computers & Graphics*, 32(5), (Oct 2008), 511-524.

[9] Ouyang, T. Y., Davis, R. ChemInk: A Natural Real-Time Recognition System for Chemical Drawings International Conference on Intelligent User Interfaces, IUI 2011.

[10] Peterson, E., Stahovich, T. F., Doi, E., Alvarado, C. Grouping Strokes into Shapes in Hand-Drawn Diagrams. In *Proc. of the 24th AAAI Conference on Artificial Intelligence*, 2010.

[11] Ullman, D. G., Wood, S., Craig, D. The Importance of Drawing in the Mechanical Design Process. *Computer & Graphics* 14(2), 263-274, 1990.

[12] Valentine, S., Vides, F., Lucchese, G., Turner, D., Kim, H., Li, W., Linsey, J., Hammond, T. Mechanix: A Sketch-Based Tutoring System for Statics Courses. In *Proc. of Innovative Applications of Artificial Intelligence (IAAI 12)*. Toronto, Canada, 2012.

[13] Wais, P., Wolin, A. and Alvarado, C. Designing a Sketch Recognition Front-End: User Perception of Interface Elements. In *Proc. of Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM)*. Riverside, CA. 2007.