A Collaboration System and a Graphical Interface
for the MIT Microelectronics WebLab

by

Yifung Lin

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 24, 2002

Copyright 2002 Yifung Lin. All rights reserved.

The author hereby grants to M.I.T. permission to reproduce and
distribute publicly paper and electronic copies of this thesis
and to grant others the right to do so.

Author

Department of Electrical Engineering and Computer Science

May 24, 2002

Certified by

Jesus del Alamo

Thesis Supervisor

Accepted by

Arthur C. Smith
Chairman, Department Committee on Graduate Theses
A Collaboration System and a Graphical Interface for the MIT Microelectronics WebLab

by

Yifung Lin

Submitted to the
Department of Electrical Engineering and Computer Science

May 24, 2002

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

WebLab, is an online laboratory for microelectronics device characterization that allows students to carry out microelectronics device characterization through the web. Currently, Weblab lacks the interaction and collaboration available in an actual physical laboratory. To bring the educational value of collaboration to the system, we have built a deployable WebLab Collaboration System with a redesigned, intuitive user interface and a full set of collaboration features. The system uses a dual domain interface for individual and team work, and a token scheme to manage team changes. We ran experiments with our new system to test the effectiveness of collaboration in student work and learning and solicit user feedback. Our results suggest that collaboration enhances students’ abilities to produce quality work and learn concepts. The user feedback shows a significant improvement on the overall user experience of our new system versus the current WebLab system. We anticipate that the WebLab Collaboration System will be deployed to the web in the summer of 2002.

Thesis Supervisor: Jesus del Alamo
Title: Professor of Electrical Engineering
# Table of Contents

1. INTRODUCTION ...................................................................................................... 4  
   1.1 Overview ............................................................................................................. 4  
   1.2 Related Work ...................................................................................................... 6  
   1.3 The WebLab Collaboration System .................................................................... 8  
   1.4 Outline of This Thesis ......................................................................................... 

2. THE CURRENT WEBLAB SYSTEM .................................................................. 10  
   2.1 The WebLab Components .................................................................................. 10  
   2.2 The Hardware of WebLab .................................................................................. 11  
   2.3 The Software of WebLab ................................................................................... 12  
   2.4 The JS Prototype ............................................................................................... 15  
   2.5 Chapter Summary ............................................................................................... 17  

3. DESIGN GOALS FOR THE WEBLAB COLLABORATION SYSTEM .......... 18  
   3.1 Goals for the WebLab Collaboration System .................................................. 18  
   3.2 Difficulties ......................................................................................................... 20  
   3.3 Chapter Summary ............................................................................................... 22  

4. THE WEBLAB COLLABORATION SYSTEM ............................................... 23  
   4.1 The WCS User Interface .................................................................................... 23  
   4.2 The WebLab Collaboration System Architecture ............................................. 29  
   4.3 Chapter Summary ............................................................................................... 37  

5. COLLABORATION EXPERIMENTS AND DEPLOYMENT .................... 38  
   5.1 Experiment Method ............................................................................................. 38  
   5.2 Experiment Results ............................................................................................. 39  
   5.3 Results Analysis .................................................................................................. 42  
   5.4 Student Questionnaire ....................................................................................... 46  
   5.5 Deployment ......................................................................................................... 47  
   5.6 Chapter Summary ............................................................................................... 48  

6. CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK .................. 49  

APPENDIX A. COLLABORATION COMMANDS .................................................... 52  
APPENDIX B. TWO PART COLLABORATION EXPERIMENT EXERCISE .... 54  
APPENDIX C. WCS USER FEEDBACK QUESTIONNAIRE ............................. 58  
BIBLIOGRAPHY ........................................................................................................ 61
Chapter 1

Introduction

1.1 Overview

Real, hands-on experience is important when it comes to learning. Although students can learn from textbooks or from lectures, their understanding is often enhanced through tangible experimentation with the material. Not only do students learn more when they get to perform tests, they are also more interested and find the learning to be more fun. This is true in several fields of engineering. Unfortunately, laboratories in engineering education are very expensive and students do not usually get many opportunities to learn through actual experimentation. Recognizing the lack of laboratories and the importance of hands-on experience, the I-Lab project was launched by ICampus, an MIT-Microsoft initiative to enhance learning through the use of technology. I-Lab’s mission is to explore the technology and pedagogy of online laboratories, real laboratory experiments that can be accessed through the web. I-Lab is deploying a series of remote, web enabled laboratories where students can “carry out experiments from anywhere at any time.”¹ The project’s motto is, “If you can’t come to the lab... the lab will come to you!”¹

WebLab, an online laboratory for microelectronics device characterization, is the most advanced project among the I-Lab projects. The system, originally architected by

¹ i-lab.mit.edu
Lane Brooks and Christopher McLean\textsuperscript{2}, enables students to conduct real-time measurements on microelectronic devices through a Java applet. Any Java enabled web browser can access WebLab and connect to the hardware devices behind the server. This provides an efficient, cost effective way for students to perform microelectronics laboratory experiments. The students do not need to worry about a lack of devices, facilities, strict laboratory hours, or transportation. And, the university can provide hundreds of students with the laboratory experience using just one set of instruments and devices. WebLab is very successful. 175 MIT students\textsuperscript{3} used it to perform experiments in the fall semester of 2001 alone. In addition, 30 students from Singapore\textsuperscript{2} also connected to the remote laboratory as part of their studies in the Advanced Materials program of the Singapore-MIT Alliance.

Unfortunately, WebLab suffers from a several drawbacks when it is compared to a real laboratory. Students in real laboratories do not perform their experiments in isolation. There are fellow students working on lab assignments and instructors helping students. The interactive environment allows students to learn better through group exchange and assistance. It also gives them a live and more interesting place to work. WebLab, on the other hand, can only communicate to remote students individually through its interface. Students working on WebLab experience no interaction with others. This retracts from the educational experience. Furthermore, the WebLab interface is made up of a series of text input boxes which have little resemblance to actual instruments and devices. Such an interface is unintuitive and makes it difficult for


students to learn device characterization. WebLab’s shortcomings are serious barriers to bringing the actual laboratory experience online.

1.2 Related Work

To improve WebLab’s interaction, Victor Chang added collaboration in his Java Servlet prototype\(^4\), hereby referred to as the JS prototype. The JS prototype has functionality that allows users to share input values and results. It was built on the old WebLab interface, and presented as an initial step towards collaboration. The JS prototype consists of a Java servlet running on the WebLab server. When a user connects, the user’s Java applet makes a persistent socket connection to the Java servlet. Then, the servlet and the applet start a dialog to enable collaboration. The servlet provides the applet with communication to other clients and the applet provides the servlet with the user’s actions. Although the JS prototype was never formally released, Chang had a chance to carry out educational studies on it in a contained environment.

Chang had a number of suggestions after the JS prototype which we note here:

- **Team management** – A group management system is necessary for deployment. One large team would make it difficult for the users to manage, and it would be a barrier to their learning and experimenting. Only one user can make changes to the team’s test bench at any one time so in a large team, there might be too many people waiting to share experiments and results. Users need the ability to form a variety of teams on the fly.

---

- A simple user interface – The users of the JS prototype needed to toggle between five WebLab frames: an individual HP4155 instrument control frame, an individual graph frame, a group HP4155 instrument control frame, a group graph frame, and a connected users list and one additional third party chat frame. Having to switch between so many frames made it difficult for the user to navigate, and often the bottom frames got lost beneath the ones on top. Furthermore, multiple frames make it harder for the user to handle all the information in his Java applet since he cannot see everything at one time.

- Embedded chat – Chat functionality should be built into the system so that users can communicate in real time. The JS prototype borrowed chat functionality from a third party chat program which is not a feasible approach in a deployed system for general purpose use. Third party chat programs require all of the users to own the third party chat software and know each others’ contact information ahead of time. In addition, not all third party chat programs are supported across multiple platforms. An embedded chat feature would allow users to communicate easily with anyone using the system.

- Enhanced collaboration – Although we cannot provide face-to-face communication, we can automate the communication process to make it seamless for the user. In the JS prototype, the token holder needed to explicitly upload the work to the other users for them to see any change in the system. Not only does the token holder need to go through the trouble of sending the updates, the other users in the team do not get any updates unless the token holder decides to do so.
• Testing – The system needs to be tested so that it is robust and scalable. The JS prototype only had a maximum of three users logged on at one time, and it did not need to consider users logging on and off because it was restarted for each experiment. However, a deployed system needs to be running at all times so it needs to be robust enough to handle large numbers of users logging on and off from various locations without crashing or confusion.

1.3 The WebLab Collaboration System

By building upon Chang’s work in this thesis we have developed what we call the WebLab Collaboration System, hereby referred to as the WCS. The WCS follows the JS prototype architecture and uses a servlet with persistent socket connections to the clients. The WCS provides synchronous communication, and it is available for any internet connection that is not behind a firewall. Unfortunately the only form of communication through a firewall is HTML, which does not provide a persistent connection.

To make up for not having the actual instruments and devices, we\textsuperscript{5} have remodeled WebLab to look like a typical circuit as opposed to the old WebLab interface consisted of a series text input boxes. This new interface gives the user a more intuitive environment for their experiments. Not only is the new interface easier to use, it also makes it easier for the user to picture and understand the devices. Additionally, the new interface is also more concise so it does not take up as much valuable screen space as the old interface.

For the interactive environment, we have embedded collaboration into the WebLab system. The collaboration includes functionality for users to form teams, chat

\textsuperscript{5} Victor Chang and Yifung Lin under the supervision of Professor Jesus del Alamo
among teams, and share experiments and results synchronously. The WCS automatically transfers inputs and results to team members automatically when the users make changes to the team circuit or the graph. The collaboration features make WebLab more lively, and we feel that they make a significant step towards bringing the rich laboratory experience online.

1.4 Outline of This Thesis

In Chapter 2, we briefly describe the WebLab system before outlining the design goals of the WCS in Chapter 3. Then, Chapter 4 provides a detailed description of the WCS, and Chapter 5 contains the steps we took to deploy the WCS and includes user studies and feedback. Finally, we conclude what we have learned and suggest future work in Chapter 6.
Chapter 2

The Current WebLab System

In order to articulate our new developments with the WCS, we must first examine the WebLab System as it stands today. This chapter presents a brief summary of how the current WebLab System Version 4.2 (deployed June 2001) works.

2.1 The WebLab Components

Figure 1: The Architecture of the WebLab

Figure 1 shows the architecture of the WebLab system. The front end of WebLab consists of a Java applet and a series of ASP pages. Clients log in to the front end by
connecting to the ASP pages through the HTTP internet protocol from their Java enabled browsers. Then, they can launch the applet to perform their experiments. The backend of WebLab consists of a server running Windows 2000 Server and a set of hardware instruments and devices that actually perform the experiments and measure the results. The server uses a GPIB card to connect to the hardware instruments and communicate to them using the GPIB protocol.

2.2 The Hardware of WebLab

WebLab can support up to eight Devices Under Test (DUT). These are typically semiconductor devices such as transistors or diodes. Microelectronics students are instructed to study the DUTs’ responses to a variety of current and voltage signals to assist their understanding of device operation. At the heart of WebLab is a Hewlett Packard HP4155B semiconductor parameter analyzer which sends signals to a DUT, measures the response, and graphs the results. A Hewlett Packard HPE5250A switching matrix routes signals from the HP4155B to the DUT selected by the remote client.

The HP4155B semiconductor parameter analyzer has a total of eight two way ports for connecting to a DUT: four SMU (Semiconductor Measuring Unit) channels, two VMU (Voltage Measuring Unit) channels, and two VSU (Voltage Source Unit) channels. The HPE5250A switching matrix allows the eight ports to be connected to any of the DUTs. Figure 2 shows a HPE5250A sitting above a HP4155B, and figure 3 shows a close up of the HP4155B’s data display screen.
2.3 The Software of WebLab

---

7 Operating the HP4155B. The Fachhochschule Münster University of Applied Sciences.
The WebLab site runs on Microsoft Windows 2000 and IIS (Internet Information Server). On the front end, the Java applet provides the WebLab functionality for the user. In the back end the WebLab driver and the VISA driver assist in communication with the hardware. For a better understanding of the software components, refer to figure 1 which shows the architecture of the WebLab system. WebLab is a database-backed web application built using ASP pages. Using the SQL Server database allows WebLab to manage users, log tests, and store test setups.

The communication in the WebLab system is a complex process. When a user runs a test using the Java applet, the job is first checked to see if it is executable by the hardware and then checked to verify that it will not cause damage to the device. The port information specified by the user is then sent to an ASP page on the server. Next, a VBScript on the ASP page opens a connection to the commercial VISA driver through the custom built WebLab driver. Once the connection is opened, the VBScript can send the port values from the user’s Java applet to the VISA driver. Then, the VISA driver uses GPIB commands to communicate to the HP4155B and the HPE5250A. The HP4155B and the HPE5250A support GPIB commands so they carry out the test and send the results back to the VISA driver in GPIB. Finally, the results go back through WebLab driver to the ASP page where the Java applet reads them. Once the results reach the Java applet, it produces a graph similar to that of the HP4155B’s graphing utility. This graph can be highly customizable by the user.

The transfer of the of the port values between the Java applet and the server is done through a query string made up of ampersand separated name, value pairs. An example of a query string would look like:
username=yifung&analyzer=hp4155&mode=SWEED&SMU1VName=VG&SMU1IName=IG...

The query string has all the required information to capture the entire test setup. This query string can be saved and reloaded from the database by each user.

The user interface to WebLab consists of two frames: a HP4155 frame and a graph frame. The HP4155 frame is made up of a menu and several text input boxes. The user uses the input boxes to set the values for SMUs, VSUs, VMUs, variable setup, and user defined functions. The menu allows the user to reset the applet, load inputs, save inputs, run measurements, load results, and switch devices. Figure 4 shows the HP4155 frame for the current WebLab system. The interface was designed to capture the functionality of the HP4155B. Although this design was done well, users were often confused by the non-intuitive array of inputs.

Figure 4: The HP4155 frame
One of the most impressive components of WebLab is the applet’s graph frame. The graph frame displays the results in a graph. Users can select the data to plot and scale the axis of the graph with the Axis Setups below the graph frame. The other buttons on the left side of the graph frame allow the user to see the data and download the data. In addition, the graph frame includes a convenient auto-scale features that centers the most interesting part of the data for the user. Figure 5 shows the graph frame.

2.4 The JS Prototype

The JS prototype was building using the current WebLab interface described above. Unfortunately, this interface is not suitable for collaboration, and it caused significant clutter and confusion in the JS prototype. The JS prototype required an
individual domain, a team domain, a connected members list, and a third party chat frame. Each of the domains consisted of a HP4155 frame and a graph frame. Thus, the JS prototype based on the current WebLab interface required a total of 6 frames, and it did not including the functionality for ad hoc teams. Figure 6 shows the WebLab components of the JS prototype. Students using the JS prototype also used a third party chat frame.

Figure 6: User interface of the JS Prototype (without third party chat frame)
The JS prototype is WebLab’s pioneer system for collaboration. With the JS prototype, Chang introduced the concepts of separate individual and group domains and the token scheme. Having separate individual and group domains allowed users to do their own individual experiments in addition to working with the group. This keeps all users engaged. Separate domains was well received by users. The token scheme kept group changes under control by allowing only one token holder per group to modify data in the group domain. The token scheme was also received well by the users.

The reason the JS prototype required six individual frames is because the HP4155 frame and the graph frames were large so they could not be combined into one frame. The two frames also needed to be doubled to have an individual and a group domain. The connected users frame was necessary for users to see who is online and pass the token, and the third party chat software was necessary for communication.

2.5 Chapter Summary

This chapter has provided a brief understanding of the WebLab system. It has described the architecture and components of WebLab and the JS prototype. In the next chapter, we will outline the goals of the WCS.
Chapter 3

Design Goals for the WebLab Collaboration System

WebLab is unique enough that it requires its own implementation for collaboration as opposed to using a general collaboration utility such as Microsoft NetMeeting. NetMeeting allows users to carry out work on a shared desktop. Although using NetMeeting could work for WebLab if all the users share the same Java applet, it is not an elegant or practical solution. NetMeeting would require that all the users know each other beforehand, have NetMeeting running on a Windows environment, a host IP address, and a predefined application for chatting. Furthermore, synchronizing the entire desktop, including the graphics, between the users is a slow process that requires a great deal of overhead. By designing and building our own implementation, we can produce a system that is efficient and easy to use.

This chapter lists the design goals for the WCS and then describes some of the difficulties of the design.

3.1 Goals for the WebLab Collaboration System

In addition to the considerations we learned from the JS prototype in chapter 1, we also designed the WCS based on several goals:

- Simplicity – The complexity of a system usually grows exponentially with each addition element. By keeping the system simple we make it easier to expand the

---

8 Only one computer behind a router has a host IP address.
functionality, find bugs in it, minimize the bugs in it, and teach others how it works.

- **Ease of use** – The WCS is designed for deployment and use so it should be as easy to use as possible. An easy to use system is essential to producing an overall rich user experience. Although the concept of making a system easy to use makes sense, it is usually difficult to do in practice. The designers have a hard time figuring out what the users expect and what works best for them. Companies in the software industry normally run usability tests and solicit user feedback. We ran educational experiments using the system and asked the participants to complete a usability questionnaire after the experiments.

- **Liveliness** – We wanted the WCS to bring as much of the actual laboratory as possible to the user. Making the system lively with respect to interaction amongst users was fundamentally important in achieving this goal. Although it is impossible to completely duplicate the live communication online, it is possible to show users evidence that the site is live. The best example of this is probably the popular online shopping site Amazon\(^9\). Amazon produces an amazing shopping experience, and one of the major reasons is because it is vividly live. Users see personal customer reviews of products, number of customers who found a review helpful, customer purchase recommendations, and information on similar purchases that customers who purchased a particular product also made. Amazon is used by millions of users, and it shows. Although the interaction on Amazon is not direct, face-to-face interaction, the communication on the online store could be arguably more useful than that of a normal store. Our aim was to apply this

\(^9\) www.amazon.com
principle towards the WCS to produce a rich laboratory experience by making it as live and interactive as possible.

- Deployment – Our goal in building the WCS was to make it available for users. Thus, the objective was to produce a robust system that we could deploy as opposed to traditional research where prototypes or test system are built to prove a particular concept.

3.2 Difficulties

We identified a number of notable difficulties during our design of the WCS. These barriers are worth examining since they shaped our system.

Our first difficulty was that there was too much information for the user to deal with. Giving the user both an individual domain and a group domain requires two test benches and two graph frames. On top of that, our goal to embed a chat and implement ad hoc teams required even more information that the JS prototype. We had a difficult time figuring out how to present all the information to the user in an elegant and simple way. Hence, we decided to construct a whole new graphical interface for WebLab.

Another difficulty we faced was that rebuilding the graphical interface and adding the collaboration functionality required the collaboration components of the system to be rewritten. This is because all of the collaboration commands for the JS prototype were generated and used in pieces of the old interface. The new interface was overhauled so little of the old collaboration functionality could be recycled. Furthermore, the implementation of the JS prototype assumed only one default team, and it was not generic enough to handle multiple teams. We had originality planned on extending the JS prototype to include the new functionality, but we soon realized that the collaboration
needed to be written from scratch. Perhaps it was better that we faced this difficulty.

Good software practice calls for engineers to build prototypes, learn from them, and then throw them away to replace them with completely new implementations.

We also faced slight difficulties in making the system new and available at the same time. We wanted to use the Swing package\(^\text{10}\) for our new system instead of the traditional Java Awt package\(^\text{11}\) since Swing is newer with more functionality. It would also provide our system with a fresh, new look, but we were not sure if the new Swing package would be widely used and supported on the internet. For instance, at the time when we were designing the WCS, Swing was not supported on the MIT Athena machines. The Athena consultants confirmed to us that the plug-in was available, but there was a bug in it. Fortunately, the bug in the plug-in was resolved on Athena by the time our educational experiments for the WCS took place.

Our last major difficulty was scalability and robustness which go hand in hand. We had no experience designing real-time collaboration systems suitable for deployment. Unfortunately, we also had little resources available for testing scalability. Our resources consisted of testing the system ourselves and hiring a handful of students to assist with the testing. We had no tools for load testing like our counterparts in the software industry. To fully test the scalability and robustness of the WCS, we would probably need to purchase a testing tool; however, such tools could be expensive and require time to configure. Fortunately, WebLab has a number of students that use it. Our best bet would most likely be to let the students do their assignments on the WCS as a trial application and provide them with a bug report forum where they can report problems.

\(^{10}\) http://java.sun.com/j2se/1.3/docs/api/javax/swing/package-summary.html

\(^{11}\) http://java.sun.com/products/jdk/1.2/docs/api/java/awt/package-summary.html
with the system. In case of serious system failures with the WCS, the students could use
the individual, non-collaborative versions of WebLab to complete their assignments.

3.3 Chapter Summary

In this chapter we discussed the goals we had for the WCS. These goals included
considerations we learned from the JS prototype as well as new goals we came up with.
We ended this chapter by discussing the significant difficulties we faced with our new
collaboration system. In the next chapter, we outline the architecture and implementation
of the WCS.
Chapter 4

The WebLab Collaboration System

This chapter presents a detailed description of the WCS. It begins with the user interface and functionality, and it is followed by the system architecture and components.

4.1 The WCS User Interface

Figure 7 shows the user interface of the WCS. Instead of having multiple independent frames like the JS prototype, the WCS uses several internal frames that are placed on an open source scrollable desktop designed and implemented by Tom Tessier. The entire desktop fits on a standard desktop resolution of 1280x1024. This is possible because the frames in the new interface are compact and significantly smaller than the ones in the old interface.

Where the old interface used a channel definition panel consisting of a series of text input boxes, the new interface uses an image of the device being tested. The advantage of the new device interface is that it is more intuitive and compact than the old channel definition panel. To configure the device, the user clicks on the ports to bring up input dialog boxes. Figure 8 shows a typical SMU input dialog. After inputting the information into the dialog, the port’s image changes to reflect the circuit element defined by the user. The tooltip also changes to display the new information when the user places the mouse over the port.

13 The new graphical interface was envisioned and designed by Victor Chang with the help of Yifung Lin under the supervision of Jesus del Alamo.
Figure 7: The WCS User Interface

Figure 8: SMU input dialog
To the right of the device, there are a series of new icon buttons. These buttons allow quick access to common tasks performed by the user. To figure out what a button does, the user can mouse over the button to read the tooltip or make sense of the intuitive icon on the button. Alternatively, the user can use the traditional menu bar above the device to perform tasks.

The graph frame has also been made more compact. The axis’ setups have been placed in a dialog that pops up when the user presses the setup button. The new, smaller graph frame allows it to be placed right next to the device in the same frame. This way, the user does not need to toggle back and forth between the device frame and the graph frame.

4.1.1 The Individual and Team Domains

The two frames on the left of the WCS interface are the test benches. The top test bench is in the user’s individual domain. Changes made to the individual domain are not propagated to fellow team members. The bottom test bench is in the team domain, and it is synchronized automatically within each team. The use of two domains worked well for the JS prototype so the WCS incorporated it as well. Having dual domains allows users to do their own experimenting in addition to working as a team.

Inside the menu of the team domain is a Transfer tab that allows the users to transfer all the information from the team domain to the individual domain and vice versa.
4.1.2 The Token Scheme

The information in the team domain is synchronized automatically within the team when changes are made in the team domain. Who gets to make the changes? The WCS uses a token scheme like the JS prototype. There is one token holder per team designated by an asterisk by his or her username in the team member frame. Only the token holder can make changes to the team domain. The token holder can pass the token to other users so that all users have a chance to share information. If a token holder leaves the team, the token is automatically passed to a random team member.

Users who do not hold the token will have portions of the team domain’s menu bar disabled and grayed out so that they cannot make changes to the team domain. These users cannot transfer the information from their individual domains to their team domains, and when they try to open an input dialog in the team domain, they get an error message informing them that they must be the token holder to make changes in the team domain.

Users who do hold the token have their entire team domain enabled. Each change the token holder makes is immediately propagated to the other team members so that all the information in the team is synchronized. The information in the team domain is also saved on the WebLab server so that new users joining the team can be synchronized with all the previous changes that happened before the user joined.

4.1.3 The Collaboration Frames

The collaboration frames are located on the right side of the WCS desktop. These frames provide ad hoc teams and chat functionality.
The first frame is the *available teams frame* shown in figure 9. This frame lists the currently available teams that the user can join and allows the user to create a new team. To join a team, the user can click the team name and then click join or simply double click the team name. To create a team, the user types a new team name in the text input box and presses enter. Then, a new team under that name is created and the user is automatically placed into it. The WCS also notifies all the other users in the system that the new team was created. Ad hoc teams created in this fashion are removed when there are no more team members left in the team.

![Available Teams Frame](image)

Figure 9: Available Teams Frame

The second frame is the *team member frame* shown in figure 10. The title bar of the *team member frame* displays the name of the team that the user is in. Inside the frame, there is a list of users currently in the user’s team with an asterisk next to the token holder’s user name. The token holder can use this frame to pass the token by clicking on the recipient’s user name and clicking the pass token button or simply double clicking the recipient’s user name.

The last frame is the *chat frame* shown in figure 11. This frame contains a scrollable text area where chat messages are posted and a text input box for the user to send messages to team members. The text area displays in chronological order all of the
team’s messages since the user joined. New messages are placed at the bottom of the text area in real time and the old messages are moved up. Messages in the text area also include team status messages such as users entering and leaving the team and token exchanges.

Figure 10: Team member frame

Figure 11: Chat frame
4.2 The WebLab Collaboration System Architecture

The architecture of the WCS is the same as the architecture of WebLab version 4.2 with the addition of a collaboration server running on the WebLab server. This collaboration server is a Java servlet implemented with the Java servlet package included in the Java 2 SDK, Enterprise Edition\(^{14}\). The communication between the clients and the collaboration server is done through persistent socket connections. The collaboration server must be running at all times for any form of collaboration to take place. Figure 12 shows the architecture of the WCS.

---

\(^{14}\) http://java.sun.com/j2ee/sdk_1.3/
The WCS servlet is made up of just a handful of objects. The heart of the servlet is the \textit{TestBenchCollab object} which extends the generic \textit{OReily DaemonHttpServlet} class. The \textit{TestBenchCollab object} starts threads for incoming clients, keeps track of teams and users, and has a number of methods for exchanging information in collaboration. The \textit{SocketConnection class} is the thread object spawned by the servlet to listen to the client's messages. The \textit{WebLabTeam object} is used for managing ad hoc teams, and the \textit{Member object} handles client information. Figure 13 shows an object model of the WCS servlet.

![Object Model of WCS Servlet](image)

\textbf{Figure 13:} The object model of the WCS servlet. The ratios are defined as follows: $\ast$: zero or more, $+$: one or more, $!$: one to one

\subsection{4.2.1 The Command Strings}

When a client starts the WCS, the Java applet running on the client's computer makes a request to connect to the collaboration server through a designated port. Upon receipt of this request, the collaboration servlet accepts the connection and spawns a new

socket connection thread that listens to and handles the client’s messages for the duration of his or her session. After the servlet accepts the connection, the client’s applet begins its own thread that listens to and handles messages from the servlet for the duration of the session. All of the WCS’ collaboration functionality is made possible through messages between the clients’ applets and the collaboration servlet through the socket connections.

The messages between the clients and the servlet are in the format of command strings. Each command string is in the format of a command name followed by the argument. The WCS uses a null character to separate the command name from its argument. The null character is an appropriate separator character because it is easily programmed, and users cannot type it into the chat frame. A separator users could type into the chat frame would cut off parts of chat messages where the user types in the separator. Appendix A shows a table of command strings from the client to the servlet and a table of command strings from the servlet to the client.

4.2.2 Logging on to the Collaboration Server

The first exchange of information that takes place between a client and the servlet is the log on sequence. First the client sends a JOIN_INIT command with the client’s username as the argument. This tells the servlet that the client is joining for the first time. Then, the servlet takes the username and saves it in the current socket connection thread which handles this particular client. Next, the servlet creates a new Member object representing the client and assigns it to the generic Lobby team. Afterwards, the members of the Lobby are notified by the servlet that the new user has joined. To synchronize the client’s available teams frame, the servlet sends the client an EXISTING_TEAMS command followed by a list of the current team names. Finally, the
servlet sends the client a JOIN command with the team Lobby as its arguments along with a CONNECTED_MEMBERS command with the members of the team Lobby as its arguments. This gives the client the necessary information to update its team member list to reflect the Lobby team and display the Lobby members. Finally, the server broadcasts a NEW_MEMBER \textit{username} command to the rest of the Lobby members so they are aware of the new user. Figure 14 shows the steps of a client logging on to the collaboration server in chronological order.

![Diagram showing steps for client log on in chronological order from top to bottom](image)

Figure 14: Steps for client log on in chronological order from top to bottom
The steps for switching a client from one team to another follow a similar process. The difference is the client’s JOIN_INIT username command becomes a JOIN team name command. This tells the server to remove the client from its current team and insert it into the new team instead of just assigning the client to the generic Lobby team. When the client is removed from the old team, a REMOVE_MEMBER username command is broadcasted to the rest of the team so that their member lists can be updated to display up to date information. Figure 15 shows the steps of a client joining a new team.

**Figure 15:** Steps for client joining a new team in chronological order from top to bottom
4.2.3 Synchronizing the Team Domain

Synchronizing the team domain amongst team members is done in several pieces. The test bench setups are synchronized using a setup string that specifies all the port variables. The token holder’s applet automatically sends an UPDATE_INPUTS new setup string command to the servlet when any change is made to the team setup. The servlet responds by sending the command to the rest of the members in the token holder’s team. Upon receipt of the command, the team members’ applets update their team domain with the new setup string. A similar process happens to the data when the token holder runs a job, autoscales the graph frame, or changes the axis setup in the team domain. The token holder’s applet sends the new information to the servlet, the servlet broadcasts the new information to the rest of the team members, and the team members’ applets update their team domain.

The synchronizing method above works for the normal case when the team members stay constant for the entire session, but there is a problem when a new user joins the team in the middle of the session. The new user must be synchronized with all the team updates that were made before he or she joined. To deal with this problem, the WebLabTeam object in the collaboration servlet stores an aggregate of update commands to maintain the team domain’s state at all times. The number of updates that need to be stored is a fixed size because only the most recent change needs to be stored. A maximum of six update commands are necessary to hold the team domain’s state: one command for each input, result, autoscale, x axis setup, y1 axis setup, and y2 axis setup.
4.2.4 Chat Messages

Chat messages are broadcast to team members in real time. When a user sends a chat message, the message is not immediately posted to his or her chat frame even though it could be since the message is already known. Instead, the message is sent to the servlet and then the servlet broadcasts the message to the team members using a MSG message command. Each team member posts the message to his or her chat frame when it is received. By posting the message for the sender in the same manner as the other members, the sender has a confirmation that the message reached the collaboration server, and everyone receives the message at roughly the same time. This is the behavior a sender would expect rather than having the message post immediately to his or her applet before it is even sent to the other users. Although new messages need to come back from the servlet before they are posted, the process usually happens very fast.

4.2.5 Maintaining Real Time User Information

The current team and token information relevant to each user is maintained on the collaboration server by the servlet. In the initial stages of the WCS, the user information was maintained on both the client’s applet and the servlet to reduce processing for the server from looking up user and team information. This method turned out to be inferior because there were too many points for failure. A failure in one client often caused the team information to become out of sync. For example, if one client sends the token to another client but there is an error delivering the token, then neither client thinks it has the token and the token is lost. However, if the token information is maintained on the server, an error in transmission would not lose the token because the server would still hold one designated token holder for the team.
Keeping the information on the server also makes it easier to identify an error. When an error in the servlet occurs, the error is logged for future reference. However, errors in clients could be difficult to identify and track down because it is sometimes impossible for a remote client to know that he or she is out of sync with the other user. Another problem with keeping the information on the client arises when the same user logs in twice. Both of the client applets would try to send information to the servlet and have it broadcast it to the other members. A server that did not maintain user and team information would not be able to tell that there is a duplicate user. The WCS does not allow a user to log in more than once. As soon as the user logs in at a new location, the old session is terminated.

The WCS servlet maintains team information using a Java hashtable and a Java vector. The hashtable provides a quick team lookup by returning the WebLabTeam object for a given team name. Although hashtables are efficient for looks ups, they are not efficient for listing values. The vector provides an efficient way to list the teams for new users. Members of a team are also stored using vectors.

The WCS accomplishes the goals set forth in chapter 3. The servlet is a simple component consisting of only four modular java classes, and the communication between the clients and the servlet is done through a protocol of simple commands. The graphical interface makes the system intuitive and easy to use. In addition, the embedded chat and automatically synchronized group domain makes the system lively since team members interact and see changes to the team domain simultaneously. The system addresses reliability because it accounts for users logging on and off multiple times from multiple locations and gracefully catches errors in command transmission.
4.3 Chapter Summary

The collaboration server is the heart of the WCS. It routes all of the collaboration information from one client to another. The server also manages ad hoc teams, team members, and the token holder. The next chapter describes the collaboration experiments we performed with the WCS.
Chapter 5

Collaboration Experiments and Deployment

The WCS, has been tested the system ourselves in small sessions consisting of three or four people. Several bugs showed up during the initial sessions. After each session, the new bugs were fixed and a subsequent session was scheduled. The number of bugs decreased each time, and we eventually ended up with a system robust enough for actual experimentation. This chapter describes collaboration experiments we performed, feedback we received, and the future deployment of the system.

5.1 Experiment Method

The goal of our experiments was to test the usability of the WCS as well the effectiveness of online collaboration for WebLab. To encourage the students to try their best, we designed a two-part incentive experiment. In the first part, the students were required to complete five exercises. These exercises were made to get them acquainted with the system and learn about the BJT device we set up. The second part was a one exercise quiz on the device to see how effective the students learned. The two-part exercise can be found in Appendix B.

In the first part of the test, we divided the students up into groups of individuals, teams, and tutored sessions. The students were told to learn as much as they could in anticipation of the quiz to be taken individually. Our incentive was to offer the students a flat rate of fifty dollars for the first part and five dollars for each point scored on the ten-
point quiz. At the end of our experiments we had the students fill out a short questionnaire to give us feedback on the system.

We felt that our two part experiment was an effective way to test collaboration and improved upon the experiment performed by Chang using the JS prototype\textsuperscript{16}. Having the individual quiz at the end allows us to see if the students actually learn better after using collaboration. If there was only one collaboration exercise, the entire team might do well based on one individual in the team doing all the work while the others do not contribute or learn. Using the quiz provides an indicator of whether or not the entire team learned.

A total of fifteen MIT microelectronics students participated in our collaboration experiments. For the first part of the experiment, the students were divided up into three teams of three, four tutored students, and two individuals. The students in the teams were asked to use only the collaborative features of the WCS to communicate to each other and turn in a collective assignment. The tutored students were individually tutored on the exercises through the WCS by a teaching assistant. The individual students worked alone using the new interface without any collaboration. We had originally anticipated in having four individuals but two of our students unfortunately dropped out at the last minute. Similarly, one of our teams could not perform the experiment so we were left with three teams instead of four.

5.2 Experiment Results

For both parts of the experiment, we recorded the times and the scores. The averages and standard deviations for each of the groups were calculated but not graphed.

Since the sample size was small, we decided to graph the actual data points. Figure 16 is a graph of the score results on part one (maximum is 50 points), and figure 17 is a graph of the time results of part one. Figures 18 and 19 are the score (maximum is 10 points) and time graphs for part two respectively. The team and tutor labels for the part two graphs represent individual efforts from students who were grouped in teams or tutorials in part one.

Figure 16: Scores on part one between individuals, teams, and tutor sessions

Figure 17: Times to complete part one for individuals, teams, and tutor sessions
For part one, the average score was lowest for the individuals and highest for the tutor sessions. For part two the average score was rather similar regardless of grouping. The teams completed part one the quickest and the individuals completed part two the quickest. In both parts, the tutored students spent significantly more time completing the exercises.
5.3 Results Analysis

Unfortunately, we were unable to gather a large enough sample size to draw conclusions we are confident about. In particular, the scores for part two had a great standard deviation, and part one only had two individuals so most of the standard deviations for the individuals group were very high. Although we cannot draw confident conclusions, our results are enough for us to speculate how effective collaboration was.

The results from part one alone suggest that collaboration leads to better work. Specifically, the tutored students turned in nearly perfect assignments and significantly outperformed the teams and individuals. The teams came in second and outperformed the individuals by roughly the same amount the tutored students outperformed the teams. The results makes sense since individual students have no help while teams of students can work together to agree on the best solution. A team of students should be able to collectively turn in an assignment that is at least as high as the highest scoring assignment they would turn in individually. In this sense, a tutor session can be though of as a team where one of the students knows all of the answers. This is not exactly the case because the teaching assistant cannot give the solutions to the student, but he or she can make suggestions, tell the student things to watch out for, and say if the solution is correct or not.

The time taken to complete part one is interesting. The teams were the quickest while the tutoring sessions took the longest. We hypothesize that the tutored students take longer than the individuals because there is extra collaboration involved. The tutored students ask questions and the teaching assistant makes suggestions and points out mistakes that the student would not have realized alone. Thus, students who are
tutored take longer times to do the assignment but produce more accurate answers. Why do the teams finish before the individuals? One might think that team work would be similar to tutored work and also take longer than individual work. In the WCS experiments, however, the teams finished part one the quickest because they divided up the five exercises amongst the team members. When a team member finished an exercise, the rest of the team simply glanced to see that it looked correct. Working in this fashion, the teams were able to finish part one faster than individuals and tutored students. The conversation below is a dialog take from one of the teams working on part one. Their user names for the purpose of the experiment were One, Two, and Three.

Three (12:12): so which setups have you guys done already?
One (12:13): None. Which ones have you done? I can start doing the others.
Two (12:13): I was doing the first one. Don't have a graph though
Three (12:13): I'll start with number 5
One (12:14): I'll do number 2
Three (12:15): how do you change the connections?
Two (12:15): click on SMU1 and then select either V, I, or COMM on the right
One (12:16): Actually, I'm working on number four first.

As we can see, the students divided up the individual exercises. This particular team did that first before doing anything else. Afterward, they asked each other questions about WebLab, using the WCS, and figuring out the assignment.

One (12:17): Hmmmm... how do I do a floating connection?
Three (12:17): can you just not specify anything for that smu?
One (12:18): I'll try that.
Two (12:20): has anyone ran measurements yet? I get an 'error setting download variables' message
One (12:22): Click on the download checkbox in the SMU window for the variables you want to measure.
One (12:22): For IV characteristics, we want Vin vs Iout, right?
Two (12:23): yeah, I think so.
Two (12:23): can anyone change the team setup?
One (12:23): Yeah. I have the "token" right now. I can pass it if you want.
Two (12:23): Don't need to pass it, maybe just show a setup that works to help us out
One (12:24): I don't have one that works yet myself...
Two (12:24): I clicked download on the SMU window but get an error. Did you change anything in User defined functions?
One (12:24): Not yet.
One (12:24): What error?
Two (12:25): Error setting download variables when I try to run. Then I put the varialbes in user defined functions and I got something
Two (12:26): What's compliance?
One (12:26): Hmm... Let me check out the user defined function. Did you specify that the voltage supply should sweep from -2 to 2 v?
One (12:26): Dunno.
Two (12:27): yeah, under the SMU window I said -2 to 2 with .1 steps
One (12:27): Did you download V from that window and I from the user defined functions?
Two (12:28): no, I had them both downloaded from SMU
One (12:29): Did you download V from SMU1 and I from SMU2?
Two (12:29): for the first setup I had V and I on SMU1
One (12:30): Hmm... let me try that.
One (12:31): I can run the measurements just fine.
One (12:31): The results just don't appear to be in the window they told us to look in.
Two (12:32): ok, where did you download the variables from?
Three (12:33): #5 is done

After roughly an hour, the team makes last minute checks and finishes all of the exercise. They gather their solutions and turn them in.

One (01:07): Cool. When you get done, come over by Yi Fung and we'll turn all the graphs in.
Two (01:07): can one of you check #4 for me, i am getting the same for 4 and 1
One (01:07): Sure. I'll do number four and put it in the group window. Can you pass me the token?
One (01:08): They do look like they should be the same, but I'll do four just to check.
One (01:09): Is that what you got?
Two (01:09): yeah, that's what i got first. can you print that out? i had the wrong scale
One (01:09): Sure. I'll print it.
Two (01:10): thanks

The results from part two suggest that collaboration not only leads to better work, it may also lead to better learning. However, the minute differences in the average scores also suggest that the heart of the learning is based on each individual. This seems to make sense. Students can enhance their learning with collaboration, but the extent of the learning still depends on their own abilities. The team and tutored students from part one took considerably more time to complete part two than the individuals. Perhaps this is because the collaboration students learned more things to watch out for from part one so they took more time on the quiz to insure that everything was correct.
An observation of time versus score, shows that, there is some correspondence in the amount of time spent and the score received on the exercises. The more time the students worked on the exercises, the better their scores were. This is probably true since the team and tutorial students could get more help using collaboration with more time.

Figure 20 shows a graph of time versus score on part one.

Figure 20: Graph of time versus score on part one

Figure 21: Graph of time versus score on part two
For part two, there does not seem to be any resemblance between time and score. This is probably because there was no help so if the student could not figure it out spending more time would not help.

**5.4 Student Questionnaire**

At the end of the experiments, the students using the WCS collaboration features were given a survey so they could give us feedback on the system. We were interested in their overall experience with the system as well as any suggestions they had. The questionnaire (see Appendix C) consisted of four questions answered on a scale of 1-7, with 7 being the highest and a question asking if collaboration was helpful and why. The scale based questions addressed the friendliness of the user interface, liveliness of the WCS compared to an actual laboratory, ease of communication, and the overall experience of the new system compared to the current WebLab Version 4.2. Figure 22 shows a graph of the average and standard deviation of the student responses to the scale based questions. The horizontal dash marks the average and the vertical stripe shows one standard deviation above and below the average.

The results show that the students were mostly satisfied with the WCS and considered it a significant improvement to WebLab. When asked if collaboration was helpful and why, all but one of the students said it was helpful. The reasons they gave included being able to ask questions about the assignment, how to use the system, discuss the solution, and compare results. The tutored students found it particularly helpful having a teaching assistant there to help them. The single student who said that collaboration was not helpful said that he or she prefers to work alone.
The user suggestions included having more than one frame shared between team members, the functionality to see another team member’s individual test bench without going through the trouble of passing the token to that member and having him transfer his individual test bench to the team domain, and illustrated examples of how to use the system in the help section. Having more team frames would allow the team to work on more exercises at one time, but it would also add to the confusion. The ability to see another team member’s individual frame would assist in the communication of the team’s data. Illustrated examples would definitely make the WCS easier to use. At the time of our experiments, we had not upgraded the help functionality of WebLab to reflect the new interface.

5.5 Deployment

The collaboration server as it stands could be deployed with a few minor touch ups. To deploy the system, the java servlet must be running at all times. For the collaboration experiments, the servlet was served to the internet using Jakarta Tomcat 3.2
by Apache\textsuperscript{17}. Tomcat can also be used to deploy the system. In order to do so, the WebLab server must have Tomcat running at all times. Otherwise, the collaboration servlet would not be running, and none of the collaboration functionality would work. Tomcat can be scheduled to run at all times using the Windows schedule task manager. In addition, when Tomcat is first started, the servlet must be initialized by sending it an HTTP get request through a web browser. Afterwards, the collaboration server should be able to run on its own unless the servlet crashes or Tomcat fails. In either case, Tomcat needs to be shutdown and restarted and the servlet must be reinitialized.

Since the WCS is more complex and has more room for failure, the single user version of WebLab should also remain deployed. This gives users a lightweight alternative to use if they do not require collaboration or if the WCS is down. The collaboration experiments allowed us to test up to eleven users using the WCS at one time, but we have not had the resources to test more traffic. Ideally, the system should scale up to the necessary number of users, but there may be bugs involving large numbers of users that we are not aware of.

### 5.6 Chapter Summary

The results of running experiments with the WCS suggest that collaboration was helpful to students for working on exercises and learning the behavior of microelectronics devices. Although more data is necessary to confirm our results, we received positive user feedback from the system. The users were satisfied with the WCS and considered it a significant improvement over the current WebLab version 4.2.

\textsuperscript{17} http://jakarta.apache.org/tomcat/
Chapter 6

Conclusions and Suggestions for Future Work

In our efforts to bring the actual laboratory experience online, we built the WCS guided by the goals we outlined in chapter 3. Based on our experiments and experience with the WCS, we believe our system has addressed the goals in the following ways:

- Simplicity – The collaboration server is a simple Java servlet consisting of only four modular classes and well under 1000 lines of code. The servlet is relatively straightforward so it should be easy to maintain and improve. In addition all of the collaboration functionality is achieved through simple commands between the client and the collaboration server. The simplicity of the WCS makes it easy to track down bugs.

- Ease of use – We have redesigned the user interface to be a new intuitive graphical interface. The new interface was well received and considered a significant improvement over the current WebLab version 4.2 text input based interface. The graphical interfaces also allowed us to compact the multiple collaboration frames into one frame. This makes it easier for the user to navigate through the information.

- Liveliness – By having an embedded chat frame and automatic synchronization of the team data between team members, we have made the WCS a live and interactive system for users to use.
• Deployment - The system was designed to gracefully catch errors in command transmission and handle users logging on and off multiple times from multiple locations. We did not have the resources to test the robustness of the system under large loads, but the WCS performed well during our collaboration experiments.

Although we have made noteworthy progress with the WCS, there are still improvements that can be made to the system. The following additions would make a considerable improvement to the system:

• The user should be able to log on and off of the system at will and switch back and forth between the single user client and the collaborative client. Currently, users of the WCS are automatically logged in to the collaboration system, and the only way for them to lose the team test bench and other collaborative features is to close their Java applet and open the single user applet. The ability to toggle between the single user and multi-user variations in the same applet would make the WCS more flexible. In addition, when a client loses his connection due to an error in the network, the client’s applet should automatically try a few times to log the client back on. Currently, the clients that lose their connection must close their Java applets and restart them to log back on to the collaboration server.

• More tutoring features should be added to WebLab. Our experiments show that tutoring is an effective method for teaching students, but tutoring may not always be available. WebLab should have online tutoring forums where students can ask questions when a teaching assistant is not available. The forums would also allow
the students to learn from and answer questions other students asked. Other features could include email lists where students can email and discuss questions.

- The teams in the WCS could be improved to include long term teams and team management. Long term teams would allow students in classes to form teams for the remainder of the class. There could even be functionality for the students to save their team setups. Team management would provide the functionality for administrators of the teams to manage team members.

- A bug reporting system should be setup for WebLab so users can report bugs they find on the fly. This way, users are encouraged to report bugs and WebLab has a formal way to receive and respond to them.

The collaboration experiments show that there could be a positive effect for collaboration in student work and learning. For the experiment to be conclusive, more participants would need to be tested and perhaps a more suitable exercise that tested more concepts should be developed. We hope that our research will provide some insight in collaborative learning. Perhaps this could be an area for further study.

We built the WCS to introduce a system that could bring the collaboration aspects of an actual laboratory experience online, and we anticipate that the system will be deployed in the summer of 2002.
## Appendix A

### Collaboration Commands

#### A.1 Commands Sent from the Applet to the Collaboration Server

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG</td>
<td>Ask server to broadcast chat message to team members</td>
</tr>
<tr>
<td>UPDATE_INPUTS</td>
<td>Requests the server to update the inputs for the team members’ team domains</td>
</tr>
<tr>
<td>UPDATE_RESULTS</td>
<td>Requests the server to update the graph frame for the team members’ team domains</td>
</tr>
<tr>
<td>AUTOSCALE</td>
<td>Alert the server that the graph frame for the team members’ team domain should be auto scaled</td>
</tr>
<tr>
<td>SETSCALE</td>
<td>Instructs the server to update the scale of the graph frame for the team members’ team domain</td>
</tr>
<tr>
<td>JOIN_INIT</td>
<td>Initial join request. Asks the server to add the client and update the client’s available team frame and team member frame</td>
</tr>
<tr>
<td>JOIN</td>
<td>Request leave the current team and join a different team</td>
</tr>
<tr>
<td>NEW_TEAM</td>
<td>Instructs the server to start a new team and update all users’ available teams frame</td>
</tr>
<tr>
<td>PASS_TOKEN</td>
<td>Pass the token to the new designated user</td>
</tr>
<tr>
<td>LOGOFF</td>
<td>Log the client off of the collaboration system</td>
</tr>
</tbody>
</table>
### A.2 Commands Sent from the Collaboration Server to the Applet

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG</td>
<td>Post the message to the chat frame</td>
</tr>
<tr>
<td>UPDATE_INPUTS</td>
<td>Set the team domain test bench inputs to the ones specified</td>
</tr>
<tr>
<td>UPDATERESULTS</td>
<td>Set the team domain graph frame results to the ones specified</td>
</tr>
<tr>
<td>AUTOSCALE</td>
<td>Auto scale the team domain graph frame</td>
</tr>
<tr>
<td>SETSCALE</td>
<td>Update the scale of the team domain graph frame with the specified changes</td>
</tr>
<tr>
<td>ADD_MEMBER</td>
<td>Add the specified member to the team member frame</td>
</tr>
<tr>
<td>REMOVE_MEMBER</td>
<td>Remove the specified member from the team member frame</td>
</tr>
<tr>
<td>JOIN</td>
<td>Set the current team to the one specified</td>
</tr>
<tr>
<td>ADD_TEAM</td>
<td>Add the specified team name to the available teams frame</td>
</tr>
<tr>
<td>REMOVE_TEAM</td>
<td>Removes the specified team from the available teams frame</td>
</tr>
<tr>
<td>CONNECTED_MEMBERS</td>
<td>Set the members of the team member frame to the ones specified</td>
</tr>
<tr>
<td>TOKEN</td>
<td>Turn on modifications to the team domain if the user specified is the client, otherwise turn off</td>
</tr>
<tr>
<td>RESET</td>
<td>Reset the test bench inputs in the team domain</td>
</tr>
<tr>
<td>EXISTING_TEAMS</td>
<td>Set the teams of the available teams frame to the ones specified</td>
</tr>
<tr>
<td>ERROR</td>
<td>Alert the user that the specified error occurred</td>
</tr>
</tbody>
</table>
Appendix B

Two Part Collaboration Experiment Exercise
WebLab Collaboration Experiment

Characterization of an npn BJT in a diode-connected configuration

April 2002

This is an exercise designed to test the latest version of WebLab that is currently under development. This WebLab version includes two brand new features: a brand new graphical interface to specify the test vector, and a collaboration system that allows several students to work together on a lab assignment, even if they are in different locations.

This experiment has two parts. The first part is a lab assignment designed to be carried out according to the instructions given to you on-site by Yifung Lin and Victor Chang. The second part is a diagnosis exercise designed to evaluate the learning that took place in the first part.

1. Lab assignment

In this assignment, you will characterize the current-voltage characteristics of an npn bipolar junction transistor (BJT) in a diode-connected configuration. The terminal connection configuration of this device is available on line. The output of this exercise consists of screen dumps of WebLab Channel Definition panel and the corresponding WebLab Measurement Results panel.

There are as many as six possible configurations for an npn BJT to be connected as a diode. Five are shown in the figure below.

![Diode-Connected Configurations]

Your assignment consists of programming the WebLab Channel Definition panel to provide a direct measurement of the I-V characteristics of the diode-connected transistor in each of the five configurations shown above. The current and voltage should be defined exactly as in the diagram above (including signs). You may use the User Defined Functions to help you on this task. The Measurement Results panel should show a semilogarithmic graph of the I-V characteristics with \(-2 \leq V \leq 2\) V and \(10^{-12} \leq I \leq 10^{-1}\) A (I in y-axis in a logarithmic scale, V in x-axis in a linear scale.)
For each configuration, turn in a screen dump of the lab report available under the File menu. This exercise will be graded on a scale of 0 to 50 points with correct measurements for each configuration counting as 10 points. Points will be subtracted for every discrepancy with the above instructions.

Please proceed in order starting from configuration 1 and finishing with configuration 5 above.
2. Evaluation exercise (10’)

As an evaluation of how much you learned in the above lab assignment, now consider a 6th bipolar transistor connected in a diode configuration:

![Diode Configuration Diagram]

Program WebLab’s *Channel Definition* panel to carry out a direct measurement of the I-V characteristics of the above configuration. As in the exercise above, you may use the *User Defined Functions* to help you on this task. The *Measurement Results* panel should show a semilogarithmic graph of the I-V characteristics with \(-2 \text{ } V \leq \text{ } V \leq \text{ } 2 \text{ } V\) and \(10^{-12} \text{ A} \leq I \leq 10^{-1} \text{ A}\) (I in y-axis in a logarithmic scale, V in x-axis in a linear scale.) Turn in a screen dump of the *Channel Definition* panel and the *Measurements Results* panel.

This exercise will be graded on a scale of 0 to 10 points. Points will be subtracted for every discrepancy with the above instructions. This is a time limited exercise. You only have 10 minutes to carry it out.

*Important note:* For all measurements, hold the applied voltage between -2 and 2 V.
Appendix C
WCS User feedback Questionnaire

WebLab Collaboration System Survey

Dear Student:
Thank you for taking part in today’s WebLab Collaboration System experiment. Please take a moment to provide anonymous feedback for the future development of WebLab by filling out this form legibly.

User interface of the WebLab Collaboration System

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not user friendly. Tool difficult to use.</td>
<td>Somewhat easy to figure out how to use</td>
<td>Very user friendly. Easy to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Livelihood of the Collaboration System.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not lively. Tool seems dead.</td>
<td>Somewhat lively but nothing close to a real laboratory.</td>
<td>Very lively. Similar to that of an actual laboratory.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

How would you rate the communication of the WebLab Collaboration System versus face-to-face to face communication.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Difficult to communicate with team members</td>
<td>Somewhat easy to communicate with team members but not as effective</td>
<td>Just as simple as face-to-face communication</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No improvement or experience was worse.</td>
<td>Somewhat improved experience</td>
<td>Much improved experience. More fun to use</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Overall experience of new system compared to current system**

1-------------------2-------------------3-------------------4-------------------5-------------------6-------------------7

Was collaboration helpful for your learning (the microelectronic device or how to use the system)? Why?
Please suggest improvements to the WebLab Collaboration System that could make it more effective and help bring the actual laboratory experience online.
Bibliography


61