State of the Research: Circuit Simulator Front End

Fiona Foo and Andy Kearney

Summer 2011

## Goals

Our goal for this summer was to improve upon and existing pen-based interface for circuit recognition software that allows the user to create and interact with logic circuits in a natural and efficient way. Our primary objective was to create intuitive feedback from the recognition that users could easily understand and interact with. Additionally we wanted to extend the implementation to allow for the embedding of sub-components within circuits.

The motivation for the easy to understand feedback was to provide a pleasant way to correct a circuit in the presence of recognition errors. We hoped to communicate effectively how the circuit recognized gates provide information that the user could use to adapt their drawing style. Additionally we wanted to show errors with parsing the circuit and provide simple ways to fix these errors such as connection or orientation errors.

## Informal User Study Findings

### Initial Study

##### Design

The intention of the initial informal user study was to get a grasp on the direction we should be heading for the summer. With this in mind, our initial study used the actual system as it was at the end of the summer of 2011. This allowed us to view the interactions of first time users with the system and observe major issues with the current version.

The first task was to draw a circuit using the system from a picture and have the system correctly recognize it. This involved using the system or redrawing until it was recognized correctly. Next the user was asked to simulate the circuit and experiment with the toggles and truth table.

The user was asked to speak their thoughts as they interacted with the system, which gave feedback as to the aspects that the user enjoyed or was frustrated with. Additionally we had two members of the User Interface team taking notes and looking for aspects of the system or interactions the user struggled with.

##### Results

The study provided useful insight particularly to the hover widget buttons as well as to the need for feedback. Users found that the buttons did not look clickable so they were less inclined to explore the system. Also, the buttons appeared below ink on the sketch, which was a clear problem with the circuit.

A key finding for the direction of our summer was the lack of feedback provided to the user. The version in place had gates be uniquely colored as well as the name of a gate display when the user hovered over a gate. Users repeatedly did not see or understand these feedback mechanisms, which greatly limited their effectiveness in correcting or even seeing errors.

A smaller finding was with the ambiguity of some of the functions of buttons. For instance, changing the interpretation of a gate was activated through a button called “Label” but inputs and outputs were also called labels. Additionally when strokes were selected the option to “Delete All” appeared, which some took to mean delete the selection. In reality this option deleted all strokes on the screen.

### Embedding Sub-Circuits 1

##### Design

The motivation for this study was to explore possible interfaces for the ability to add previously saved circuits as components to current projects. Since this capability was not implemented at the time, we used paper prototypes to simulate the experience. This allowed us to test out and eliminate some ideas and gather opinions without wasting time fully implementing an undesirable interface.

The task for the user was to create a circuit that we showed to the user which consisted of two full adders forming a four bit adder. The user would use each proposed interface, in a random order, to construct the circuit. User were encouraged to discuss their opinions as they used each interface and were interviewed after finishing all interfaces.

We ended up eliminating one possible interface early because of a strong reaction against it and because it required the system to be recognized already, which defeated the purpose of our embedding system. The four remaining possible interfaces were:

1. [On the fly recognition notices a particular shape and prompts user to select a circuit to load](https://www.cs.hmc.edu/twiki/pub/Sketchers/InformalStudyNotes/1-onfly.pdf)
   * When you draw a rectangle, the system will automatically ask you if you want to embed a sub-circuit. If you click the Embed button, you will be asked to load in a sub-circuit.
2. [Clicking an Add Sub-circuit hover widget prompts user to select a circuit to load](https://www.cs.hmc.edu/twiki/pub/Sketchers/InformalStudyNotes/2-hoverwidget.pdf)
   * When you hover over a point, the system will show three buttons. If you click the Embed button, you will be asked to choose a circuit to load.
3. [Load Sub-circuit option under File menu prompts user to select a circuit to load](https://www.cs.hmc.edu/twiki/pub/Sketchers/InformalStudyNotes/3-file.pdf)
   * The File menu includes the option to Load Sub-circuit, and if you click that option, you will be asked to choose a circuit to load.
4. [Pop-out menu with completed circuits that allows you to drag in components](https://www.cs.hmc.edu/twiki/pub/Sketchers/InformalStudyNotes/4-embed-tab.pdf)
   * When you click the Embed button on the left side, a list of circuits will pop out. You can drag the circuit name into the main panel.

##### Results

In general we found that the user preferred the options with the fewest clicks and the greatest discoverability. The two favorite methods were the pop-out menu and the hover widget. Users said that these two options would be very easy to find because there was explicitly a button for it. Also, users said that these options would be easiest to add multiple sub-circuits conveniently (with a preference for dragging in from the left menu).

Based on these results we planned to implement these two interfaces and have another study. The motivation for wanting this study was because it was difficult to simulate dragging with a paper prototype since one of us was acting as the computer.

### Embedding Sub-Circuit 2

##### Design

Using the first embedding sub-circuit study as a guide, the embed panel and hover widget were implemented into the system. The goal of this study was to determine any improvements needed for the embedding sub-circuit interface, as well as to explore any other problems with the system that needed to be addressed before release.

The task for the user was to create a full adder from an equation that was provided to them and get the system to simulate the circuit correctly. This purpose of this stage was to allow the user to explore the system as well as complete the full process of creating and using sub-circuits. When this stage was completed, the user was asked to create a two bit ripple carry adder using each of the embed options in a random order. Users were asked for general feedback for the system as well as specific feedback for the embedding interfaces after each circuit was constructed.

##### Results

In general the use of the embedding sub-circuit interface went smoothly. Both users reported having a very easy time using the saved circuit with both of the proposed interfaces. Based on the comments of the users, we decided to keep both designs in the system because of the way they work together. The users said that when loading in a sketch for the first time it’s better to use the hover widget, but that the embed panel would be better for adding in a lot of circuits to a sketch. As a result, the embed hover widget now adds to the list of embeddable circuits to the list.

The main issues with the system at the time of this user study had to do with selecting strokes. This was reportedly the most frustrating aspect of the system because of the current bugs. An example of these bugs is that at this time, it is not possible to leave selection mode without use of the stylus button because of recent work with event handling. The only other reported frustration was with the failure of the recognizer and refiner to understand what was meant with the drawing. However users seemed to have a relatively easy time to correct most errors.

Users did however have some trouble moving and rotating gates. It was not clear to the user that had such problems that the gate needed to be rotated to correct the inputs and outputs until given direction. Also, it was not clear to the users how to move the gates as opposed to rotating them. An interface for distinguishing the two options must be implemented to avoid confusion and frustration when interacting in this way.

### Overall Results

In general, the most lacking aspect of the program in terms of the front end from the previous summer was the feedback provided to the user. Errors with circuits or recognition were not explained well or at all in some cases. Circuits would occasionally recognize correctly but would not be able to simulate because of hidden parsing errors. Additionally these problems, when exposed, needed easy methods of correcting in order to avoid frustration.

For the interfaces considered for embedding, users generally enjoyed the option that required the least amount of work to use. The option of accessing the feature through the toolbar was commonly cumbersome for users. The options favored by users were the side panel and the embed widget, each of which was cited as being easy to discover as well as requiring fewer clicks on the user’s part.

## State of the User Interface

### Feedback

All feedback options may be enabled or disabled from a menu on the toolbar of the main window. This allows the user to customize the experience of correcting circuits to their liking, depending on which they find helpful. The main additions to feedback are error highlighting as well as varying options involving ghost gates. A ghost gates is a dotted representation of an ideal gate that is displayed behind the actual gate drawn by the user. The current possible feedbacks are:

1. Shapes while drawing – This option will recognize gates as the user draws and display a ghost gate behind the shape of the gate the drawing was recognized as.
2. Labels after Rec – This option will show a text tooltip on all gates after recognition of the name of the gate that it recognized each as.
3. Shapes after Rec - This option will show ghost gate on all gates after recognition of an ideal template of each gate.
4. Shapes after Label – This option will display a ghost gate behind the strokes when an incorrectly recognized shape is labeled by the user. The intention is that the user will be able to see a picture of the gate so that they can draw more like it in the future.
5. Shapes on Hover - This feature will allow the user to see the ghost gate of any gate by hovering the stylus over that gate.
6. Highlight possible errors – This option will add yellow highlighted boxes to the screen over areas where parsing failed. The highlighted boxes will display a text box with the error and tips for correcting it when the user hovers over the box. These boxes are displayed on recognition, but update as the errors are corrected.
7. Color shapes – Shows the gates colored when they are recognized as a way to distinguish between gate types.
8. Highlight Endpoints – Correctly connected endpoints of wires will be shown as a green circles and disconnected endpoints are red x’s.
9. Highlight meshes – This feature will emphasize any strokes the user hovers over as well as the connected components.
10. Tooltips – This will display a text box containing the gate type of any gate when it is hovered over.

These feedback mechanisms allow for the option of viewing how the recognizer and parser are behaving without requiring that the user use them. The system has also been improved to include several error correcting techniques based on the feedbacks. The first of which is orientation errors with the gates. The recognizer rotates gates around to find the best match for the strokes to templates. Occasionally this will result in incorrect rotations, which causes the gates to not receive inputs from the correct places. With any of the ghost gate features enabled, this problem becomes clear and is easily fixed by dragging and rotating the ghost gate to match the expected orientation. The second error correcting technique deals with connection errors with wires. Wires will not connect automatically to gates to far away, or will think they are connected by an internal endpoint to wires close to the endpoint. With error highlighting, gates with incorrect numbers of inputs and outputs are signaled with messages and yellow highlight boxes. This information along with endpoint highlighting allows the users to quickly identify these problems. In order to fix these issues as they arise, endpoints can now be dragged to components to explicitly extend the wire to the new shape.

Shapes while drawing feedback was designed with the intention of peeking into the recognition, while being unobtrusive to the user. Since it has been found previously that users did not like the system changing their drawing while they were still drawing, this feedback avoids directly changing the ink in any way. It makes use of a new recognition pipeline that is run when strokes are collected that only runs up through recognizing the gates. Then the most recent three gates recognized have ghost gates drawn behind them. The number was reduced to three in order to avoid clutter on complicated circuits.

Additionally, a separate window, the Template Window, has been implemented to view the template that the image recognizer has matched the strokes to. The intention of this was to give the user a better understanding of why the system recognized their strokes in some way. The window is brought up with an expander, similar to the way that the notes window is activated. The window itself is told when the user hovers over a gate, and will update the image shown to the template that the particular gate being hovered over used.

### Practice Window

The practice window is a separate drawing window that is triggered from an expander on the toolbar of the main window. The purpose of the practice window is twofold: to give the user an idea of what the recognizer is looking for and to train the recognizer on how the user is drawing. The window has a drop down menu that allows the user to select a gate to practice drawing. Upon selecting a gate, the window will bring up a ghost gate on the canvas. The user will trace the drawing or draw their version of the gate on the screen and then click the recognize button. The recognition of this is much simpler than the main window because the recognizer already knows what they have drawn is one gate. The window displays the type that the gate is recognized as and a confidence rating, as to provide feedback to the user about how well their drawing of the gate matched.

Through the recognition and use of the ideal gate, the user is hopefully trained to draw more like the template. The recognizer also learns through this process though. Since it is known what the user meant to draw, the template can be added to the image recognizer and phase out old, outdated exemplars of the gates. Though this process, the user and the recognizer work together to improve the recognition of their circuits in the future. The window also has the option to freehand to see how the recognizer would recognize things without choosing one gate, which does not train the recognizer since the intended gate is not known.

### Sub-Circuits

The addition of sub-circuits to our system is important because it is a feature required in order to complete several of the home works for CS5. This addition raised several complications that we needed to address. First of all we needed to change the representation of gates and circuits to allow for multiple, distinct outputs from a single source. Also, since we wanted the user to be able to view sub-circuits without adding them to the sketch, we needed to make a recursive data structure to hold the sub elements. Finally, because we’ve added multiple recursive elements to circuits, saving and loading needed to be changed in order to be able to save and load these behaviors.

Inputs and outputs are now indexed so that multi-input/output gates are correctly calculated. Circuit elements are created for each sub circuit shape that is added and is kept around to hold onto important behavior information.

We created a data structure called Project that holds on to all important information needed to be saved and loaded within our program, which includes the behavior and the sketch. A project contains a list of any subprojects used, which allows saving and loading with xml files recursively possible. Additionally, with the embed panel we allow the user to view any sub-circuit that has been added to the list. This will load in the project for that sub-circuit and display the sketch in a new window. Any sub-circuits within that window are also able to be clicked to view in a new window. These features also display the truth table of the viewed project.

When the program is opened, previously saved files are loaded into the embed panel.

### Exporting To LogiSim

LogiSketch will be piloted on some CS 5 students this fall that will need to submit their circuits for assessment. Thus users can export their LogiSketch sketches to a .circ file which can be opened by LogiSim. This functionality is compatible with our implementation of sub-circuits and arbitrarily-deep embedded sub-circuits can be exported to LogiSim. This is implemented by saving a sketch’s LogiSim representation (generated by the SaveToCirc class) each time we save the sketch. When a sketch is exported, we compile the LogiSim representations of its subcircuits with its own representation to generate a comprehensive LogiSim file.

## Ideas for the Future

* Ghost Rotation is not obvious; so several interfaces have been suggested. The options should be tested with a user study.

|  |  |  |
| --- | --- | --- |
| Option | Pros | Cons |
| Create a separate edit menu option that allows the user to explicitly enter a rotation mode. | Avoids complications with other pen down events such as drawing and moving. | Cumbersome to rotate several gates, since the mode would have to be selected for each gate. |
| Create a separate hover widget for the ghost gates that is drag-able around the center with clear indication of arrows. | Easy to find and use on large number of gates. Also avoids events required such as moving and drawing. | Causes even more of the area around the stylus to be blocked by hover widgets. |
| Explicitly mark the area of ghost gates that can be rotated. Currently changes cursor, but this isn’t entirely clear. | Obvious and easy to use for multiple gate rotations. | More visually intrusive ghost gates. Hard to imagine how this would work with the sub circuit information |

* Allow the user to click down and drag sub-circuit boxes from the left panel, with an easily discoverable and intuitive UI.
* Path finding for wires in the clean circuit should be better.

## Known Bugs - see tickets on trac