Properties of Real-World Digital Logic Diagrams

Christine Alvarado
Harvey Mudd College
Claremont, CA
alvarado@cs.hmc.edu

Michael Lazzareschi
Pomona College
Claremont, CA
michael.lazzareschi@gmail.com

Abstract

Despite the growing number of sketch recognition systems for education, little attention has been paid to how students actually draw in practice. We examine freely-drawn digital logic diagrams created by students in an electrical engineering class in order to inform the design of a sketch recognition digital circuit simulation tool. Our analysis reveals considerable drawing style variation between students and that standard drawing style restrictions made by sketch recognition systems to aid recognition generally do not match the way students draw. We identify drawing behaviors that can aid recognition while placing fewer unnatural constraints on students' drawing styles, and we describe specific recognition tasks whose solutions will lead to more robust free-sketch recognition systems.

1. Introduction

Diagrams play a central role in education. Particularly in engineering, diagrams allow students to reason about a physical system, such as a circuit or a mechanical device. Simulation programs enhance the power of diagrams by allowing students to interactively explore the behavior of their design. Yet these programs are fundamentally limited by their mouse and keyboard interfaces. Menus and buttons prevent students from simply drawing their desired symbols, forcing them continually to consult menus to choose pieces of the diagram.

Tablet computers provide a pencil-and-paper-like interface, allowing users to sketch directly on the screen, alleviating many of the constraints of the traditional mouse and keyboard interface. A number of recent educational technologies attempt to combine the freedom of drawing on paper with the power of computer simulation tools in a number of domains including physics [6], chemistry [9], and electrical engineering [4]. The power of these systems comes from their ability to recognize a student's hand-drawn strokes as symbols in a particular domain.

One of the most difficult problems in creating a sketch recognition system is handling the trade-off between ease of recognition and drawing freedom. The more a system constrains the user's drawing style, the easier recognition becomes. Existing sketch recognition systems place a variety of restrictions on the way users draw in order to aid recognition. For example, some systems require users to draw each symbol with a single stroke while others require users to pause between symbols.

Although these restrictions aid recognition, researchers have paid little attention to how well they match the way students naturally draw. These restrictions may force students to change their drawing style so much that the burden of using the system outweighs the benefit. Of course, several factors influence how many and which restrictions users will accept: the utility of the tool, the domain and task, and how much of a burden these restrictions place on their drawing style. Nevertheless, we believe that the fewer restrictions we place on the user *without sacrificing recognition accuracy* the better. To leverage students' natural drawing behavior to improve recognition accuracy, we first must understand this behavior.

This paper examines how students naturally draw digital logic diagrams in order to inform the design of a recognition system for education. We focus on three aspects of students' drawing styles:

- **Stroke order:** Do students complete one symbol before moving to the next?
- **Stroke timing:** Do students pause between drawing different symbols?
- **Stroke number:** How many strokes do users draw per symbol? Do users draw more than one symbol with a single stroke?

We focus specifically on how *students* draw because how they draw may differ from how experienced designers draw. For example, students' symbols are probably messier, and their circuit construction may be unconventional. Educational software must cope with these idiosyncrasies.

We focus on a single domain because previous work has shown that domain-specific knowledge is essential in de-

signing a robust recognition system, and systems tailored to a particular domain out-perform general recognition engines [1, 4].

Finally, we focus on digital logic diagrams in particular because little work has been done in this domain, and digital circuits present important challenges for recognition: Many of the gates are similar in appearance, and the fact that the form of wires is not defined by a consistent shape leads to a great variety in possible drawing styles. Handling these challenges likely will lead to advances that can inform recognition algorithms in other domains.

Examining the above questions, this paper makes three contributions to the development of sketch recognition-based educational technologies. First, it examines how well standard drawing-style restrictions match the way students naturally draw. We find that students' natural drawing patterns violate standard restrictions up to 34% of the time, and that there is considerable variation between students. Second, our analysis informs the design of a free-sketch recognition system for digital circuit diagrams. We identify new recognition challenges that arise from the way students draw in practice, and we identify aspects of students' natural drawing styles that can aid recognition. Third, we believe similar drawing patterns exist in other domains, and our analysis framework provides a template to investigate this belief.

2. Background and Related Work

To understand why sketch recognition systems almost always incorporate drawing style restrictions, one must understand precisely why sketch recognition is difficult.

We focus on the task of *stroke-based* sketch recognition. Unlike a static digram (e.g., a diagram drawn on paper and then scanned into the computer), our input comes from a digital collection device such as a tablet computer or a digitizing pen. Stroke-based sketches consist of a set of *strokes*, or collections of points sampled from when the user put the pen down until the user lifted the pen. Each point in a stroke has both position (x, y) and time information.

We highlight two core challenges of stroke-based recognition: the problem of partitioning the strokes into individual symbols (*stroke grouping*) and the problem of identifying the individual symbols (*symbol identification*).

Stroke grouping is difficult because it is inherently linked with symbol identification. Figure 1 shows a sketch from our data set. (We altered the thickness of the strokes for this example). If the system could correctly group the three bold strokes in this sketch, it likely could identify those strokes as an XOR gate using a standard pattern matching technique such as a neural network. Unfortunately, stroke grouping is not as easy as it appears. Simple spatial and temporal grouping approaches do not work: the three strokes that

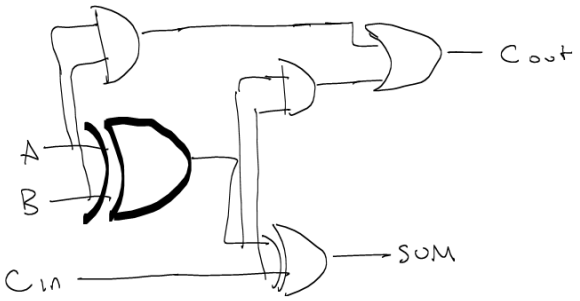


Figure 1. A typical digram from our dataset (stroke thickness altered for illustration).

form the XOR gate are not all touching each other, but they are touching the input and output wires. In fact, the reason these three strokes should be grouped together is precisely because they form an XOR gate. But the system cannot tell that they form an XOR gate until it knows to group them together. In other words, if the computer can find the correct grouping, it will be able to match the strokes to a shape in its library. However, naively trying all combinations of stroke groups is prohibitively time-consuming.

Symbol identification is difficult because of the potential variation in the way users draw shapes. For example, a user may draw an AND gate with one stroke, two strokes, or more. This variation presents a challenge because a system cannot know how many strokes each object will contain, nor the order in which these strokes will appear.

Most existing recognition systems place restrictions on the user's drawing style that mitigate one or both of the above challenges. Here we list four common drawing style restrictions that address these challenges, ordered from most restrictive to least restrictive, and give examples of systems that use each:

1. Users must draw each symbol using a pre-specified pattern or gesture. (e.g., Palm Graffiti®, ChemPad [9])
2. Users must trigger recognition after each symbol (or pause notably between symbols). (e.g., HHreco [5], QuickSet [3])
3. Users must draw each symbol using temporally contiguous strokes. (e.g., AC-SPARC [4])
4. Users may not use a single stroke that spans more than one object (e.g., SketchREAD [1]).

Finally, some systems place few restrictions on the way users draw, but rely on user assistance in stroke grouping. To trigger recognition in MathPad^2, for example, the user must circle pieces of the sketch [6].

Although many systems restrict the way users can draw, little work has been done on understanding how people draw naturally. Oltmans *et al.* present some preliminary

observations from a dataset of sketches from several domains collected in a laboratory setting [7]. We provide a more thorough and sound analysis of their preliminary findings. Shilman *et al.* have examined free-from handwritten notes in detail, specifically focusing on identifying structure and handwriting within these notes [8]. We focus on properties of diagrams extracted from within freeform notes. Finally Anderson *et al.* examined diagrams created by professors while lecturing [2]. Because we aim to build tools to support student learning, it is critical to understand the students' behaviors, in addition to the professors'.

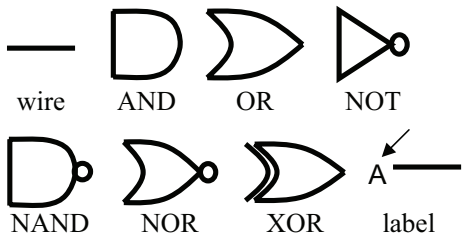


Figure 2. The symbols in the digital circuit domain

3 Data collection and analysis

We collected the complete set of sketches and notes produced by students in a digital design and computer architecture class. We gave each student participant a tablet computer for the entire semester and asked them to use Windows Journal whenever they would normally use paper: in their note-taking, problem sets and circuit design labs. Our complete dataset consists of hundreds of files containing handdrawn notes, equations, and diagrams. Our current study focuses on the diagrams students drew when completing their problem sets and labs because our sketch-based simulation tool will support these aspects of the course. We extracted diagrams from the first three weeks of students' problem set and lab work (after the first three weeks the course focus moves away from low-level circuit design).

We converted the Journal files to our own format and then hand-labeled each diagram by grouping strokes into objects and tagging each stroke group with the appropriate symbol name from the symbols in Figure 2, or "other" if it did not match any of these symbols. In grouping wires, we grouped strokes along a single path from one object to another. We grouped wire strokes that split off from other wires into a separate wire symbol. For example, in Figure 1, we grouped the horizontal wire from input A to the bold XOR gate separately from the bent wire that progresses upward into the AND gate. Even though these two wires carry the same signal, we consider them separate objects because they have different destinations.

The dataset we used for this study consists of 98 diagrams from 13 students. The number of diagrams analyzed per student ranged from 4 to 11 (different students drew different numbers of diagrams when completing the same assignments). The average number of strokes per diagram was 50 (min=12, max=127, s.d.=24) and the average number of symbols per diagram was 24 (min=7, max=82, s.d.=10).

We examined both quantitative and qualitative properties of our data. We gathered statistics from the data to examine patterns in the three properties listed in Section 1: stroke order, stroke timing and stroke number. To investigate stroke order we counted how many of each symbol type students drew with temporally contiguous strokes and how many they did not. A symbol is drawn with temporally contiguous strokes if the strokes in the symbol are not interrupted in time by strokes in a different symbol. To investigate stroke timing we compared the pause time (i.e., the time between the last point in one stroke and the first point in the next) between consecutive strokes in the same symbol (e.g., the triangle and circle in a NOT gate) to the pause time between consecutive strokes in two different symbols (e.g., the circle in a NOT gate, and the wire connected to the gate). Finally to investigate how consistent students are in drawing each type of symbol, we counted the number of strokes each student used to create each type of symbol. We also considered how often a single stroke spanned more than one symbol.

When we observed an interesting trend in the quantitative analysis, we visually examined the spatial and temporal properties of individual sketches to help explain our observations. Often this analysis included replaying the strokes in the order in which they were drawn to understand how the student created his or her sketch.

4. Results

4.1 Stroke Order

How often students draw symbols with consecutive strokes varies both by symbol type and by student. Figure 3 illustrates how often students drew each type of symbol with consecutive strokes. Overall, students drew 19% of all symbols and 14% of gates (not including wires, labels or other) with non-consecutive strokes. Figure 3 also reveals that some symbols almost always consist of consecutive strokes (e.g., XOR, NOT), while others do not (e.g., NAND, Wire). In addition, on average a student drew 82% of her symbols with consecutive strokes, but this percentage varied from 96% to 70%.

Examining individual sketches, we found two qualitatively different drawing patterns underlying symbols drawn

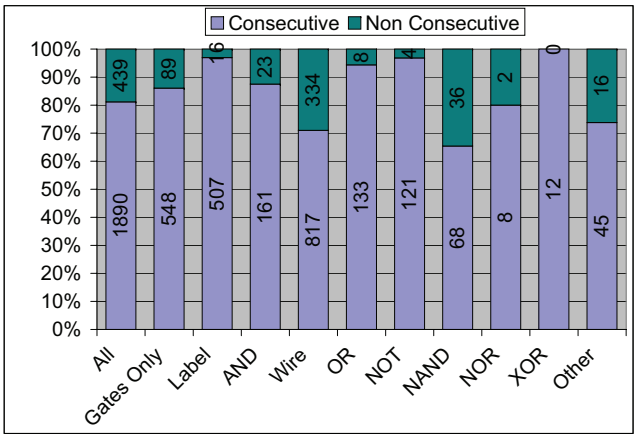


Figure 3. The percentage of each symbol drawn with consecutive and non-consecutive strokes. The numbers indicate the total number of each type of symbol.

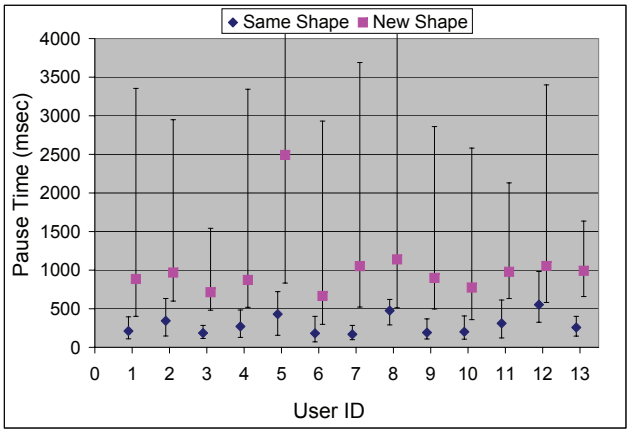


Figure 4. The median pause time between consecutive strokes in the same symbol and in different symbols.

with non-consecutive strokes. In some cases, the student drew the majority of the symbol using consecutive strokes, but returned to the symbol later in her drawing process to add a single stroke. These single, non-consecutive strokes appear to correct a part of the gate that was not fully connected when the gate was drawn, or simply to trace over an existing part of the gate, perhaps while the student contemplated what to draw next.

In other cases, the student left a symbol (frequently a wire or a NAND gate) obviously unfinished and returned later to finish it. For example, students often drew a portion of a wire, moved on to draw a gate some distance from the end of the wire, and then returned to extend the wire to connect to the gate. In many cases when drawing a NAND gate, students drew the body of the gate, moved away and drew some wires, and then returned to draw the "bubble" on the end. At first glance it appears that students realized sometime after drawing the AND gate that they wanted to invert the gate's output. But in fact we observed this behavior when students were instructed to build a circuit using *only* NAND gates (and wires). It seems that because they knew they were going to create a NAND gate, they simply placed enough of the gate on the page to attach its inputs, and added the bubble on the end only when they added the output wire.

4.2 Stroke Timing

Our analysis reveals that while students do pause longer when switching symbols, how long they pause between symbols overlaps greatly with how long they pause between strokes in the same shape. Furthermore, how long students pause between strokes varies from student to student.

Figure 4 shows the median pause time per student between two strokes that are part of the same symbol and two strokes that are part of different symbols. The error bars on the graph show the interquartile range[1]. The data points on this graph show that students do generally pause longer when switching between symbols than when continuing to draw the same symbol. This difference is significant for all students (Wilcoxon rank sum, $p << 0.001$).

Unfortunately, the error bars reveal significant overlap between the two distributions for many students, making it impossible to find a reliable time threshold that would indicate the start of a new symbol. To investigate pause time as a discriminative measure, we measured how many errors per user would result from using the single best user-specific pause time threshold to classify each stroke as continuing the previous symbol or not. Table 1 shows the results of this analysis. Note that pause time is a more reliable indicator of a symbol boundary for some students (e.g., 3 and 13) than for others.

4.3 Strokes per Symbol

Finally we find that students vary in the number of strokes they use to draw each symbol and that many students themselves do not consistently draw each symbol with the same number of strokes.

Figure 5 illustrates how many strokes each student drew when creating each of the five major circuit symbols: AND gates, NAND gates, OR gates, NOT gates and wires. The graphs plot the fraction of each type of gate that the student

[1]We analyze the data using non-parametric methods because normal probability plots revealed that our data is not normal.

User ID	Threshold (msec)	Error (%)
1	326	20.3
2	519	21.1
3	385	10.6
4	227	20.7
5	471	16.3
6	264	27.2
7	312	16.2
8	396	29.1
9	325	17.9
10	155	22.4
11	391	18.2
12	577	34.7
13	475	14.8

Table 1. Total classification error (new vs. same symbol) for optimal time threshold

drew with one stroke, two strokes, three strokes, etc.

These data reveal several trends. First, in general the number of strokes per symbol varies quite a bit across students. For example, student 10 tended to draw AND gates using one or two strokes, while student 8 drew AND gates mostly with 3 or 4 strokes. Additionally, this variation is greater for some symbols than for others. Most of the time most students drew NAND gates with exactly three strokes (although student 8, who used three strokes to draw AND gates, used more strokes to draw NAND gates). On the other hand, students drew OR gates with a wider range of numbers of strokes.

Second, students vary in how consistent they are in drawing each individual type of symbol. Some students consistently used almost exactly the same number of strokes every time they drew a symbol (e.g., students 1, 3, 4, 13); others varied widely in how many strokes they used per symbol (e.g., students 8 and 12); while the rest displayed moderate variation. Individual students, however, tend to be either consistent or inconsistent in their style across all symbol types. For example, the students who were not consistent in how they drew their AND gates also were not consistent in the number of strokes they used to draw wires.

Not surprisingly, wires show the widest variation in stroke number both between students and for each student individually. This variation likely arises from the free-form nature of wires. Still, Figure 5(f) shows two important trends. First, most of the time most students draw each wire with a single stroke. Second, several students (students 3, 4, 10, and 13) draw the vast majority of their wires using three or fewer strokes. Note that these students do not necessarily tend to use fewer strokes to draw the other symbols.

Finally, we found that students in our study rarely used a single stroke that spanned multiple symbols (fewer than five instances in the entire dataset). We believe that this trend is quite domain-specific, as others have observed the tendency for students to draw multiple symbols with a single stroke in other domains, such as mechanical engineering [7, 4].

5 Discussion

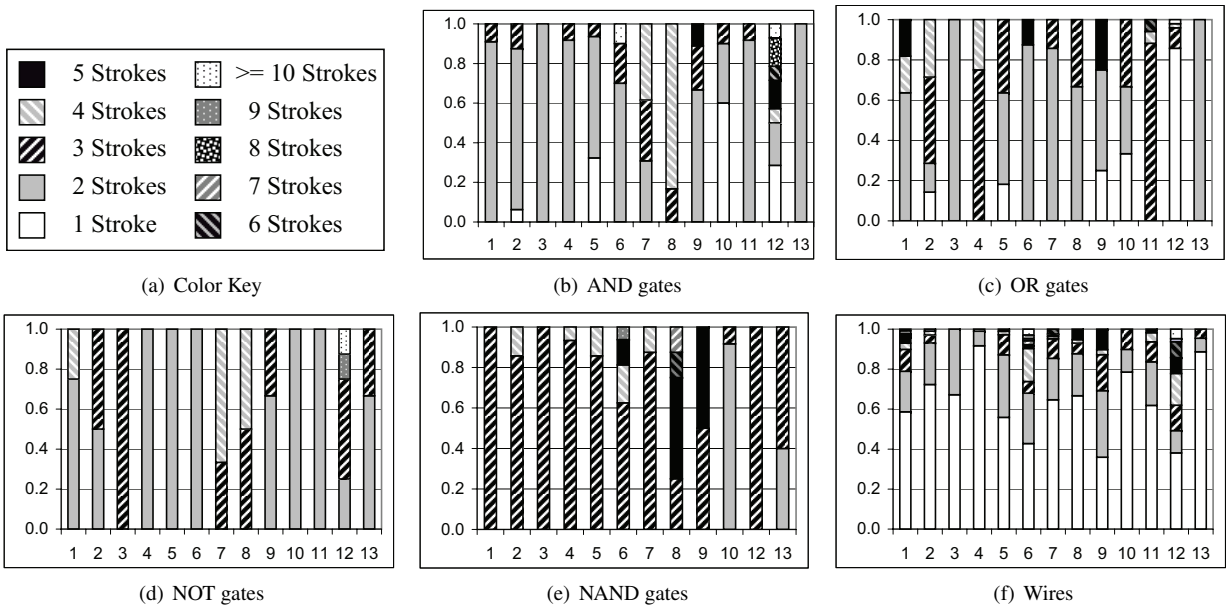
Here we consider how our results inform the design of digital circuit sketch recognition systems for education. We identify trends in the way students draw that will aid recognition, and we describe specific recognition tasks that must be solved to build robust free-sketch recognition systems.

Pause time can aid stroke grouping. In the median case, students did pause about half a second longer between strokes when they started drawing a new symbol than when they continued with the same symbol. While a student's natural pause time alone is not sufficient to reliably identify the boundaries between symbols (because of the overlap in the time distributions), it can be combined with other stroke properties (such as position or length) potentially to improve stroke grouping. Furthermore, teaching users to extend how long they pause between symbols even by milliseconds might vastly improve grouping and, consequently, recognition. Direct user testing is necessary to determine whether or not adapting to a slightly longer pause time interrupts a student's work flow.

User-specific learning may aid recognition. Overall we found that few (if any) properties were consistent across all students. However, in many cases, students themselves were consistent across symbols and sketches. A recognition system that can learn how (and how consistently) a user draws each symbol, how often she draw symbols with non-consecutive strokes, how long she pauses between symbols, etc., will likely outperform a recognition system that uses a single model for all users. The challenge will be to identify user-specific properties reliably without requiring the student to do excessive work.

Recognition systems must identify symbols that are not drawn with consecutive strokes. While students draw many shapes with consecutive strokes, a recognition system must cope with non-trivial number of cases where they do not. The specific drawing patterns we identify in Section 4 will aid recognition systems in this task. For example, tailored recognition algorithms might identify "touch-up" strokes and handle them differently from strokes that add significant visual content. Recognition routines that identify wires can expect the user to extend the end of the wire with a stroke later in time, but not to add a stroke to the middle of the wire.

Symbol recognizers must incorporate a wide range of drawing styles (for most users). We saw that students varied in the number of strokes they used to draw each symbol. Furthermore, some individuals were inconsistent in



(a) Color Key (b) AND gates (c) OR gates

(d) NOT gates (e) NAND gates (f) Wires

Figure 5. Fraction of gates drawn with different numbers of strokes, by student.

how many strokes they used to draw each symbol. Forcing these students to draw a symbol with a specific number of strokes is likely to interfere with their drawing style and thought process. On the other hand, if a system can identify a student who is consistent in the way he or she draws gates and wires, it can tailor its recognition routines to that student to improve recognition.

6 Conclusion

Sketch-based technologies have the potential to revolutionize education, but we must match these technologies to what the student does naturally so that they support the learning process, rather than interfere with it. We present drawing trends from a single domain in order to inform the design of a sketch recognition-based tool for digital circuit design, but our analysis framework can, and should, be applied to other domains. The results of these analyses, including the results presented here, reveal important sketching behaviors that can be used to improve recognition without placing unnatural constraints on the way students draw, leaving students free to think about the material they are learning, instead of focusing on exactly how to draw it.

7 Acknowledgements

We thank all of the E85 student participants. We also thank Sarah Harris, Ran Libeskind-Hadas, and Alex Snoeren for helpful editing feedback. This work is supported in part by an NSF CAREER award (IIS-0546809).

References

[1] C. Alvarado and R. Davis. SketchREAD: A multi-domain sketch recognition engine. In *Proc. UIST*, 2004.

[2] R. Anderson, R. Anderson, C. Hoyer, C. Prince, J. Su, F. Videon, and S. Wolfman. A study of diagrammatic Ink in lecture. *Computers and Graphics*, 29(4), 2005.

[3] P. R. Cohen, M. Johnston, D. McGee, S. Oviatt, J. Pittman, I. Smith, L. Chen, and J. Clow. Quickset: Multimodal interaction for distributed applications. In *ACM Multimedia'97*, pages 31–40. ACM Press, 1997.

[4] L. Gennari, L. B. Kara, and T. F. Stahovich. Combining geometry and domain knowledge to interpret hand-drawn diagrams. *Computers and Graphics*, 29(4):547–562, 2005.

[5] H. Hse and A. R. Newton. Recognition and beautification of multi-stroke symbols in digital ink. *Computers and Graphics*, 29(4), 2005.

[6] J. LaViola and R. Zeleznik. MathPad^2: A system for the creation and exploration of mathematical sketches. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 23(3), 2004.

[7] M. Oltmans, C. Alvarado, and R. Davis. ETCHA sketches: Lessons learned from collecting sketch data. In *Making Pen-Based Interaction Intelligent and Natural*, 2004.

[8] M. Shilman, Z. Wei, S. Raghupathy, P. Simard, and D. Jones. Discerning structure from freeform handwritten notes. In *Proc. ICDAR*, 2003.

[9] D. Tenneson. Technical report on the design and algorithms of chempad. Technical report, Brown University, 2005.