Experiences Developing an E-whiteboard-based Circuit Designer

Ray Liu, Lisa Wong and John Grundy

Department of Electrical and Electronic Engineering, University of Auckland
Private Bag 92019, Auckland, New Zealand
john-g@cs.auckland.ac.nz

Abstract

E-whiteboards - large image display surfaces (LIDS) that support data input with pen-based sketching - have become more readily available in recent times. We describe a prototype environment making use of this new user interface technology aimed at the teaching of digital circuit design and behaviour. In this paper we focus on some of the key user interface design and implementation challenges we faced when developing a prototype of this system. We also discuss results from usability evaluation of the prototype and directions for future research based on these results.

Keywords: user interfaces, human-computer interaction, pen-based user input, large screen displays, CAD tools

1. Introduction

Typical approaches to electronic circuit design are very similar to those of CASE (Computer-Aided Software Engineering) and UI (User Interface) construction tools for software design: users interact with a desktop application performing direct manipulation and dialogue-based data input and output operations. In recent years an alternative interface technology for supporting interactive, visual language-based design has been developed: the pen-based input, large image display surface "E-whiteboard" [1, 2, 10]. This paper describes our research exploring the use of such a user interface technology for an interactive circuit designer. This simple designer was aimed at teaching elementary circuit design principles in Electrical Engineering, but its primary purpose was to investigate some of the key issues in adopting E-whiteboard-based approaches to design applications in general.

Sketching-based interfaces have been tried in a variety of application areas, both with and without large image display technologies. SILK [5, 6] provides a sketching based design environment for user interfaces where users sketch an interface and then transform it into a lower-level, formalised and computer-drawn interface specification. Amulet [8] provides a single-stoke input mechanism for rapid icon specification, using Rubine's algorithm [9]. A similar approach using a large image display with pen-based input has also been developed, integrated with the Visual Basic development tool [2]. Control of presentations and annotation of presentation slides via pen-based input on an E-whiteboard has been explored [1, 10]. Knight [3] provides a sketching-based interface for UML diagramming, creating computer-drawn UML icons from pen input guestures. Denim [7] provides a sketching-based interface for web user interface construction.

We describe the motivation for our work: the design and prototyping of a circuit designer using pen-based input on a large image display surface output device. We illustrate the user interface of our tool both for circuit design and animation of designs to illustrate circuit behaviour and briefly discuss our main implementation approaches. We discuss some user interface issues presented when designing our prototype tool and outline further areas for research and application of this user interface technology to other design environments.

2. Motivation

When designing circuits in a tutorial situation teachers typically use a whiteboard and sketch AND, OR, NOT gates and connector wires, explaining the rationale for decisions. Often they then highlight the state of inputs, outputs, gates and wires when the circuit is "run" i.e. has power supplied to its inputs. We wanted to explore the use of a large image display surface with pen-based input to provide teachers with an "E-whiteboard" circuit designer. The hardware set-up we used is illustrated in Figure 1. A picture of this E-whiteboard, or Large Image Display Surface, is shown in Figure 2.

The technology used is straightforward and aimed at providing a low-cost environment for pen-based sketching and large image display [1]. A large mounted opaque glass screen has a data projector behind it, projecting the screen of a PC onto the surface. A mimio pen-based data capture device allows the PC to capture pen movement, which is translated by software into digital images projected onto the "drawing" surface by the data projector. As the user moves their pen, a digitised set of curves appear in real-time. The user can also interact with menus, pop-ups and other UI features by tapping on them with the mimio pen input device. The mimio device also provides audio capture, transmission and playback, though this has not been used in this project.

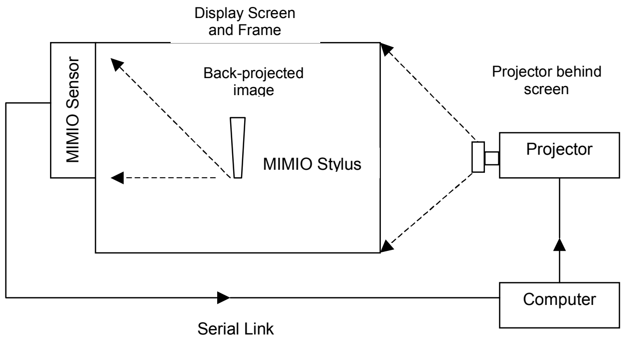

Figure 1. Hardware used for Large Screen interface.

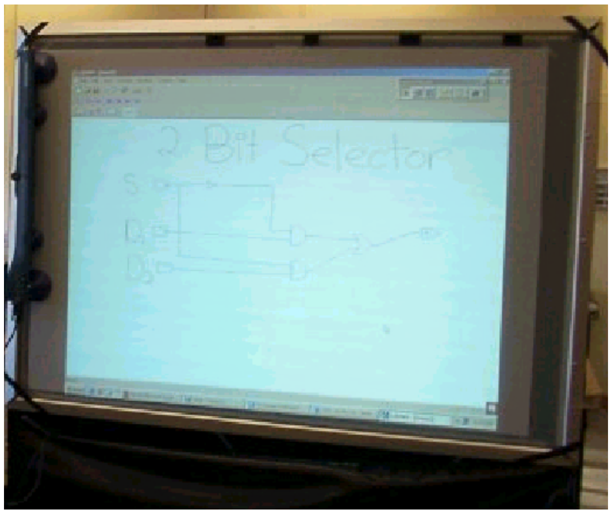

Figure 2. Example of LIDS circuit designer in use.

In this sample LIDS design application we wanted to provide a simple E-whiteboard-based circuit design tool for tutorial teachers, giving them the ability to sketch circuit elements (AND, OR, NOT, input and output gates, connector wires) with the pen, capturing this evolving circuit design digitally. We wanted to have circuit elements computer-drawn, but possibly to preserve hand-sketched circuit elements if desired. As the user draws the various elements of a circuit we wanted the designer to recognise them as particular circuit elements and build up an internal data structure to capture the circuit design specified by sketching. The user could, however, over-ride the design tools recognition algorithm if it made mistakes via a pop-up menu. Users should be allowed to move and delete circuit elements as required via direct manipulation with the input pen. A circuit design should be able to be "animated" (i.e. user specify input values to connector wires) to show the resultant output values (1 or 0) produced by the simulated digital circuit. Circuit designs must be able to be saved and reloaded and printed out.

In future design environment applications, we wanted to explore the use of the LIDS technology for CASE-style diagramming, as in Knight [3], user interface design, as in SILK [5] and [2], and building design. However, in contrast to this work we wanted to further explore issues of computer-drawn components vs sketched, and the use of the LIDS technology for collaborative, co-located vs remote-located work. The circuit designer prototype described in the following sections is being generalised to support both computer-drawn and sketched elements and remote-located collaboration to explore these issues.

3. Interface Design

We chose to provide users with a reasonably standard Windows-based user interface for our circuit designer. This is illustrated in Figure 3.

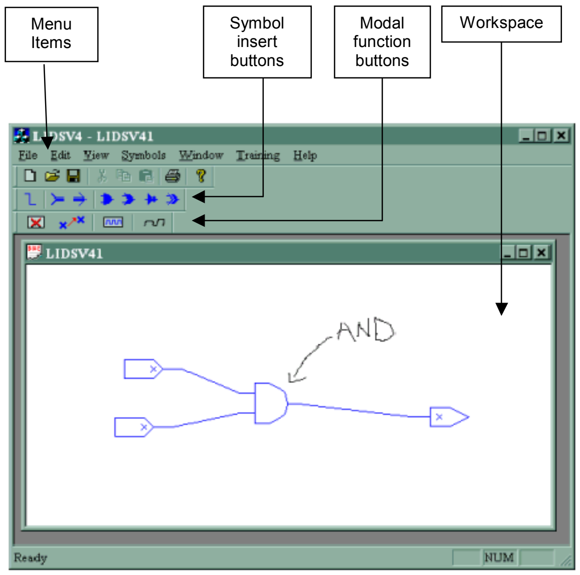

Figure 3. Interface of the circuit designer.

The mimio data capture device is configured to control the standard Windows mouse movements, with a tap of the pen on the whiteboard surface simulating a left mouse button click in that position. The interface does not make use of the right-mouse button as this currently cannot be simulated by the mimio pens. While the mimio system provides multiple pens and an "eraser", we chose to only use one pen and to provide deletion via modal button and element selection. Note the interfaces below are displayed on our large back-projected E-whiteboard surface and the circuit design manipulated only by the mimio pen-based input device.

To add gates to a circuit design three fundamental direct manipulation operations can be used: the user sketching a shape, either an abstract gesture or one corresponding to the desired circuit element, on the whiteboard board; using a pop-up menu to have a computer-drawn shape added in the selected position; or dragging a computer-drawn circuit component from a tool palette into the desired position. We chose to focus on using the first approach, as illustrated in Figure 4 (1), where the user is sketching an AND-gate. A shape recognition algorithm processes the sketched points and creates a new circuit gate, draws it and adds it to the underlying circuit data structure (Figure 4 (2)). This recognition algorithm can make a mistake and the user can change the element type using a pop-up menu by tapping on the shape. Alternative replacement approaches could be to allow the user to draw over the top of an incorrectly recognised shape and have the new one replace the old. We currently use a single-stoke recognition algorithm to recognise element types i.e. use the first single movement drawn by the user. Again, a multi-stroke recognition could be used e.g. to recognise several single-strokes, or to recognise a complex multi-stroke drawn shape, to provide a wider range of possible sketched shapes to recognise.

After adding shapes to a circuit design, including input, output, AND, OR and NOT gates, the user can connect input and output pins for these circuit gates. This is illustrated in Figure 4 (3). Users may also sketch any other annotations they like on a circuit design (as shown by the "text" and highlights in Figure 4 (4)). These strokes are simply represented as an unrecognised shape in our data structure but are saved and loaded and may be moved or deleted as can the recognised circuit elements. The user can move and delete gates via direct manipulation with the pen on the electronic whiteboard surface. Moving is done by selecting one or more gates using the pen to specify a bounding box then moving the gates by clicking within the bounding box and dragging the pen to indicate relative amount to offset the gates as illustrated in Figure 4 (4).

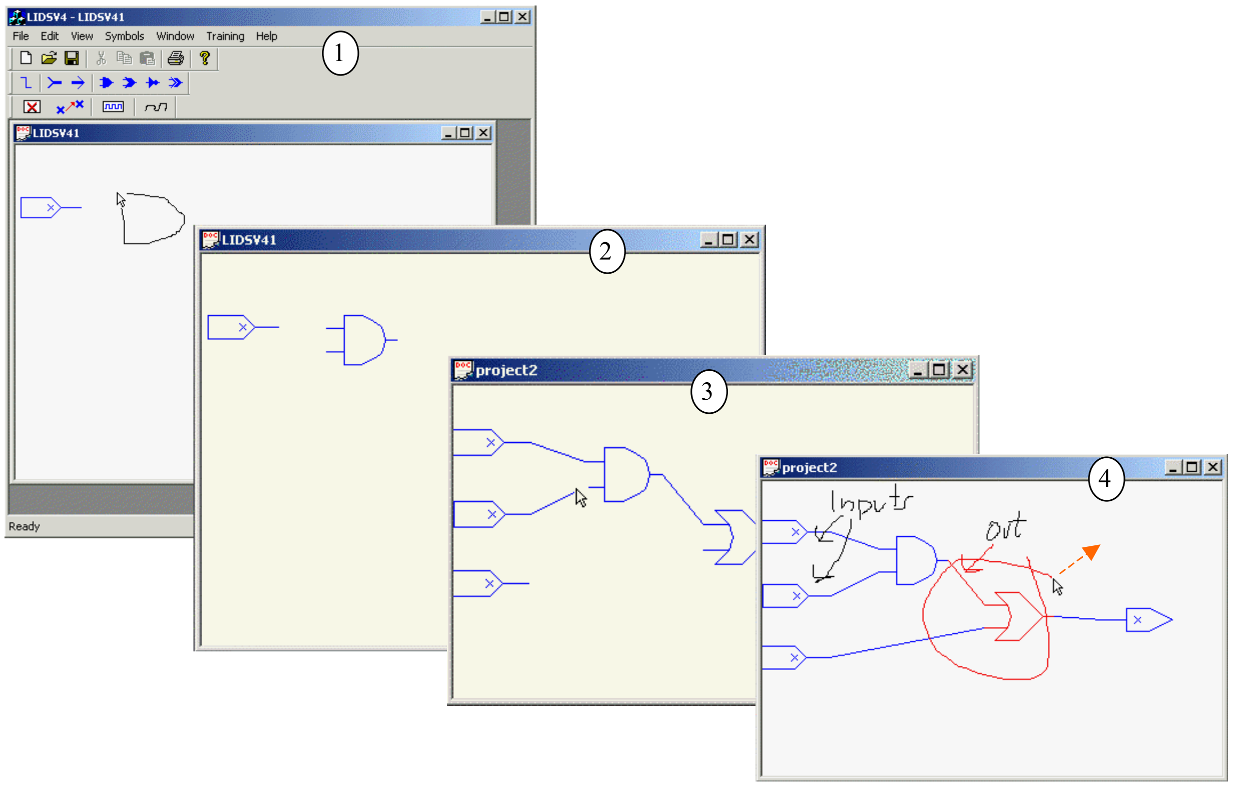

Figure 4. Manipulating a circuit design with pen-based input/large screen display output.

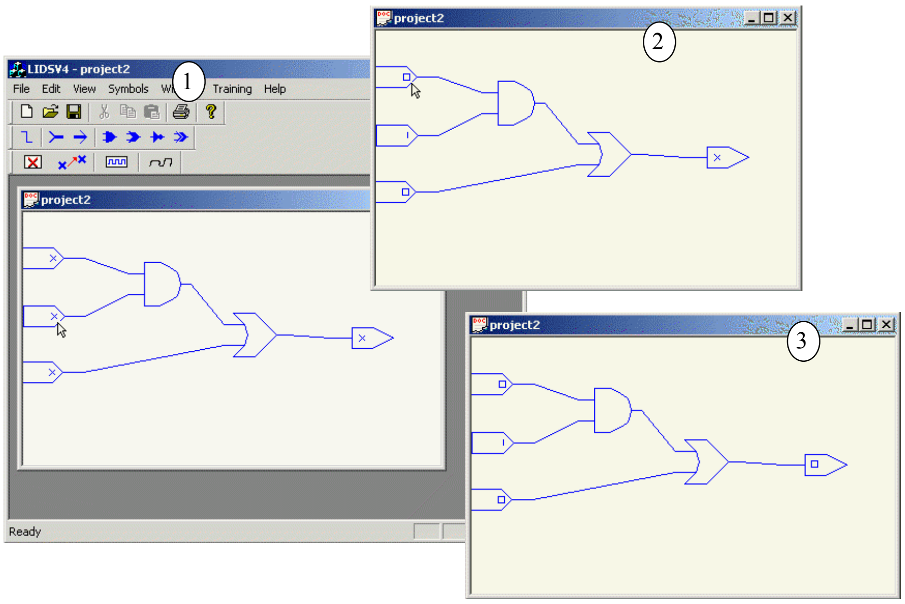

Figure 5. Animating a circuit design using pen/large-screen display.

At any time a circuit design can be "run" (animated) by a user to demonstrate its functionality. To do this, the user specifies on (0) or off (1) values for input gates via toggling these values when clicking on the gates, as shown in Figure 5 (1 and 2). This sets the initial state for the input gates to the circuit.

Once all input gates have specified initial value, the user then tells the circuit designer to propagate these signals along connector wires to gate pins. When an element receives signals for all its input pins its state is set (0 or 1) and shown to the user. On the next user-initiated step, the element propagates its value to its output pin(s). The result is illustrated in Figure 5 (3) where a circuit design is animated by the user to propagate a value to an output gate.

The propagation algorithm could be made incremental, and the state of gates illustrated in the views with colour and/or state information for gate inputs and outputs. This would allow more interactive and iterative visualisation of circuit design state. It would also be possible to leave the state visualisations in place while the user manipulated the circuit design, enabling them to visualise the changing circuit results as they changed the circuit design structure.

4. Implementation

We implemented our prototype circuit designer using the C++ programming language and Microsoft Foundation Classes GUI toolkit. The window opened by our application is back-projected onto the electronic whiteboard surface to show digitally captured circuit design content and to support user interaction with buttons and menus. Input from the mimio device is captured and sent as serial signals to the PC running the circuit designer application. The standard MimioMouse application is used to generate Windows mouse movement events and left mouse button click events from a mimio pen being moved and tapped on the electronic whiteboard surface.

We implemented Rubine's algorithm for single-stoke recognition [9]. A training interface is provided to allow different users to specify example sketches for shape classes to configure the recognition algorithm. This is illustrated in Figure 6. Circuit designs and annotations are currently saved and loaded as text files.

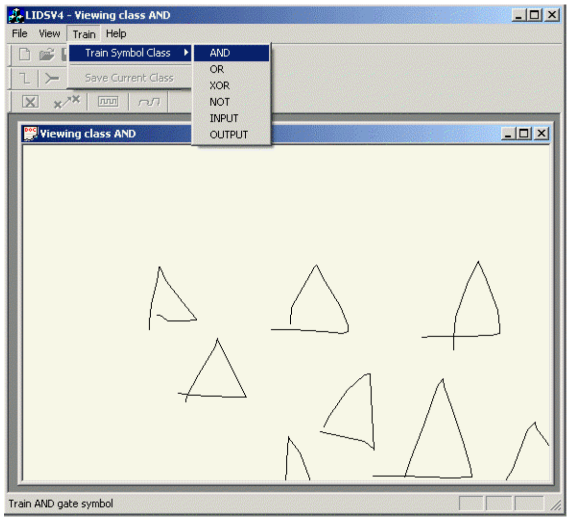

Figure 6. Shape recogniser training interface.

The architecture of our prototype tool is shown in Figure 7. MFC C++ libraries are used to render the user interface and accept pen input. The shape trainer is used to process pen input to determine sketched shape, process move/delete commands etc, and this is used to update the circuit design. The circuit data structure uses the MFC libraries to draw the circuit which is projected onto the E-whiteboard. Circuit designs are currently saved as ASCII text, which we are planning to extend to XML and also use to broadcast circuit changes between multiple remote-located designers.

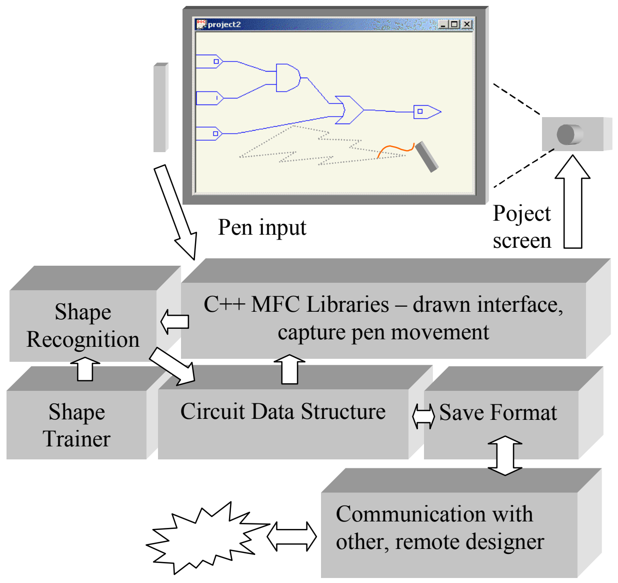
project2
Pen input
Poject screen
Shape Recognition
C++ MFC Libraries – drawn interface, capture pen movement
Shape Trainer
Circuit Data Structure
Save Format
Communication with other, remote designer

Figure 7. Prototype tool architecture.

5. Discussion

We have carried out an evaluation of our E-whiteboard circuit designer prototype by having users sketch designs in single and small group settings. The former tested basic interaction with the designer via the pen-based input and LIDS-based output technologies. The second investigated the use of the whiteboard during a simulated "tutorial" situation where one user would sketch a design and animate it, explaining the process to others. Reaction to the E-whiteboard designer was generally very positive and was expected to be a useful tool for both tutorial-based instruction, providing same-place collaboration on simple circuit design. However a number of interesting user interface design issues have presented during the development and experimentation with our prototype. These are summarised below.

The lack of a right-mouse button simulation with mimio pens led us to choose a number of interaction techniques for manipulating circuit designs with the pen device where a single pen tap operation only is used. For example, tapping on shapes gives a pop-up menu (normally a right-mouse button would be used to do this) to select element type or delete a single shape. A model move and delete facility is currently used to allow created circuit element manipulation. The user selects the manipulation mode and then selects the circuit elements to apply the operation to. Users of our prototype have suggested investigating alternative ways of moving, deleting and also resizing shapes. These have included tap-move-release of the pen for single-shape moves (the most common), use of a contextual pop-up menu and use of highlights or handles on drawn circuit elements to access particular manipulation functions. A context-dependent pop-up menu in the drawing area is currently difficult to provide for non-model operation selection as no right-mouse button vs left-mouse button click simulation exists to say when the user wants selection vs pop-up.

We chose to support shape sketching and recognition to provide an "E-whiteboard" sketch-based look and feel for our prototype. Hand-drawn circuit elements are automatically converted to computer-drawn form. It would also be possible to allow drag-and-drop of these computer drawn elements from a tool palette. However, we could also preserve the hand-drawn circuit element sketches and show their human-sketched form instead of computer-drawn elements. This would allow us to preserve a more sketch-based appearance to circuit designs, including allowing much greater flexibility for differently-sized and oriented circuit elements. Similarly, computer-drawn pin connector wires could be drawn instead of preserving user-sketched connections. Currently if user-drawn connection wires don't exactly connect pins the circuit designer then the tool moves them to directly connect the pins. If preserving hand-drawn wires then the tool would need to add a small computer-drawn line from the start pin and end pin centres to the user-drawn line to reflect pin connection accurately.

While some users have suggested that they would like the option of preserving hand-drawn circuit gates and connector wires, there are problems preserving these sketched elements when moving, resizing or deleting gates. Currently we can redraw connector wires when circuit gates are moved. With hand-drawn it is difficult to automatically "redraw" them to preserve their human-sketched appearance. An alternative would be to simply draw a straight line between connectors (though this doesn't work so well for loop-back connection wires!) but this would give the circuit appearance an odd look with sketched gates and some wires hand-drawn and some computer-drawn. Users have suggested an option to have hand-drawn circuit elements left as sketches or automatically converted as drawn into computer-drawn form, as done in Knight [3]. Similarly, a further option could allow the user to specify “formalising” of the circuit design from all-sketch to computer-drawn as in SILK [6].

Currently to over-ride a gate type the user either selects required type via a pop-up menu or deletes the gate and redraws it (having to reconnect the new gate’s pins). An alternative we plan to experiment with is to allow the user to draw over the top of an existing gate, replacing its sketch shape with the new one and re-running the shape recognition algorithm. Our shape recognition algorithm currently uses a single stroke based on a training set of shape classes to gate mappings specified by the users. This means circuit shapes can only be recognised from a single user stroke with the pen, making similarly-shaped objects difficult to draw without the use of abstract gestures. Our recognition algorithm could be enhanced to allow multiple, sequential strokes to be chained together and a composite gate type recognised from simple, multi-stroke parts, as done in the VisualBasic UI sketching tool [2].Text recognition could be added to convert some annotations to computer-recognised and rendered text.

Our circuit animation facility allows a teacher to show how a circuit works when signals are applied via input pins to the circuit. This uses a simple propagation algorithm which works well for circuits without feedback, but becomes hard to follow when outputs from a gate are feed back into the gate directly or indirectly. It also does not adequately show the signals being propagated along wires and we plan to colour or annotate wires and gates to indicate their state clearly.

We have not yet investigated the E-whiteboard circuit designer being used by multiple people simultaneously i.e. with multiple pen input, either co-located or remote-located. Co-located collaboration has been via turn-taking with a single input pen. We are building a basic UML CASE tool using the same interface technology and plan to explore both remote collaboration and same-place collaboration with this more sophisticated design environment.

6. Summary

We have developed a sketching-based circuit designer that utilises an electronic whiteboard incorporating a pen-based input device and a back-projected large image display surface. Users sketch circuit gates, connect gate pins and manipulate circuit designs using pen-based direct manipulation techniques. Multiple people can view and discuss the design in a tutorial situation and the design can be animated to indicate circuit functionality.

Further work on the prototype includes running more structured usability evaluations with an extended prototype with configurable circuit display and interaction mechanisms, to more accurately gauge potential end users’ preferred interaction techniques. Our preliminary investigation of the prototype’s usability suggests that these user preference options will be required allowing a range of complementary interaction and circuit display techniques to be supported. We are planning to use the prototype and electronic whiteboard hardware in some Year 2 Digital Electronics course tutorials to assess its effectiveness for interactive teaching purposes. We plan to add multiple views of complex circuits as well as investigate specification of complex integrated circuit components by elementary sub-circuits views. We are in the progress of implementing a more sophisticated UML CASE tool using this interface technology and will evaluate its performance for collaborative design tasks rather than tutorial-based instruction.

References

1. Apperley et al (2002): Lightweight capture of presentations for review, In *Proceedings of IHM-HCI*, Lille, France, ACM Press.
2. Apperly, M. and Plimmer, B. (2001) “Computer-Aided Sketching to Capture Preliminary Design.” The Third Australasian User Interfaces Conference (AUIC2002), Australian Computer Society Inc., Melbourne, Australia.
3. Damm, C.H., Hansen, K.M. and Thomsen, M. (2000): Tools support for co-operative object-oriented design: Gesture based modelling on an electronic whiteboard, Proceedings of CHI’2000, ACM Press, pp. 518-525.
4. Gross, M. and Do, E.Y-L. (1996): Amiguous intentions: a paper-like interface for creative design, Proceedings of UIST’1996, Seattle, WA., ACM Press, pp. 183-192.
5. Landay, J.A. and Myers, B.A. (1995): Interactive sketching for the early stages of user interface design, Proceedings of CHI’95, ACM Press, pp. 43-50.
6. Landay, J.A. and Myers, B.A. (2001): Sketching Interfaces: Toward More Human Interface Design. IEEE Computer, March 2001, IEEE CS Press.
7. Lin, J., Newman, M.W., Hong, J.I. and Landay, J. A. (2000): Denim: Finding a tighter fit between tools and practice for web design, Proceedings of CHI’2000, ACM Press, pp. 510-517.
8. Myers, B.A. (1997): The Amulet Environment: New Models for Effective User Interface Software Development, *IEEE Transactions on Software Engineering*, vol. 23, no. 6, 347-365, June 1997.
9. Rubine, D. (1991): Specifying Gesture by Examples, Computer Graphics, 25(4), pp. 329-337.
10. Smart Technologies Inc. (2002): *SMART Board*, www.smarttech.com.