Introduction of Digital Experimentation Capabilities on the ELVIS iLab Platform

by

Hamidou Soumare

S.B., Massachusetts Institute of Technology (2008)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2009

Author . [illegible] . . . .
Department of Electrical Engineering and Computer Science
May 22, 2009

Certified by . [illegible] . . . .
Dr. V. Judson Harward
Associate Director CECI
Thesis Supervisor

Accepted by . . [illegible] . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Introduction of Digital Experimentation Capabilities on the ELVIS iLab Platform

by

Hamidou Soumare

Submitted to the Department of Electrical Engineering and Computer Science
on May 22, 2009, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

iLabs are online laboratories that give students access to experimental setups enabling them to conduct real experiments remotely through the internet. This circumvents a subset of the typical problems of conventional laboratories in addition to significantly increasing equipment utilization rates. In response to the lack of financial resources at partner universities in Africa, the National Instruments Educational Laboratory Virtual Instrumentation Suite (ELVIS), a low cost all-in-one electronics workbench, has become the hardware of choice for the iLab-Africa project. In this thesis, I extended the ELVIS iLab platform to support experiments in Digital Electronics complementing its analog capabilities. The new platform now offers an even greater return on investment by providing pedagogically useful laboratory exercises covering a larger portion of Electrical Engineering curricula.

Thesis Supervisor: Dr. V. Judson Harward
Title: Associate Director CECI

Acknowledgments

I would like to begin by thanking my thesis advisor, Dr. Judson Harward for providing me with invaluable guidance and direction throughout my tenure on the project. I would also like to thank Prof Jesus Del Alamo, who created the iLab project at MIT, for keeping me focused on the important vision of iLabs.

Many thanks to Jim Hardison who was instrumental in my development efforts. I also would like to acknowledge Phil Bailey, Kimberly DeLong, Meg Westlund, Maria Karatzas and the rest of the iLabs team for all their support and guidance throught the year.

My predecessors, Bryant Harrison and Adnaan Jiwaji, were great mentors during my transition period onto the project. The feedback I got from Rahul Shroff, my partner on the project, really helped shape the fnal product.

A great deal of thanks is due to the members of the iLabs teams at Makerere University and the University of Dar Es Salaam. I couldn't have asked for more hospitable and energetic hosts.

Lastly, I would like to thank my family for instilling in me the value of hard work and education. They have provided me with a great deal of love and support throughout my life for which I am eternally grateful.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation for iLabs

The ability to perform laboratory work is an essential component of any legitimate course in science, technology or engineering. The laboratory experience provides students with many of the skills necessary to become good engineers and scientists. The requisite skill set includes the aptitude to handle real data and understand the reasons for its discrepancies with theoretical calculations as well as the ability to work in teams. Most importantly, however, experimentation allows students to pursue their intellectual curiosity and develop their interests.

Unfortunately, many courses in schools around the world, including MIT, tend to forgo the laboratory component because of large class sizes in addition to perceived limitations in class time and equipment availability. This has very serious consequences for the students affected.

Traditional laboratories have their share of problems as well. The initial investment required to set up a lab is often a major barrier. When you factor in the cost of maintenance with lab assistants, repairs etc., the financial requirements are multiplied. Labs are also usually only open during regular business hours. This oftentimes represents an inconvenience for students as well as an underutilization of resources. From a professor's or lab administrator's point of view, the biggest risk is the damage to equipment by rough handling or extreme values for parameter inputs.

This risk was the initial motivation for iLabs. Jess del Alamo, a professor at MIT, had acquired the Agilent Dynamic Signal analyzer, which he wanted to make available to his students. However, given that the equipment cost upwards of $20,000 he could not risk damage to it. The concept of iLabs was conceived from this tradeoff and has had a profound impact on science and engineering education at every level.

iLabs are online laboratories that give students access to experimental setups enabling them to conduct real experiments remotely through the internet from anywhere around the world. This circumvents a subset of the typical problems of conventional laboratories and increases utilization significantly. Given that they can access the experimental setups 24 hours a day, 7 days a week, students are able to learn at their own convenience.

iLabs usage at MIT began in 1998, with the Microelectronics Device Characterization test station, which is still being used by over 500 MIT students in three different courses [6]. The concept of online laboratories is by no means tethered to Electrical Engineering; in fact iLabs have been developed in many other disciplines:

- Chemical Engineering (Heat Exchanger)
- Civil Engineering (Shake Table)
- Physics (Neutron Beam)
- Control Theory (HVAC Control)
- Mechanical Engineering (Pendulum)
- Chemistry (Polymer Crystallization)

Given the scalability of the iLabs platform, the defining characteristic has shifted from being a protective measure for costly equipment, to a vehicle for mass diffusion of pedagogically useful labs. iLabs lend themselves to the concept of sharing laboratories allowing students and researchers to use equipment located around the world. This may result in a global network of iLabs being available as people create and publish their own iLabs in addition to using iLabs put up by other institutions. There is

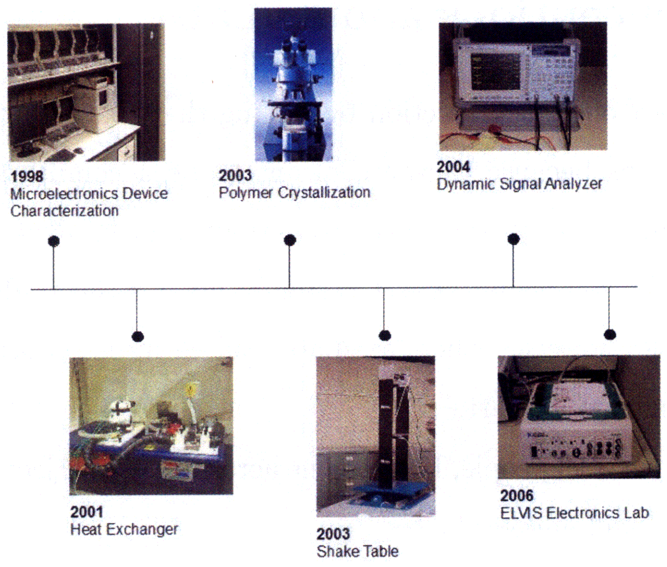


Figure 1-1: Timeline displaying select online laboratories. Not all of these labs were converted to the iLab Shared Architecture.

evidence to suggest that this is a sustainable idea given the fact that in 2005 alone, students in 10 different countries used iLabs. More iLabs come online every year.

1.2 iLab-Africa Partnership

iLabs offers the greatest potential benefit to students in developing countries and more specifically to students in African universities where the lack of sufficient financial resources, large class sizes, and bureaucracy have proven to be significant barriers to the procurement of lab equipment. Professors at these universities have shown significant interest in the iLab Project and as a result, MIT has set up a formal partnership with Obafemi Awolowo University in Nigeria, the University of Dar Es Salaam in Tanzania, and Makerere University in Uganda. This partnership began with the sharing of the MIT Microelectronics iLab, but has moved more and more towards encouraging the African universities to join the development process. This was supported by a transition from using high-end, ultra precise devices such as the Agilent DSA, to cheaper lab equipment such as the NI ELVIS workstation, which is more affordable for our African partners. This shift created a common development environment enabling us to work much more efficiently together.

1.3 Misconceptions about iLabs

I would like to start off this subsection by stating that iLabs will never totally supplant traditional labs because they are meant to be a complement rather than a complete replacement. Given the choice to perform a given experiment via an iLab or a traditional lab, you would almost certainly opt for the traditional lab. There are certain examples however, where an iLab resulted in better student performance than the traditional counterpart.

In my discussions with people, I have come across several objections to the concept of an iLab, which I will address below:

1. A major part of the experimental process is the process of wiring a circuit board, soldering and physically interacting with the equipment. This is impossible with an iLab, so iLabs have no value.

I agree that the ability to physically wire and solder is essential for an engineer. I contend however that once you have learned how to solder and wire a circuit on a breadboard, there is no incremental educational value in repeating the process for every circuit you analyze. Providing simple bread boards, solder and soldering irons to a class focusing on this mechanical process can be done at a relatively low cost. Most of the educational value of an experiment is in the scientific concept that is being demonstrated, which is just as clear through an iLab as it is with a traditional lab.

2. iLabs seem to be an expensive simulation, why don't you just use an industry-grade simulator such as SPICE?

This objection along with all of its variations came up extremely often. It is important to distinguish between simulation of the laboratory experience and simulation of some natural phenomenon. iLabs simulate the laboratory experience by extracting out the steps such as wiring that aren't critical for the concept being communicated. On the other hand, programs such as SPICE simulate the behavior of a physical entity such as a MOSFET for example. SPICE-style simulation is important for understanding the ideal behavior of an electrical device or system. iLabs are run using real devices on real equipment and so demonstrate all of the non-linearities, noise and other imperfections of real devices. This provides students with a better understanding of how a device behaves in the real-world along with the limitations of device itself and the mathematical models used to describe its behavior.

3. iLabs do not allow you to debug a faulty circuit, which is an important skill to have in practice.

Again, I completely agree with the importance of the ability to debug complex circuits. This skill is best learned through a traditional laboratory setup. Once this is learned, it is not necessary to debug every subsequent circuit analyzed. Although you cannot fully debug a circuit with an iLab, it can be simulated with the newest ELVIS iLabs. The most recent iLabs use clever switching mechanisms to allow students to dynamically take measurements across different points in a circuit, which is in effect the process of debugging.

1.4 Overview of Thesis

In this thesis, I will describe the motivations for extending the iLabs platform to include digital experiment capabilities as well as documenting the development process. Further, I will provide an update on the current state of the iLabs Africa partnership.

In chapter 2 I will provide a high level view of the iLabs shared architecture. All of the major components will be covered, namely the weblab client, service broker, and lab server as well as the information flow within the system.

In chapter 3, I discuss the National Instrument ELVIS hardware along with the LabView development environment. I will also discuss the characteristics that make ELVIS and LabView particularly suitable for deployment with iLabs in Africa.

Chapter 4 describes the iterative nature of our relationship with our African partners and how that has influenced the natural progression of the system. In this chapter I will describe past versions of the ELVIS iLab as well as discussing the needs that were met by each version. This discussion sets the stage for a discussion of the motivations and objectives of my work.

In Chapter 5, I provide a very technical discussion of the design and implementation of the changes required for the current version of the iLabs platform. Finally, in chapter 6, I make concluding remarks about the project as a whole. More specifically, I will discuss my contributions to the iLabs Africa project as well as providing recommendations for the future direction of iLabs Africa.

Chapter 2

iLabs Shared Architechture

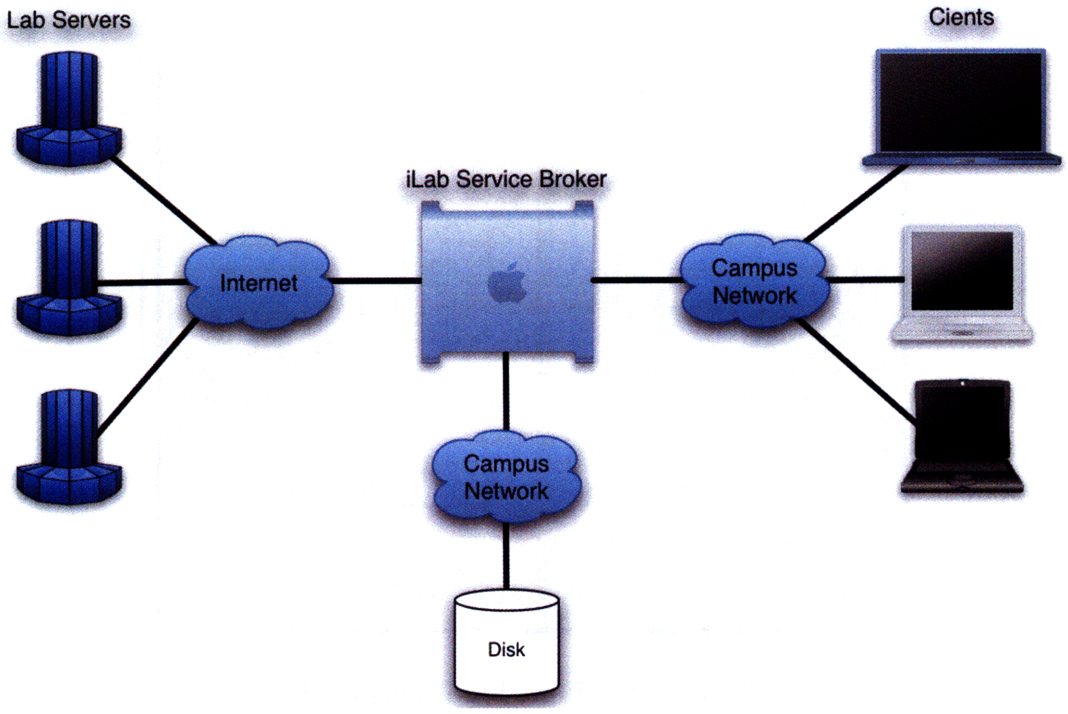


Figure 2-1: Diagram of the three major components in the ISA [11].

The iLabs shared architecture (ISA) is split up into three major subsystems. Each plays a distinct role. The lab server, service broker and lab client communicate via special xml documents and web service calls.

2.1 Lab Client

The lab client is the user interface through which students accesses an iLab. It provides an intuitive representation for the given lab being run and allows users to specify parameter values as well as graphing the experiment results returned from the setup. The current family of lab clients was adapted from the versatile, open source weblab client used in the original microelectronics lab. Thus, users of previous generations of lab clients will be familiar with the most recent lab client.

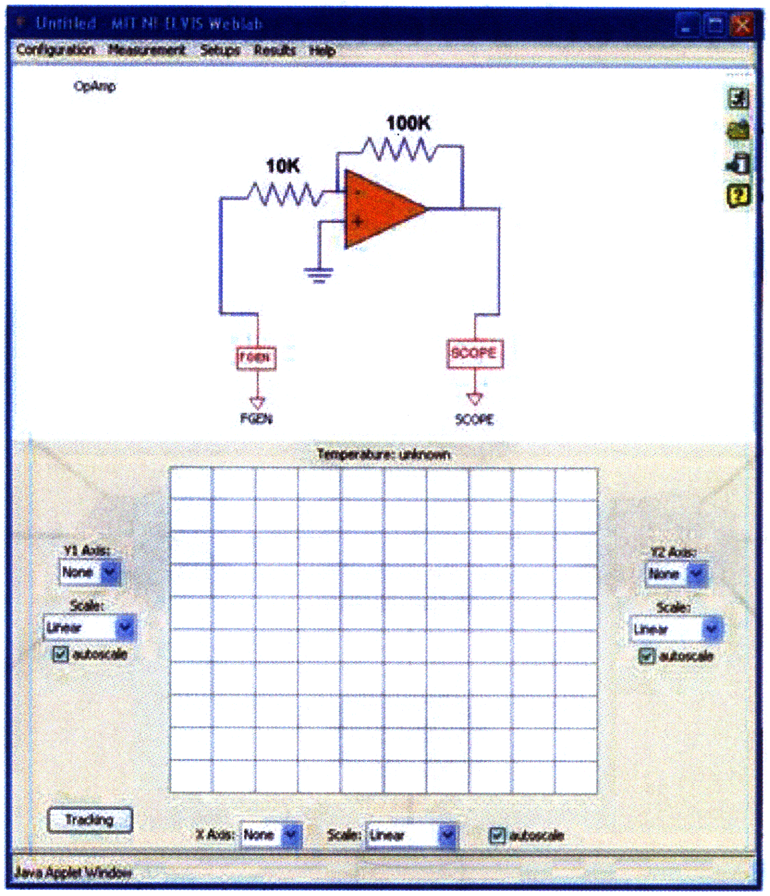


Figure 2-2: Screenshot of Weblab Client highlighting the schematic and results panel.

As can be seen in figure 2-2, the client is essentially made up of two components, the schematic panel and the result panel. The schematic panel illustrates the experimental circuit and allows users to specify parameter values for the instruments present in the setup. The results panel contains a 3 axis graph that allows a student to graph two experiment data vectors simultaneously. The results panel also contains several graphing utilities.

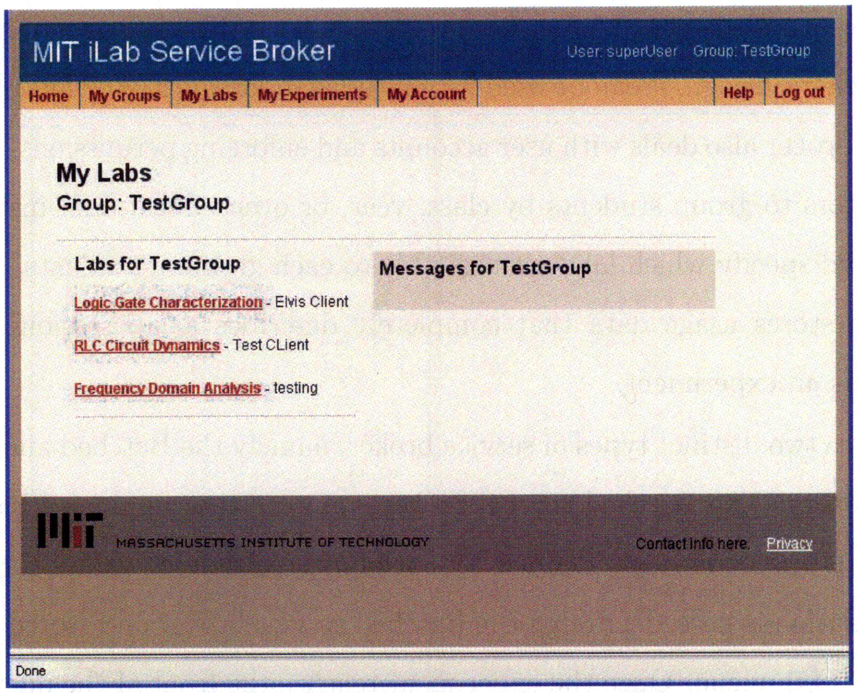
2.2 Service Broker

The Service Broker serves as the heart of the ISA by providing several administrative services. It also provides a path for communication between lab servers and weblab clients. All of the communication occurs by passing xml documents generated in the lab servers and clients. The service broker mediates this exchange of information. Given its generic design, it can be used by multiple lab servers and clients. In addition, the service broker also deals with user accounts and enforcing permissions. This allows administrators to group students by class, year, or even educational institution, for example, and specify which labs are available to each group of students. The service broker also stores usage data that completely describes a lab session whenever a student runs an experiment.

There are two distinct types of service broker, namely the Batched and Interactive Service Brokers. The Batched service broker supports the running of batched labs. A batched lab is completely defined by specifying parameter values for the setup. This information is passed through the Batched service broker and stored in a first-in first out (FIFO) queue. Once the experiment reaches the front of the queue, it is run and the results are passed back to the client. When running a batched experiment, the student does not have real time control of the lab equipment and as a result the bandwidth requirements are quite low. This type of experiment is suitable in many cases such as device characterizations.

On the other end of the spectrum is the Interactive Service broker, which facilitates the running of interactive labs. In an interactive lab, the student has complete and exclusive control of the experimental setup for a given period of time. This is a much more realistic simulation of the laboratory experience but presents several challenges such as the possibility of scheduling issues when dealing with large class sizes. This type of lab requires a significant amount of bandwidth, which is often restricted in African countries. As a result of these bandwidth issues, we decided to use the batched architecture for the ELVIS iLabs deployed in Africa.

We are in the process of deploying a Merged Service Broker, which provides a common platform for running both batched experiments and interactive experiments. Currently a large fiber optic backbone is being built to support the planned East African submarine cable. The successful completion of this project will amount to a significant increase in the available bandwidth. As a result of this future increase, we are encouraging our partners in Africa to adopt the merged architecture.



My Labs
Group: TestGroup

Labs for TestGroup

Logic Gate Characterization - Elvis Client

RLC Circuit Dynamics - Test CLient

Frequency Domain Analysis - testing

Messages for TestGroup

Figure 2-3: Screenshot of a Service Broker website displaying the labs available to the user.

2.3 Lab Server

The lab server communicates directly with the lab equipment. It uses the parameter values specified by the student to run the desired experiment and returns the experiment results. The typical lab server also provides an administrative interface with which a lab administrator can create a setup by specifying the required instruments. This administrative interface also allows for the specification of maximum and minimum supported parameter values preventing students from damaging the lab equipment with excessive parameter values.

The computer running the lab server software must be physically interfaced to the lab server hardware. The client and server processes are tightly coupled because their communication is essential to the system. Lab servers can be connected to several service brokers.

2.4 Information Flow through the ISA

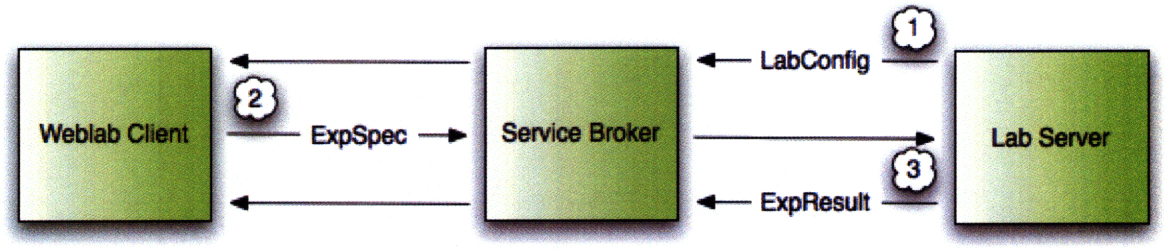


Figure 2-4: Diagram showing the origin and destination of XML documents in the ISA.

Figure 2-4 shows that communication within the ISA is facilitated by three distinct xml documents, *LabConfiguration.xml*, *ExperimentSpecification.xml*, and *ExperimentResult.xml*. The process occurs in three steps.

In the first step, a student accesses the service broker and selects a lab to run.The *LabConfiguration.xml* is created by the lab server and passed to the client via the service broker. This xml document provides information about the specific instruments that make up the setup and is used by the client to create a representation of the circuit in the schematic panel.

Next, the client creates the *ExperimentSpecification.xml* document containing all of the user specified parameter values. This is passed through the service broker to the lab server, which runs the specified experiment. In the final step, the lab server packages the results in the *ExperimentResult.xml* document, which is sent to and graphed in the client.

In the following chapter, I provide an overview of the specific laboratory hardware used in the ELVIS iLab and explore the reasons for its success with our African partners.

Chapter 3

Overview of National Instruments ELVIS

3.1 ELVIS Platform

3.1.1 ELVIS Hardware

The National Instruments Educational Laboratory Virtual Instrument Suite (ELVIS) is an all-in-one device containing a suite of 12 instruments allowing students to perform hands-on experiments in electronics. The ELVIS contains most of the essential instruments found in traditional electrical engineering laboratories. This combination of instruments allows for very complex experiments to be performed.

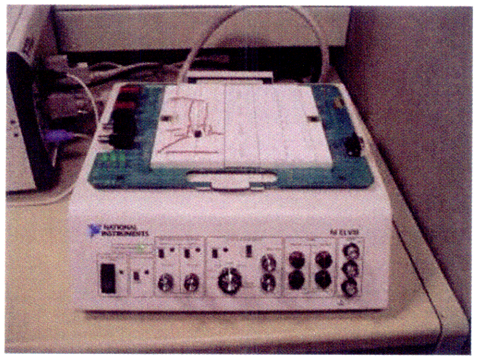


Figure 3-1: Image of NI ELVIS all-in-one desktop workstation.

3.1.2 ELVIS Software and LabVIEW

LabView is a very powerful graphical programming language developed by National Instruments. It takes a dataflow approach to programming making it very intuitive to use. LabView programs are referred to as Virtual Instruments(VI's). What is unique about the LabView programming language is that it ties the creation of user interfaces (Front Panel) into the development cycle. Controls and indicators within the front panel can be used to input data and extract information from a VI.

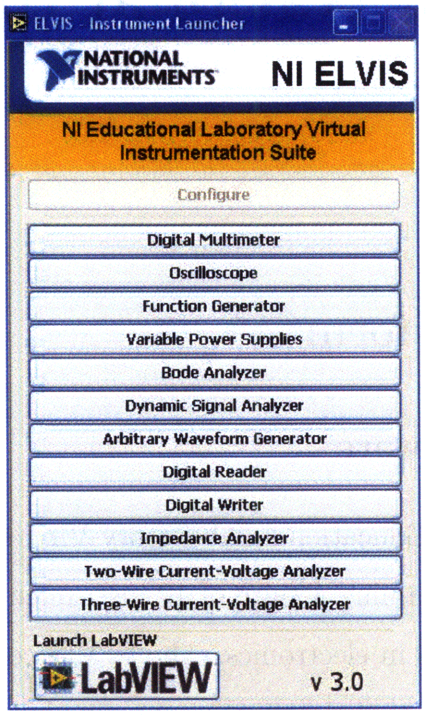
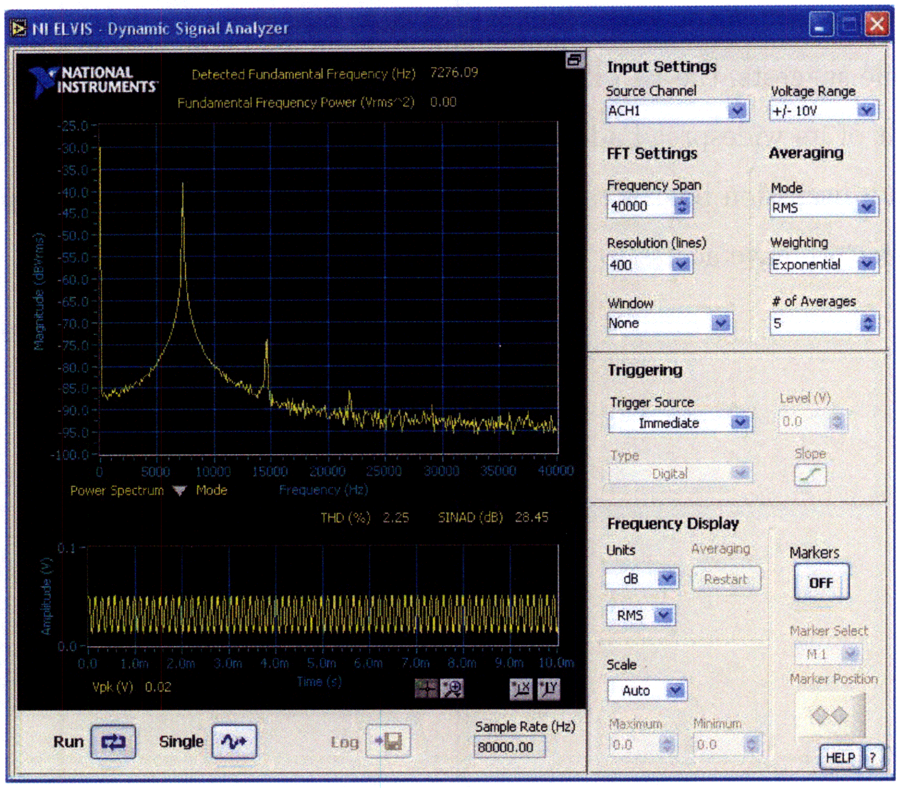


Figure 3-2: Screenshot of instrument launcher, which provides links to the user interface for all twelve ELVIS instruments.

The behavior of all NI ELVIS instruments is coded in LabView. Included with the hardware is software that provides a user interface offering complete control of each instrument (See Figure 3-3). LabView also comes equipped with Express VI's, which are a high level API that encompasses most of the common functionality for a given instrument. This dramatically reduces the development time since the Express VI's take care of general instrument tasks such as initialization and also prevent resource conflicts. Nonetheless, development on top of Express VI's requires the programming of low level VIs to tailor the Express VIs behavior to your exact specifications. Although building an instrument from low level VIs is much more time consuming, it does provide much more freedom in implementation. I chose to use Express VIs in my development efforts because their modularity allowed for implementation of the exact desired behavior. It also reduced the complexity of the design, resulting in more efficient and very readable code.


NI ELVIS - Dynamic Signal Analyzer

Detected Fundamental Frequency (Hz) 7276.09
Fundamental Frequency Power (Vrms^2) 0.00

THD (%) 2.25 SINAD (dB) 28.45

Vpk (V) 0.02

Sample Rate (Hz) 80000.00

Input Settings
Source Channel ACH1
Voltage Range +/- 10V

FFT Settings
Frequency Span 40000
Resolution (lines) 400
Window None

Averaging
Mode RMS
Weighting Exponential
# of Averages 5

Triggering
Trigger Source Immediate
Type Digital
Level (V) 0.0
Slope

Frequency Display
Units dB
RMS
Averaging Restart
Scale Auto
Maximum 0.0
Minimum 0.0

Markers OFF
Marker Select M1
Marker Position

Figure 3-3: Screenshot of User Interface for Dynamic Signal Analyzer instrument on ELVIS.

3.2 Suitability of ELVIS for iLabs in Africa

Reduction of Costs

The major theme in the discussions with our partners in Africa was cost. Their budgets simply did not allow for the acquisition of sufficient laboratory equipment to provide a meaningful experience for their students. The $2000 cost of the ELVIS platform is an order of magnitude less than it would cost to purchase all of the instruments separately. When you factor in the minimal maintenance costs and the number of students served, it is clearly an extremely cost effective solution.

Support for wide spectrum of Experiments

The mix of instruments provided by the ELVIS platform supports a wide range of experiments, which makes it applicable for students at many different levels.

Access to Extensive Knowledge Base

As its name suggests, the ELVIS platform was designed for educational purposes. As a result of its widespread adoption, National Instruments has set up several forums and discussion boards tailored directly to the ELVIS platform. Users of the ELVIS therefore have access to this extensive knowledge base.

Chapter 4

Past Development on ELVIS Platform and Motivations for ELVIS 3.0

It is important to note that the present state of the ELVIS iLab is the result of a very iterative process between developers and faculty at our partner universities. Each new piece of functionality was added in response to curricular demands and perceived limitations of the platform. In this chapter, I will provide context for each iteration of the ELVIS iLab as well as developing the motivation for my work.

4.1 ELVIS 1.0

ELVIS 1.0 was the first version of the ELVIS iLab. The reason for this version was to provide a low cost and accessible platform that would facilitate the development of labs by our African partners.

Samuel Gikandi, in his MEng thesis used code from the Microelectronics iLab as a starting point for his development efforts. The client was completely revamped to accurately display the results of experiments specific to the ELVIS as can be seen in Figure 2-2. This version of the ELVIS iLab only exposed the functionality of the function generator and oscilloscope, enabling the investigation of one monitor point

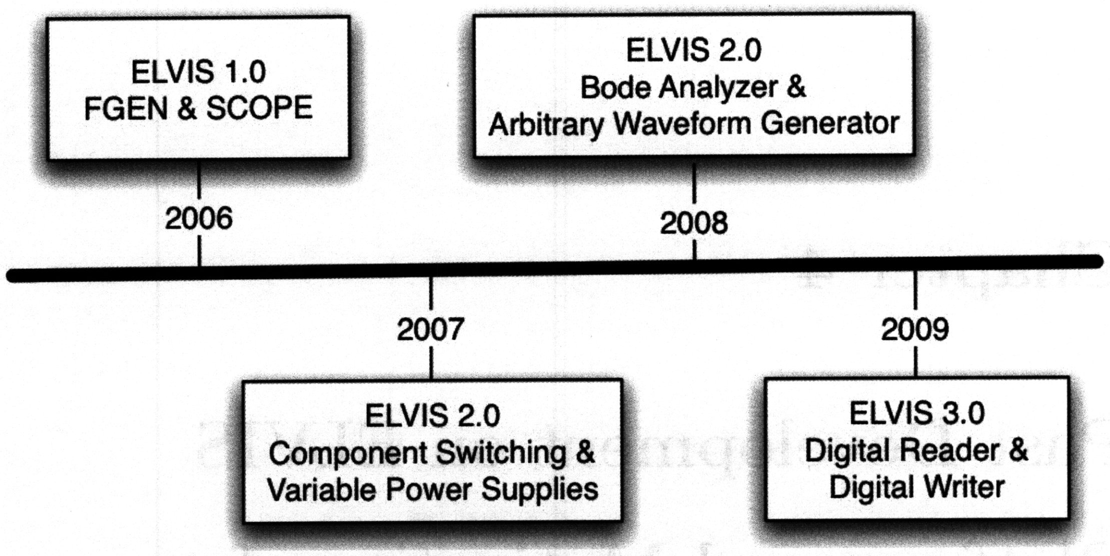


Figure 4-1: Timeline displaying the development of ELVIS iLabs.

in single input circuits. Given the low cost and potential for extending the capabilities, ELVIS 1.0 was very successful. There were, however, several limitations of the design.

Firstly, only being able to use the function generator and the oscilloscope greatly limited the types of circuits that could be analyzed. A multi-stage amplifier circuit requiring both 5V and 12V supply rails would be impossible to create on ELVIS 1.0, for example. Secondly, the points to be monitored in the test circuit had to be hardwired by the lab administrator and could not be altered remotely by the students. This limited the students' ability to fully investigate the circuit. Finally, only one experimental setup could be active at any given time, which created problems in the situation where different courses wanted to investigate different circuits on the same ELVIS station.

4.2 ELVIS 2.0

Version 2.0 of the ELVIS iLab addressed some of the limitations present in ELVIS 1.0 by exposing more of the ELVIS functionality, namely:

- Variable Power Supplies that output a DC voltage between -12V and +12V enabling Op-Amp experiments, among others.

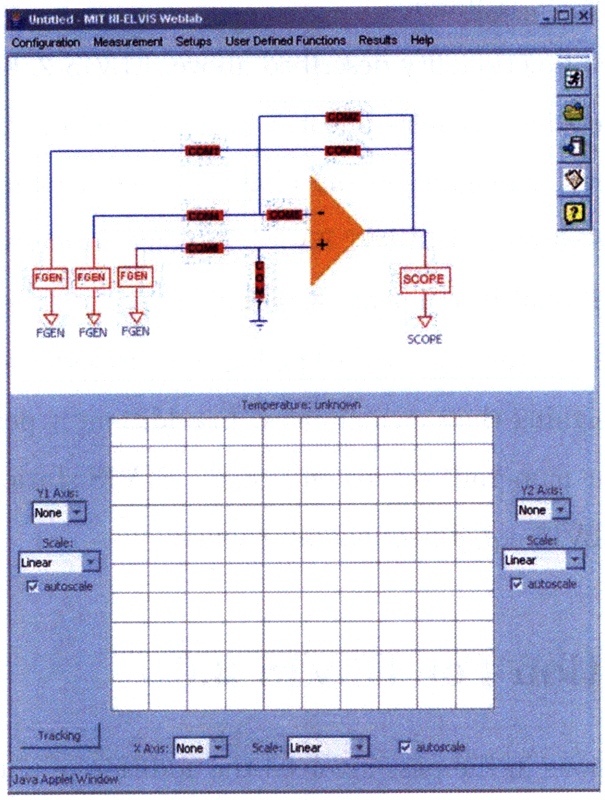
- Arbitrary Waveform Generator that allowed students to explore the respons of circuits to user-defined waveforms.

There were two major criticisms of the initial ELVIS iLab. First, the ability for students to design circuits was not supported. Second, students could only perform experiments in the Time-Domain, which neglected a significant portion of the material taught in the curriculum.

In addition to the functionality described above, ELVIS 2.0 incorporated support for the Bode Analyzer instrument, which exposed the frequency domain capabilities that allow for magnitude and phase response characterizations. Perhaps the most powerful addition in ELVIS 2.0 was the component switching capability provided through National Instrument switching hardware. The ability to switch components in a circuit provides a richer pedagogical experience since it allows students to start designing their own circuits. In addition, it permits lab administrators to put numerous setups on a single ELVIS station allowing different courses to use the ELVIS equipment simultaneously.

4.3 Motivation For ELVIS 3.0

The major contributions in the past involved the addition of a new "Domain" to the iLabs platform. ELVIS 1.0 set the foundation for the Time-Domain. ELVIS 2.0 introduced the Frequency Domain, and my work in ELVIS 3.0 was designed to contribute the Digital-Domain. The addition of the digital capabilities was the next logical step in the evolution of the iLabs platform. Furthermore, development of the digital capabilities will lay the foundation for a whole host of experiments involving logic gates, programmable arrays, ROMs, FPGAs, and various other microcontrollers. Good understanding of digital electronics is essential for engineers in this increasingly digital world.



Y1 Axis: None
Scale: Linear
☑ autoscale

Y2 Axis: None
Scale: Linear
☑ autoscale

X Axis: None Scale: Linear ☑ autoscale

Figure 4-2: **ELVIS 2.0 client displaying the representation of switchable components.**

Chapter 5

ELVIS 3.0 Detailed Design and Testing

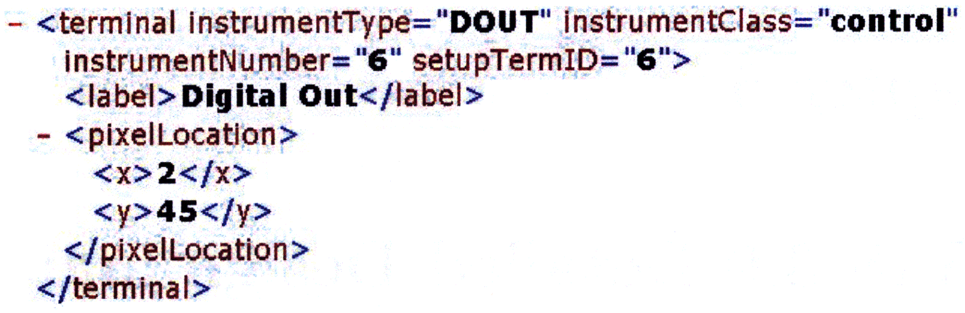
5.1 XML Document Specification

The development of the current version of the ELVIS iLab began with the XML documents since they dictate the format for the flow of information within the ISA as can be seen in Figure 2-4. Once these documents are specified, development can be done independently on the client and the lab server. Each XML document has a corresponding document type definition(.DTD) file, which is responsible for defining the legal building blocks of the XML document. It was only necessary to change *LabConfiguration.xml* and *ExperimentSpecification.xml* as well as the corresponding .DTD files.

5.1.1 Changes Made To *LabConfiguration.xml* And *LabConfiguration.dtd*

As discussed in Section 2.4, the *LabConfiguration.xml* (see Appendix A) is created by a process in the lab server that queries the databases for the instruments that make up the active setup. This document is used by the client to create the appropriate representation of the circuit. The only change made was to

LabConfiguration.dtd (see Appendix D) to allow for the new instrument type, namely the DOUT instrument. The allowed XML block for this instrument is as follows:



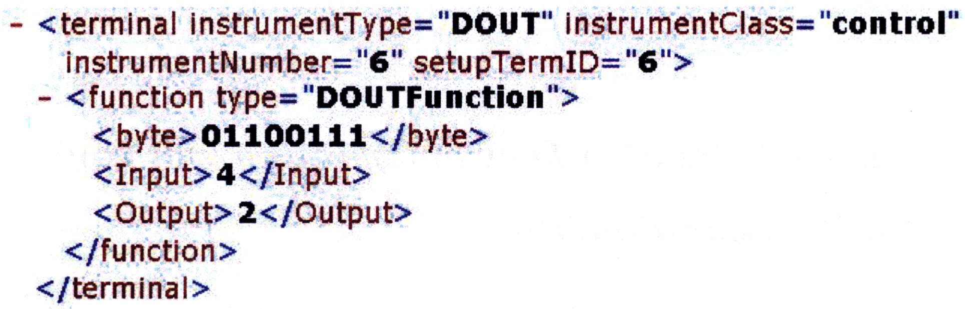
```
- <terminal instrumentType="DOUT" instrumentClass="control"
    instrumentNumber="6" setupTermID="6">
    <label>Digital Out</label>
  - <pixelLocation>
      <x>2</x>
      <y>45</y>
    </pixelLocation>
  </terminal>
```

Figure 5-1: Sample DOUT block in the Lab Configuration XML document.

This is a standard terminal definition. The label specfies the name of the instrument, while the pixel location specifies the instrument's location on the schematic panel of the client.

5.1.2 Changes Made To *ExperimentSpecification.xml* And *ExperimentSpecification.dtd*

The *ExperimentSpecification.xml* (see Appendix B) contains the user configured inputs for a particular instrument and is used by the lab server to actually run the experiment. The *ExperimentSpecification.dtd* (see Appendix E) was changed to support all of the inputs for the DOUT instrument.



```
- <terminal instrumentType="DOUT" instrumentClass="control"
    instrumentNumber="6" setupTermID="6">
  - <function type="DOUTFunction">
      <byte>01100111</byte>
      <Input>4</Input>
      <Output>2</Output>
    </function>
  </terminal>
```

Figure 5-2: Sample DOUT block in the Experiment Specification XML document.

The byte field specifies the sequence of bits to be written by the digital writer.

The input and output field are from the point of view of the circuit. The input field specifies the number of digital lines that are read by the circuit, whereas the output field specifies the number of digital lines being written by circuit.

5.2 Lab Server

The Lab server is the heart of the iLabs system and is made up of the following modular components:

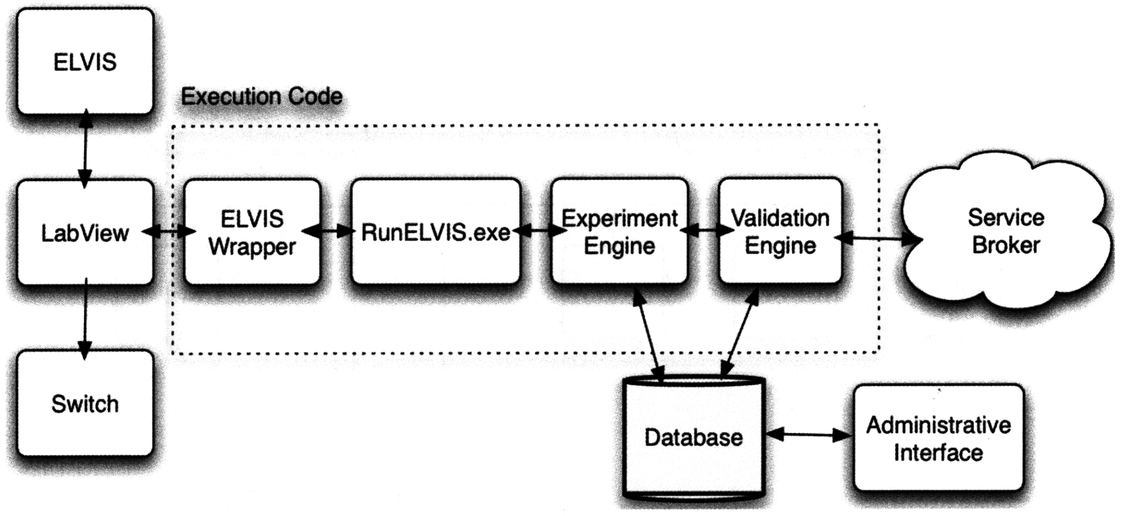


Figure 5-3: Flowchart depicting the interconnections between the lab server components.

- ELVIS and Switch hardware responsible for the actual running of the experiment.
- Execution code responsible for parsing and validating an experiment specification, passing parameter values to the hardware, and creating an XML document summarizing the empirical results.
- Database that stores data about available instruments and active setups in addition to all of the information for every experiment run.
- Administrative Interface used to create and activate setups.

5.2.1 LabView

The LabView code is made up of several modular subVI's that perform distinctive tasks.

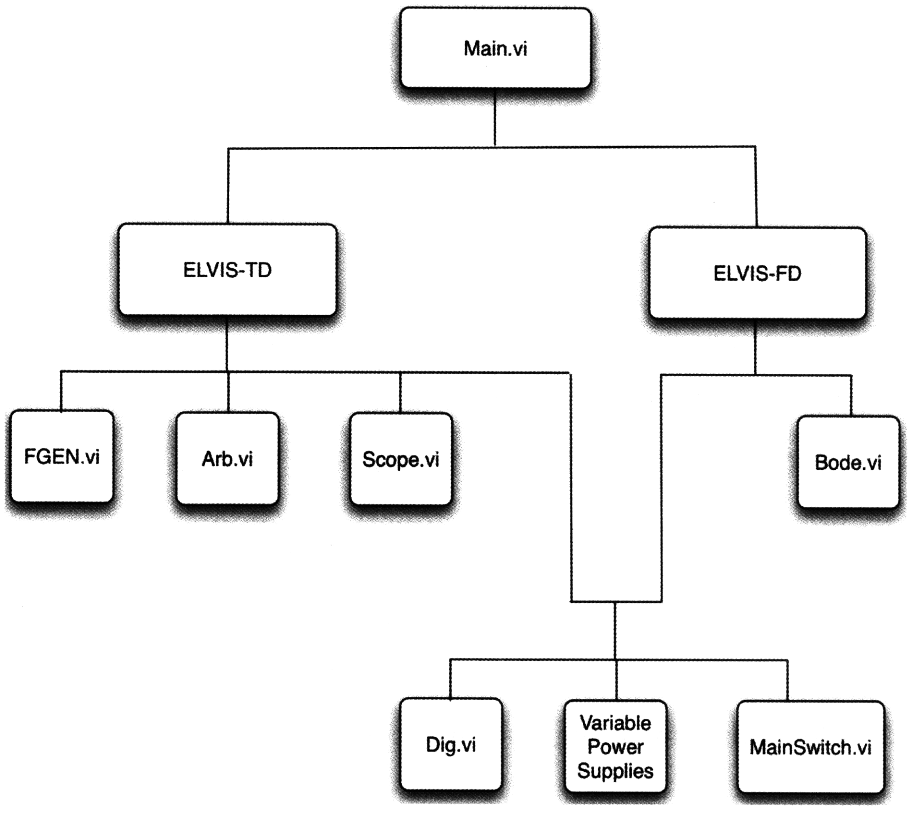


Figure 5-4: Diagram of interconnection between modular components that comprise the LabView code.

As can be seen from Figure 5-4 the LabView code is partitioned into a frequency domain tranche containing the Bode Analyzer and a time domain tranche, which contains functionality for the arbitrary waveform generator, oscilloscope, and function generator. In addition to this partition, there is a set of functionality that is common to both domains, namely the switch controller, the variable power supplies and the digital VI. The addition of the digital functionality was the major contribution in ELVIS 3.0.

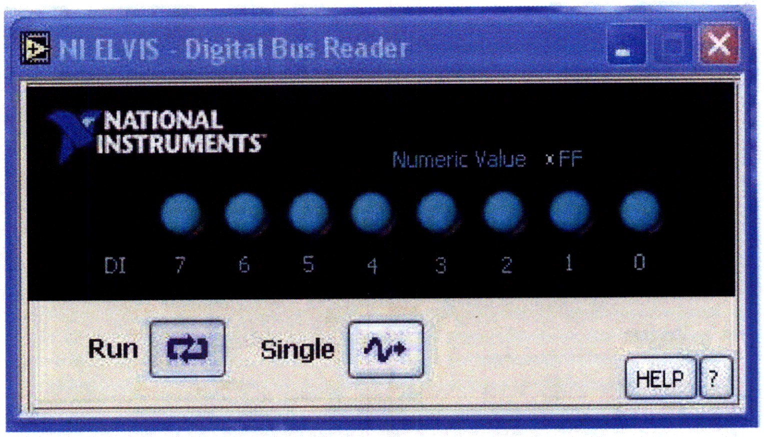


Figure 5-5: Screen shot of user interface for Digital Bus Reader.

5.2.1.1 Dig.VI

The Dig.VI contains all of the functionality required for performing experiments in digital electronics. It was designed with flexibility in mind. Figures 5-5 and 5-6 show the user interfaces provided for the digital bus writer and reader respectively. The digital writer allows for the configuration of 8 different channels and the digital reader can in turn read upto 8 channels. A lab administrator could therefore setup a circuit and wire in up to 8 points of interest in the given circuit to the digital reader, which displays the results of the experiment.

Proper understanding of any digital circuit requires the creation of a truth table. An experiment involving 3 inputs for example has 8 distinct combinations of inputs and would therefore require 8 runs to completely specify the corresponding truth table. The number of runs required is exponentially related to the number of inputs. Given that we are dealing with a batched lab, performing a large number of runs is inconvenient and very time consuming. In response to this limitation, I designed the digital VI to support continuous experiments. Dynamic experiments allow the user to specify the digital input for upto 8 clock steps, which enables the the creation of a 3 input truth table in one experiment run.

Figure 5-7 shows the labView code for the digital VI. As can be seen in the code, I decided to couple the digital writer and reader within the VI. As a result, the reader is constantly monitoring the digital lines and will pick up a digital output

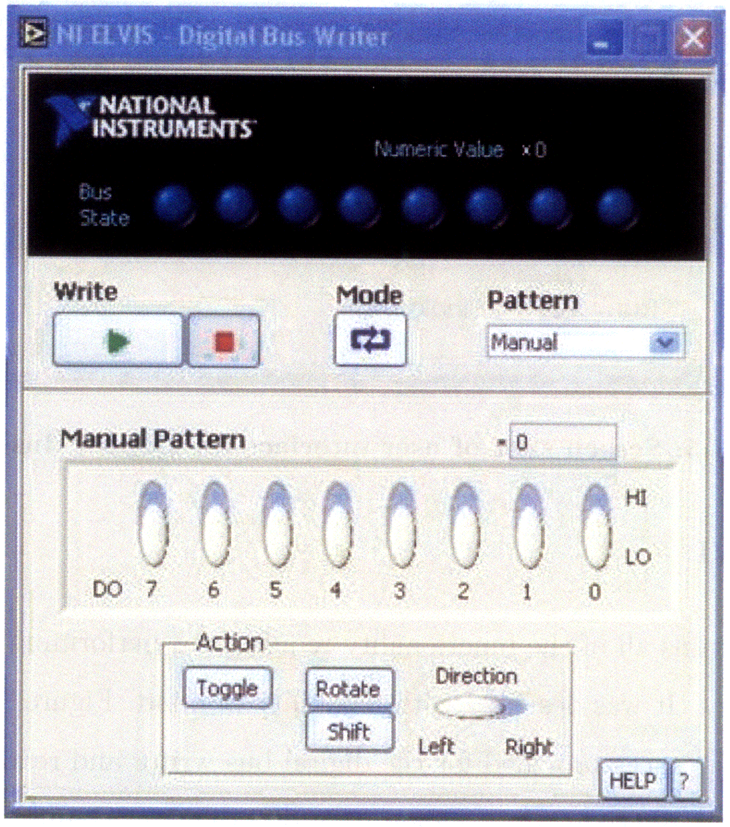


Figure 5-6: Screen shot of user interface for Digital Bus Writer.

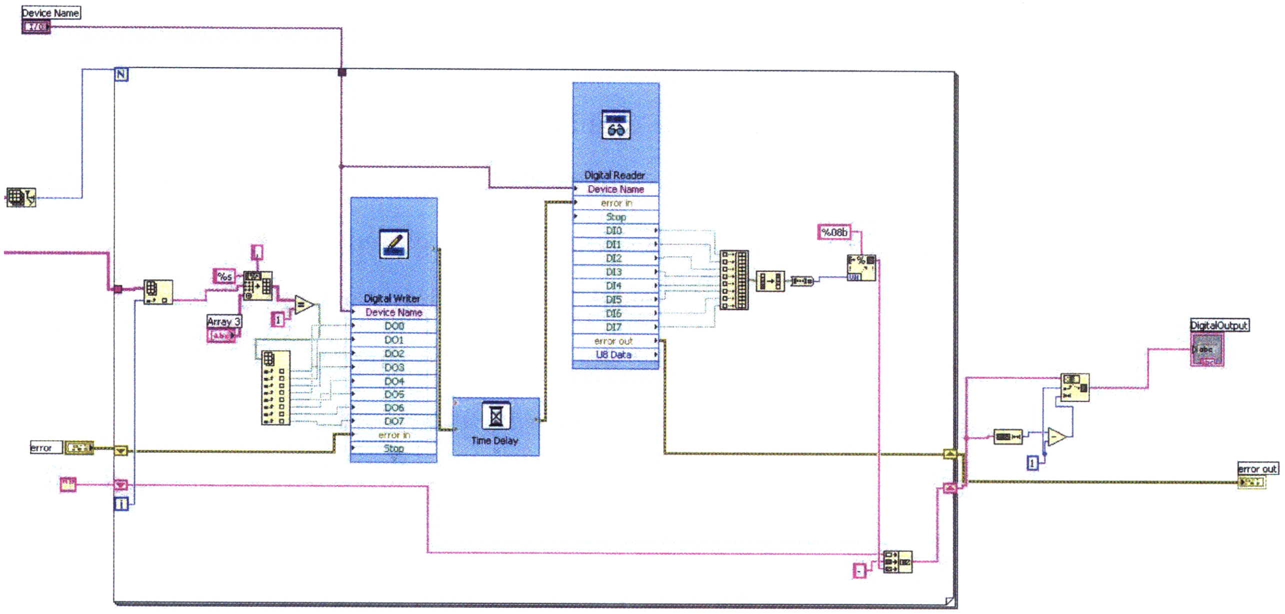


Figure 5-7: Screen shot of LabView code for Dig.vi.

whenever the digital writer is active. Another point to note is the placement of the time delay between the digital writer and reader, which allows ample time for the logic functions to settle before a reading is taken. The millisecond time delay supports the cascading of many integrated circuits since their delay is usually on the order of nanoseconds. The digital functionality is encompassed in a loop to enable continuous experiments to be run. There is a limit to the number of inputs that can be passed into the labView DLL, so in order to circumvent this problem, I only used one input for the digital VI, which is parsed by the VI itself to extract all of the key information.

5.2.2 Experiment Engine

The Experiment engine is an executable file that runs in the background and consists of 3 major methods responsible for most of its functionality namely, *loadjob*, *ParseExperimentSpec*, and *runExperiment.*

5.2.2.1 loadjob

When a user selects a setup, the *loadjob* method is called, which queries the databases and creates the *LabConfiguration.xml* document. This function needed to be changed in ELVIS 3.0 to account for the DOUT instrument type.

5.2.2.2 ParseExperimentSpec

The experiment engine is constantly monitoring the databases for an experiment to be submitted. Once an experiment is submitted, the experiment engine de-queues the job and then parses the experiment specification via the *ParseExperimentSpec* function. This function reads the user specified parameter values within the *ExperimentSpec.xml* document and populates termInfoTable and functInfoTable, which are tables that store information about the terminals present in a setup and the user defined parameters for the terminal instruments respectively.

The first step in my work was to expand both of these tables to include an entry for the DOUT instrument. Next, a case had to be added within the parsing routine to recognize the DOUT instrument and also extract its defining properties, which are the byte string, the number of inputs, and the number of outputs.

5.2.2.3 RunExperiment

The *RunExperiment* method creates a parameter list from the experiment specification, initializes the RunElvisExp executable file, which in turn calls the ElvisWrapper class that runs the experiment through the LabView DLL. It also creates the *ExperimentResult.xml* document. The parameter list had to be updated to include the byte string for the DOUT instrument, but most of the work went into creating two helper functions (*procesDOUT* and *createCombinedDOUTGraph*) whose purpose was to format the digital output from LabView into an easily manipulated form for the client to use.

processDOUT

The output of Dig.VI is formatted as a collection of bits for each clock step. We want to convert this into a collection of timesteps for each bit in order to graph the data in the client. The conversion process is performed by the function *processDOUT* as can be seen in Figure 5-8.

createCombinedDOUTGraph

This function further formats the processed DOUT data into a form that that enables the client to create a stacked strip chart, which is a standard display method for digital data (see Figure 5-9). Just like any graph, a stacked strip chart is a collection of (x,y) coordinate pairs. This collection is specified by an x-axis vector (DOUTTIME) and a y-axis vector (DOUT). In order to understand the numerical values behind a stacked strip chart, we must look at DOUT and DOUTTIME in more detail.

Let's assume for illustration purposes, that we have run an experiment (Experiment 1) in which we read a digital line that starts off high and alternates

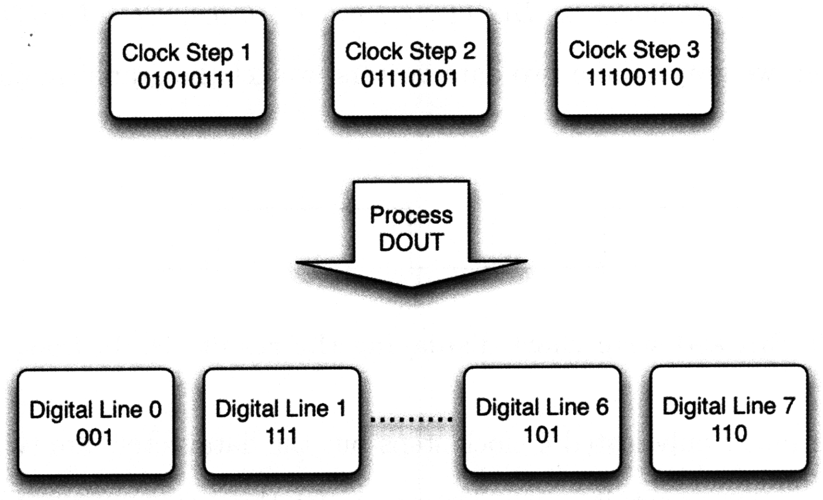


Figure 5-8: Diagram showing the effect of *processDOUT* on the sample output of a digital experiment.

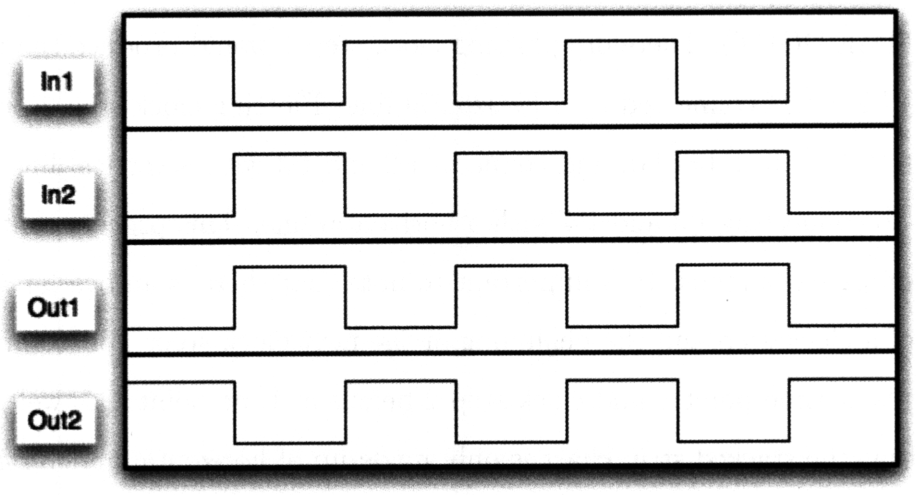


Figure 5-9: Sample stacked strip chart.

thereafter. Also assume that the experiment is run for a total of 4 clock steps. Thus the results are as follows: High, Low, High, Low. The result returned by the labserver for this experiment is: *DOUT<1 1 0 0 1 1 0 0>*, *DOUTTIME<0 1 1 2 2 3 3 4>*. When we graph these two data vectors, we get the graphical output seen in Figure 5-10.

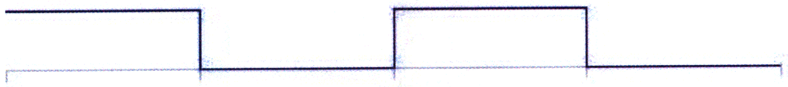
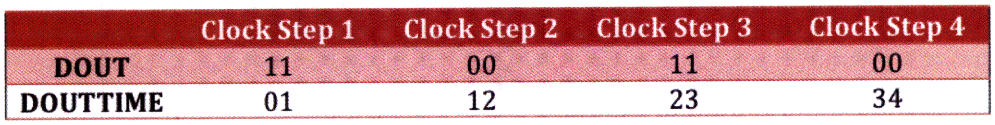


Figure 5-10: Stacked strip chart displaying the result: High, Low, High, Low.

Our experiment only lasted 4 clock steps but the datavectors are twice as long since each bit value occurs in pairs. The reason for this become clear if we split the datavectors into clock steps (see Figure 5-11).



	Clock Step 1	Clock Step 2	Clock Step 3	Clock Step 4
DOUT	11	00	11	00
DOUTTIME	01	12	23	34

Figure 5-11: Table showing how DOUT and DOUTTIME datavectors are broken down by clock step.

The points specified by the results are coordinate pairs in the form (DOUTTIME,DOUT). Therefore plotting clock step 1 would produce the points P1=(0,1),P2=(1,1) connected by a horizontal line. Plotting clock step 2 produces the points P3=(1,0),P4=(2,0) and connects P2 and P3 with a vertical line in addition to connecting P3 and P4 with a horizontal line. This process continues for the remaining clock steps. It is important to note that there is an overlap between the end of a clock step and the beginning of the next clock step. For example, clock step 1 ends at time point 1 and clock step 2 begins at time point 1 as well, which ensures that the stacked strip chart is only made up of horizontal and vertical lines. The overlap in clock steps is also evident by the repeated entries in the DOUTTIME datavector.

Now, imagine performing an experiment (Experiment 2) that investigates the behavior of a 1-input 1-output inverter. Assume that we use the same input pattern

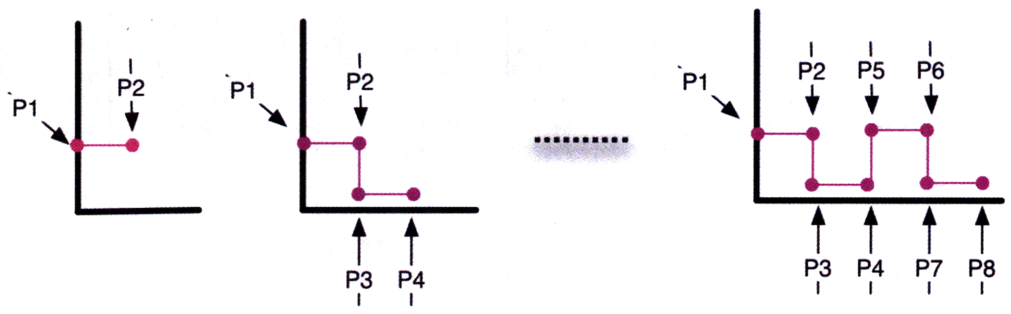


Figure 5-12: Diagram illustrating the creation of a stacked strip chart for an experiment involving one digital line and four clock steps.

as in Experiment 1. Thus the input is the following: High, Low, High, Low. The resulting output of the inverter is: Low, High, Low, High. The labserver returns the following datavectors: *DOUT<1 1 0 0 1 1 0 0 2 2 3 3 2 2 3 3>*, *DOUTTIME<0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4>*. The desired graphical output is shown in Figure 5-13.

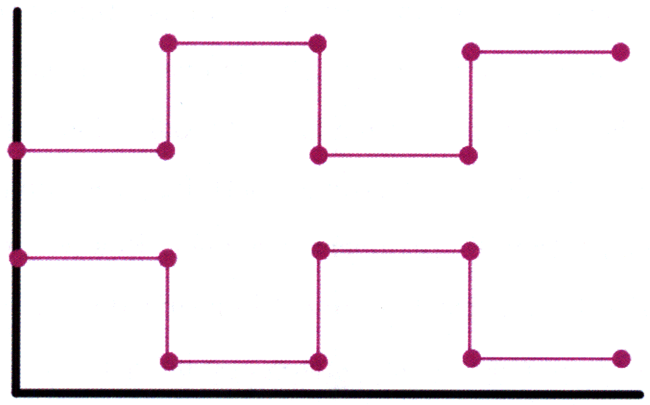
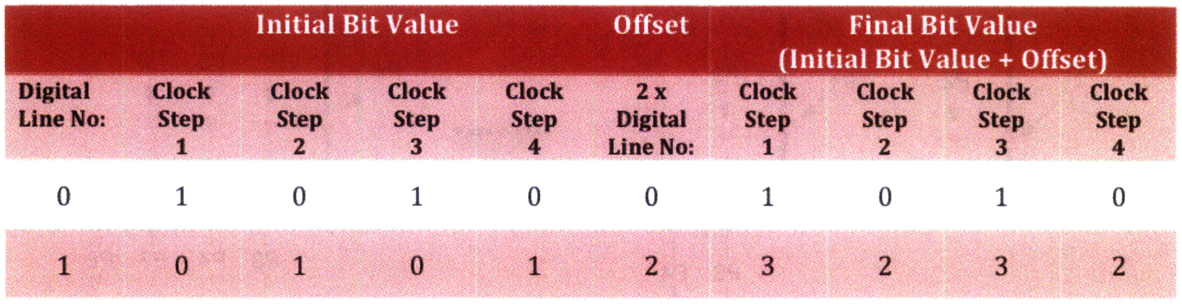


Figure 5-13: Desired stacked strip chart for experiment involving a 1-input, 1-output inverter.

You should think about this result as two waveforms stacked on top of each other. In order for this to occur, two conditions must be met. Firstly, each waveform should have the same DOUTTIME data vector, which ensures that they are aligned horizontally. The second requirement involves adding an offset equal to twice the digital line number to each bit value in the DOUT data vector to create the stacking effect. This process is shown graphically Figure 5-14.



	Initial Bit Value				Offset	Final Bit Value (Initial Bit Value + Offset)			
Digital Line No:	Clock Step 1	Clock Step 2	Clock Step 3	Clock Step 4	2 x Digital Line No:	Clock Step 1	Clock Step 2	Clock Step 3	Clock Step 4
0	1	0	1	0	0	1	0	1	0
1	0	1	0	1	2	3	2	3	2

Figure 5-14: Table summarizing the arithmetic calculations that produce the DOUT data vector for the stacked strip chart in Figure 5-13.

5.2.3 Validation Engine

The main function of the validation engine is to ensure that the experiment specified by a student meets the requirements established by the lab administrator before the the *runExperiment* function is called. The validation engine was updated to verify that the input byte string was composed of either 0s and 1s. In ELVIS 2.0, a time domain experiment was required to have an input source (FGEN of Arb instrument) and an output source(scope) and a frequency domain experiment was required to have a bode instrument. I modified the validation engine in ELVIS 3.0 to waive the requirements discussed above for both frequency domain and time domain experiments if a DOUT instrument is present. However, regardless of whether or not a DOUT instrument is present in a time domain experiment, if an input source is present, an output source must also be present.

5.2.4 Lab Server Administration Pages

The lab server administration pages provide the user interface through which lab administrators specify the instruments and diagram for a given setup. It also provides the interface for creating, deleting and activating setups in addition to a series of other useful functions for administrators. The interface for specifying instruments needed updating to include the DOUT instrument. In addition, the stored procedure *AddSetupTerminal* and the constraints for the *SetupTerminalConfig* table were updated.

5.3 WebLab Client

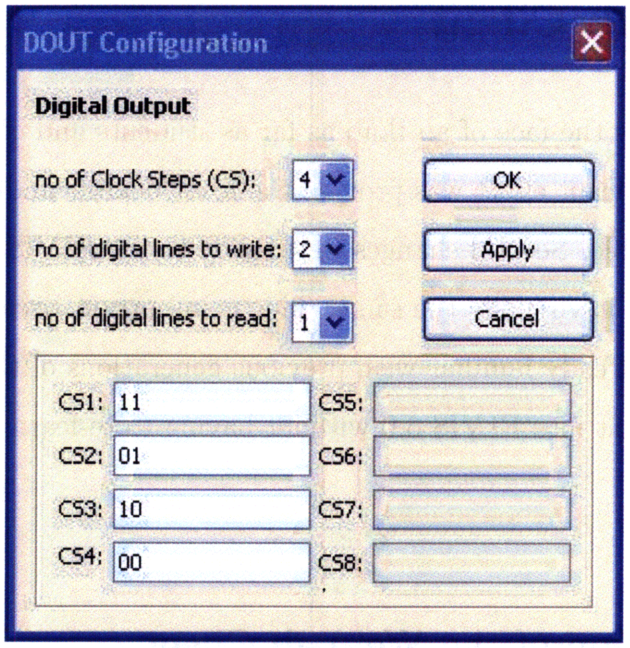
The weblab client is the face of an iLab as far as students and educators are concerned, so significant effort was provide the most flexible andintuitive user experience as possible. Several changes were made to the ELVIS 3.0 client, however the user experience is virtually the same. This is important because it allows students that are already familiar with previous generations of the ELVIS client to be able to jump right into ELVIS 3.0 without having to re-learn an entirely new user interface.

5.3.1 Operation of the WebLab Client

When a student launches the client from the service broker, the *LabConfiguration.xml* is sent to the client. This document is parsed by the client via the *parseXMLLabConfiguration()* function in the *LabConfiguration.java* class, which creates a list of terminals that make up the current active setup. The schematic panel displays an image of the circuit along with the user-configurable instruments as can be seen in Figure 2-2). In order to configure an instrument, a user must click on the instrument icon and specify the required parameter values in the dialog box that appears (see Figure 5-15). Once all of the instruments have been configured the experiment is submitted by clicking the run button in the top right corner. When an experiment is submitted, the *ExperimentSpecification.java* class creates the *ExperimentSpecification.xml* document, which is validated and sent to the labserver via the service broker. The results of the experiment are parsed by within the *ExperimentResult.java* class and then displayed in the results panel.

5.3.2 Client Updates in ELVIS 3.0

There were two types of changes/additions made to the client, those affecting the main panel of the client, and those affecting the graphical utilities of the client.



no of Clock Steps (CS): 4
no of digital lines to write: 2
no of digital lines to read: 1

CS1: 11
CS2: 01
CS3: 10
CS4: 00
CS5:
CS6:
CS7:
CS8:

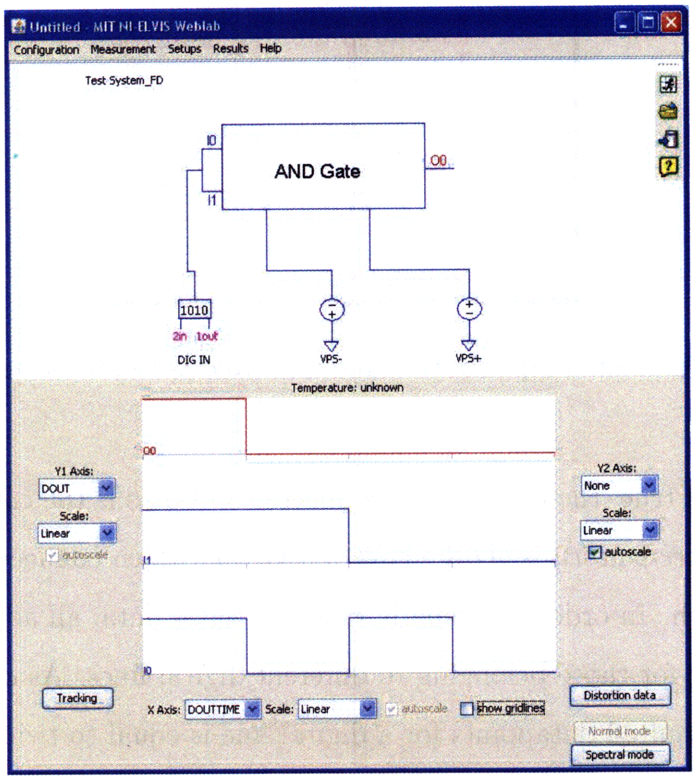
Figure 5-15: Screen shot of dialog for configuring a DOUT instrument. Users must specify the number of clock steps, the number of inputs, number of outputs in addition to the bit patterns for each clock step. Here we have specified a bit pattern for a 2-input, 1-output experiment run for 4 clock steps.

5.3.2.1 Updating the MainFrame

Figure 5-16 shows the ELVIS 3.0 client. It is important to note the addition of the check box that allows a user to toggle the gridlines on and off in the graph. This was added to improve the aesthetic qualities and simplicity of the graph when displaying stacked strip charts. In addition, when a stacked strip chart is being displayed on the ELVIS 3.0 client, the graph is automatically autoscaled and the y-axis is not labeled with its maximum, minimum or units per division. Given that digital domain data is unitless, it was important to prevent the display of false units. This preventive measure will greatly reduce any potential confusion experienced by students when performing digital experiments. Input waveforms are blue and output waveforms are red.

5.3.2.2 Updating Graphical Utilities

Significant changes were made to the graphing utilities to support the display of stacked strip charts. In order to graph data, a user must specify an x-axis and a



Y1 Axis: DOUT
Scale: Linear
autoscale
Y2 Axis: None
Scale: Linear
[✓ autoscale]
X Axis: DOUTTIME
Scale: Linear
autoscale
show gridlines

Figure 5-16: ELVIS 3.0 client displaying the results of the characterization of an AND gate specified in Figure 5-15. Each waveform is labeled and numbered.

y-axis from the respective drop down menus, resulting in pointers to the specified datavectors. Together, these datavectors specify a set of coordinate pairs representing the results. The *plot()* method draws the graph by looping through the set of coordinate pairs and connecting adjacent points by a line. In order to understand the limitations of the *plot()* method, it is important to look at an example where it produces an undesired output. Referring back to the experiment involving the 1-input 1-output inverter, the lab server returns the following data vectors: *DOUT<1 1 0 0 1 1 0 0 2 2 3 3 2 2 3 3>*, *DOUTTIME<0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4>*. Plotting this data results in the graph in Figure 5-17.

The result is almost as desired except for the diagonal line that connects the waveforms for two separate digital lines. The *plot()* function uses a datatype called a *connectpattern* that specifes a function for determining which points to connect in the graph. This datattype has an *isConnected()* method that returns true for the index of every point that should connected to the preceding point. So for example, if

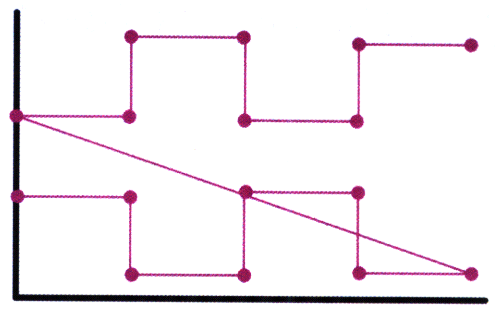


Figure 5-17: Stacked strip chart when *connectpattern* is set to *ALWAYS-CONNECT*.

isConnected(4) is true, then P4 will be connected to P3 as is the case in Figure 5-12. The default *connectpattern* is *ALWAYS-CONNECT*, which connects every adjacent point in the graph. In order to properly display digital data, all adjacent points are connected except for those belonging to different digital lines. As discussed above, the number of required datapoints for a digital line is equal to twice the number of clock steps. Therefore points immediately following a point whose index is a multiple of twice the number of clock steps, should not be connected to the previous point because they represent two different digital lines. Figure 5-18 clarifies this idea. Thus since the required *connectpattern* for a digital output is a function of the number of clock steps, I had to create the function *plot2()* that is called whenever digital data is to be graphed, in other words when the user specifies *DOUTTIME* as the x-axis and *DOUT* as one of the y-axis. This function activates the appropriate *connectpattern* depending on the number of clock steps for a given experiment in addition to setting color schemes and labeling the digital lines.

5.4 Testing ELVIS 3.0

Tests were performed on several logic gates to ensure the correct functioning of the system. I performed basic tests with single logic gates as well as more complicated tests with several logic gates. Figures 5-19 and 5-20 show sample results for our test experiments. displays some of the experimental results obtained. In addition to these technical tests, I consulted frequently with colleagues in the lab to ensure that

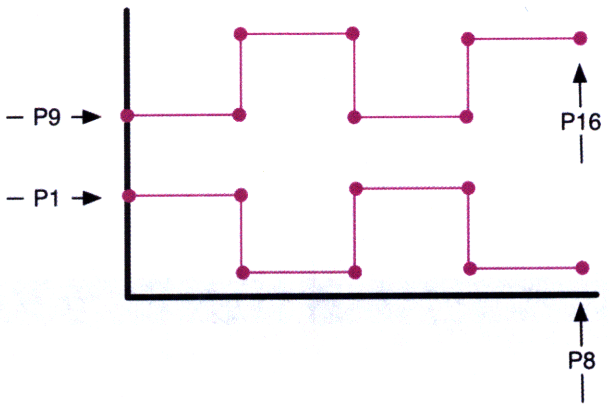
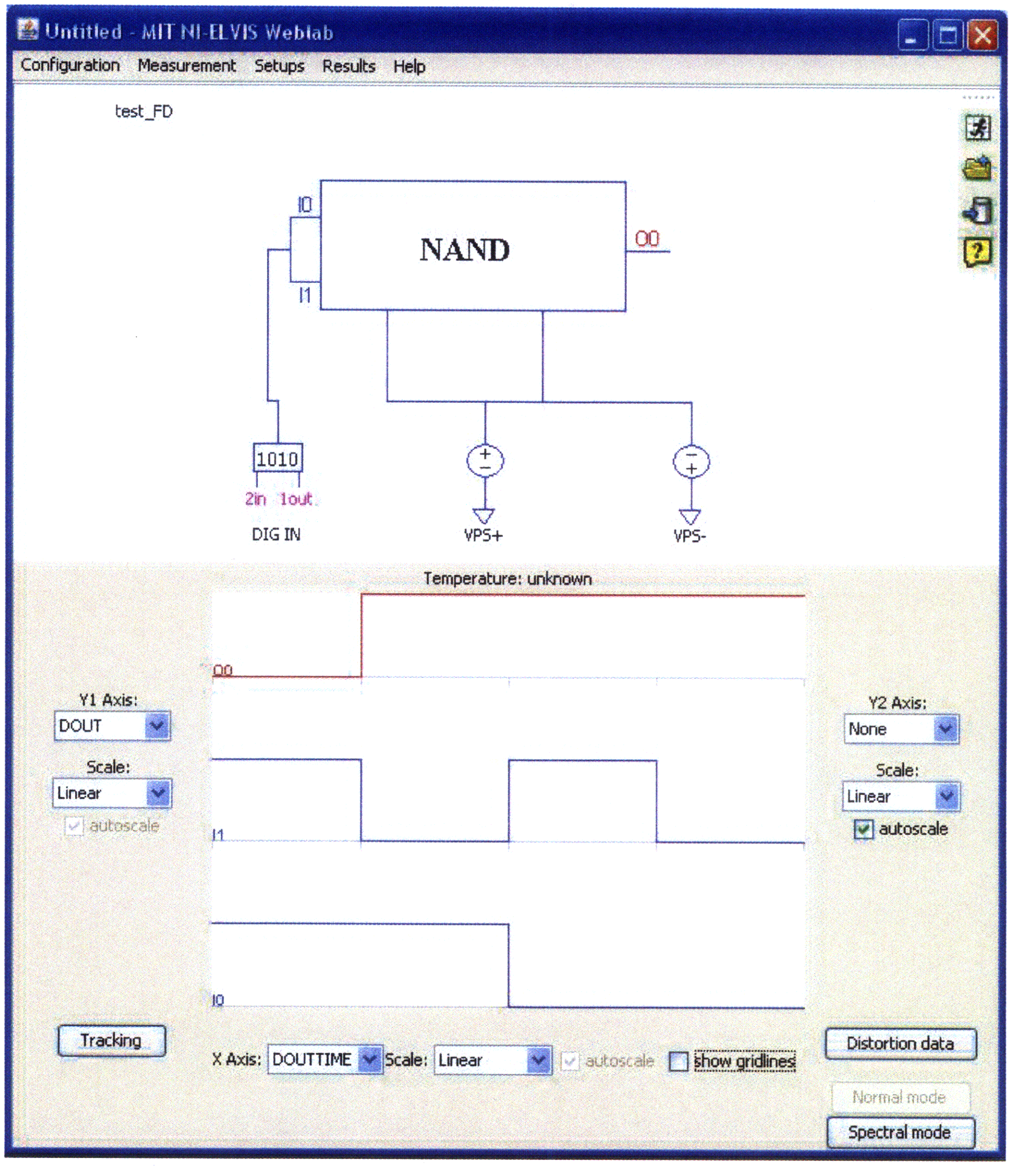


Figure 5-18: In order to properly display the stacked strip chart, P8 and P9 are not connected because they represent datapoints on adjacent digital lines.

the client was as aesthetically pleasing and intuitive as possible. These exchanges prompted several tweaks in arriving at the final product.



Y1 Axis: DOUT
Scale: Linear
autoscale
Y2 Axis: None
Scale: Linear
[✓ autoscale]
X Axis: DOUTTIME Scale: Linear autoscale show gridlines

Figure 5-19: Experimental results for the characterization of a NAND gate.

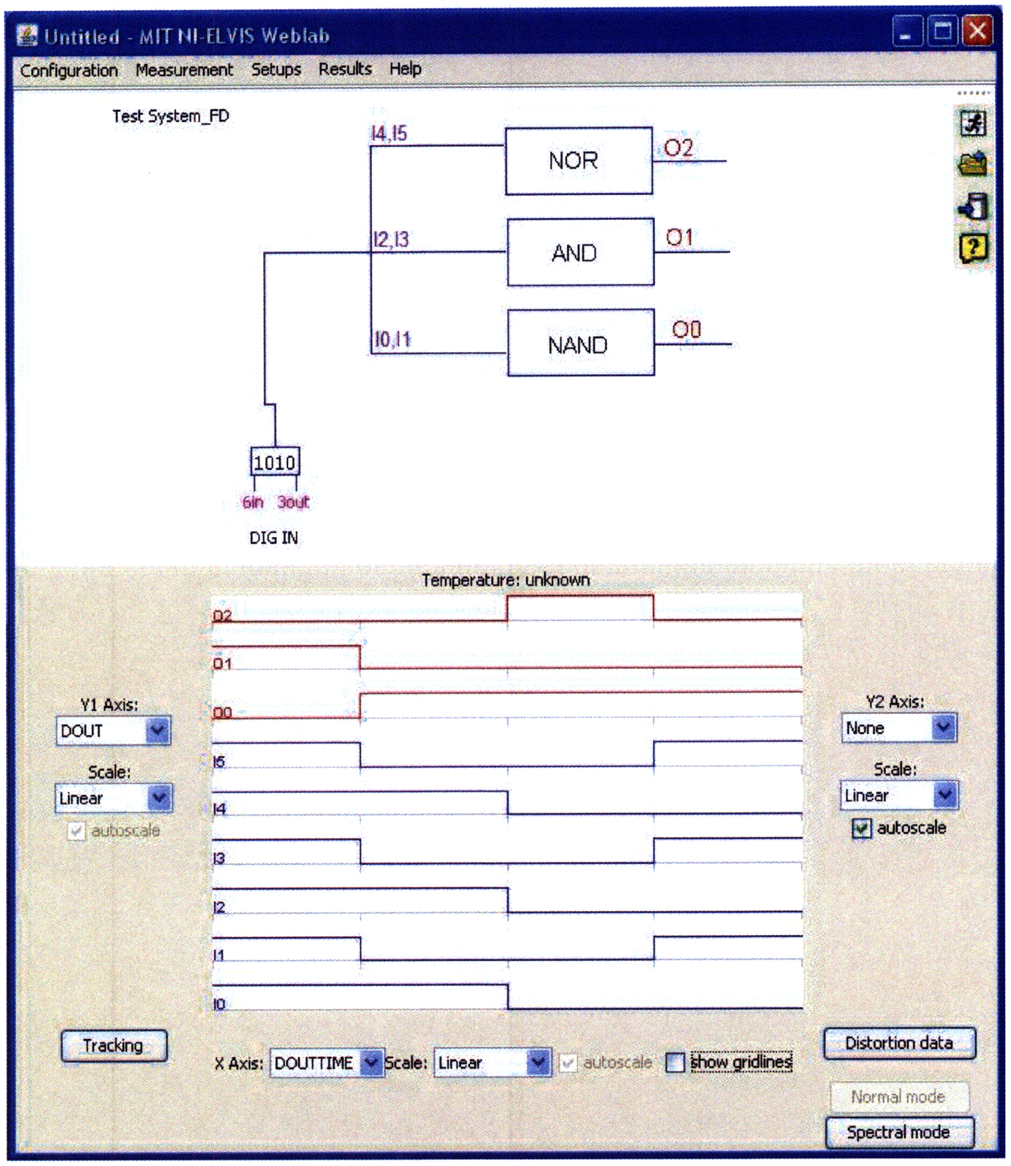


Figure 5-20: Experimental results for the simultaneous characterizations of NAND, AND, and NOR gates.

Chapter 6

Conclusions

ELVIS 3.0, by supporting experiments in digital electronics, has greatly improved the impact of the iLabs platform. The ELVIS iLabs now offer an even greater return on investment by providing pedagogically useful labs covering a larger portion of Electrical Engineering curricula. Since integrated circuits are the backbone of modern day computing, the incorporation of digital capabilities will create a new class of user of the ELVIS 3.0, namely the computer science and engineering community. Diversifying our target audience is a very important step in the ongoing iterative process of the iLab project. As we get feedback from students in other disciplines, we can improve the virtual laboratory experience for all users by creating a layer of abstraction around the common set of functionality and services required by all labs. This new layer will provide the flexibility to create multi-disciplinary labs with relative ease.

6.1 Contributions

In this section I state some of the most noteworthy contributions of my work.

- Unified parallel development efforts to create a single code base from which further development can proceed.
- Expanded the ELVIS iLab to support experiments in digital electronics.

- Facilitated a training session at Makerere University that set the foundation for the timely completion of their final projects.

6.2 Recommendation for Direction of iLab Project

There has been a great deal of progress since ELVIS 1.0 was first introduced. Currently, very complex experiments can be conducted on the ELVIS iLab. Future development of this project falls into three major catergories.

The first alternative is to continue the process of exposing instruments on the ELVIS platform. Currently, 8 out of the 12 available instruments have been exposed. The natural followup would be to expose the remaining 4 instruments. At the top of this list would be the Dynamic Signal Analyzer instrument. The popularity of this instrument is clear when you consider the amount of traffic on the open iLab DSA site. The other instruments to be exposed are the Impedance Analyzer in addition to the Two and Three Wire Current-Voltage Analyzer.

The second alternative is also related to the ELVIS platform but takes a slightly different approach. National Instruments makes several specialty plug-in boards for the ELVIS platform, which expand its functionality to include telecommunications and microcontroller programming for example. A good example is the EMONA board, which is currently being developed at Makerere University for telecommunications labs. Thus further development efforts could center around incorporating these boards into the iLabs framework. If this path is taken, a significant amount of work should go into creating an intuitive and educationally useful lab client.

The third alternative is a complete change in direction from the current path of the ELVIS iLab. During the iLabs workshops, we realized that there was a great deal of interest from faculty outside of Electrical Engineering and Physics. There was a Biology professor at the Open University of Tanzania, who was particulary excited about the iLabs concept and was eager to help develop biology based iLabs. Thus research could be done to identify and incorporate the Biology or Chemical Engineering equivalent of ELVIS and incorporate that into the iLabs framework. I am convinced that this path would have the greatest impact on the project as a whole.

6.3 Future of iLab-Africa Partnership

The relationship with our partners in Africa have strengthened significantly as a result of open and honest communication. The numerous exchanges has also accelerated their progress. A major turning point during my tenure on the project was our visit to Makerere University in January 2009. Before this visit, our time spent in Africa would be split evenly between UDSM and Makerere University. The success of the most recent trip was due to the fact that we were able to get the UDSM students to participate in the training session at Makerere. Not only did this promote collaboration between students from both Universities, but it served to strengthen their relationship. The teams from both Universities are now well versed in the technical aspects of the project so the size and scope of their contributions will increase. In order for the iLab Africa project to remain sustainable, the role of these Universities must change. They must take on a greater leadership role within the East African community and in effect become local hubs. They must help with the recruitment of Universities in order to help grow the project. Given the energy of the teams at both UDSM and Makerere I am convinced that the project is in good hands.

Appendix A

LabConfiguration.xml

```
<?xml version='1.0' encoding='utf-8' standalone='no' ?>
<!DOCTYPE labConfiguration SYSTEM 'http://localhost/labserver/xml/labConfiguration.dtd'>
<labConfiguration lab='MATEC ESyst iLab' specversion='0.1'>
<setup id='9'>
<name>Audio System</name>
<description>multistage audio system</description>
<imageURL>http://olid.mit.edu/images/setups/9RLcircuit.PNG</imageURL>
<mode type='TD' enabled='true'>
<terminal instrumentType='DOUT' instrumentClass='control' instrumentNumber='6' setupTermID='6'>
<label>Digital Out</label>
<pixelLocation>
<x>2</x>
<y>45</y>
</pixelLocation>
</terminal>
<terminal instrumentType='FGEN' instrumentClass='input' instrumentNumber='1' setupTermID='1'>
<label>FGEN Input</label>
<pixelLocation>
<x>123</x>
<y>43</y>
</pixelLocation>
</terminal>
<terminal instrumentType='SCOPE' instrumentClass='output' instrumentNumber='3' setupTermID='3'>
<label>Oscilloscope</label>
```

```
<source name='Circuit Output' channel='ACH0'>
<pixelLocation>
<x>45</x>
<y>34</y>
</pixelLocation>
</source>
<source name='Amp Stage Out' channel='ACH1'>
<pixelLocation>
<x>23</x>
<y>56</y>
</pixelLocation>
</source>
<source name='FGEN' channel='FGEN'>
<pixelLocation>
<x>123</x>
<y>43</y>
</pixelLocation>
</source>
<postProcessOptions>SPEC,DIST</postProcessOptions>
</terminal>
<terminal instrumentType='VPSPos' instrumentClass='control' instrumentNumber='4' setupTermID='4'>
<label>Var Power Supply +</label>
<pixelLocation>
<x>166</x>
<y>84</y>
</pixelLocation>
</terminal>
<terminal instrumentType='VPSNeg' instrumentClass='control' instrumentNumber='5' setupTermID='5'>
<label>Var Power Supply -</label>
<pixelLocation>
<x>345</x>
<y>23</y>
</pixelLocation>
</terminal>
<terminal instrumentType='COM' instrumentClass='control' instrumentNumber='7' setupTermID='0'>
<label></label>
```

```
<pixelLocation>
<x>100</x>
<y>150</y>
</pixelLocation>
<subCOM subCOMType ="horizR" instrumentNumber='7' setupTermID='7'>
<label>100 K</label>
</subCOM>
<subCOM subCOMType ="horizR" instrumentNumber='7' setupTermID='8'>
<label>200 K</label>
</subCOM>
</terminal>
</mode>
<mode type='FD' enabled='true'>
<terminal instrumentType='BODE' instrumentClass='output' insturmentNumber='8' setupTermID='9'>
<label>Bode Analyzer</label>
<pixelLocation>
<x>64</x>
<y>78</y>
</pixelLocation>
</terminal>
</mode>
</setup>
</labConfiguration>
```

Appendix B

ExperimentSpecification.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no" ?>
<!DOCTYPE experimentSpecification SYSTEM "http://localhost/xml/ExperimentSpecification.dtd">
<experimentSpecification lab="MATEC ESyst iLab" specversion="0.1">
<setupID mode='FD'>9</setupID>
<terminal instrumentType="BODE" instrumentClass="input" instrumentNumber="7" setupTermID = "7" >
<function type="BODE">
<startFreq>10</startFreq>
<stopFreq>5000</stopFreq>
<stepsPerDec>5</stepsPerDec>
<inputAmp>2</inputAmp>
</function>
</terminal>
<terminal instrumentType="VPSPos" instrumentClass="control" instrumentNumber="4" setupTermID = "4"
<function type="VPSFunction">
<value>5.0</value>
</function>
</terminal>
<terminal instrumentType="VPSNeg" instrumentClass="control" instrumentNumber="5" setupTermID = "5"
<function type="VPSFunction">
<value>-5.5</value>
</function>
</terminal>
<terminal instrumentType="DOUT" instrumentClass="control" instrumentNumber="6" setupTermID = "6">
<function type="DOUTFunction">
```

```
<byte>01100111</byte>
<Input>4</Input>
<Output>2</Output>
</function>
</terminal>
<terminal instrumentType="COM" instrumentClass="control" instrumentNumber="7" setupTermID = "0">
<function type="subCOM" setupTermID = "73">
<subCOM>horizR</subCOM>
<label>100k</label>
</function>
</terminal>
</experimentSpecification>
```

Appendix C

ExperimentResult.xml

```
<?xml version='1.0' encoding='utf-8' standalone='no' ?>
<!DOCTYPE experimentResult SYSTEM 'http://localhost/xml/experimentResult.dtd'>
<experimentResult lab='MATEC ESyst iLab' specversion='0.1'>

<datavector name='Gain' units='dB' scalable='false' type='vector'>-12.3682857521954 -18.9917193553
 -12.3001253042114 -6.14745399734583 -2.90944045250843 -0.100294773285425 0.527363392938682
0.839316439511952 0.993080303257754 1.01255667697537 13.0111791863401 13.2534987682982
13.4865654371622 13.6044492322361 </datavector>

<datavector name='Phase' units='degrees' scalable='false' type='vector'>-69.3246746172284
-58.2213082414745 -50.2184638790464 -34.1847322356342 -17.7876755472731 0.624672634595873
8.25502781716833 12.9608384424874 15.2443676098177 17.7371930556832 40.5524999444354
43.2629805793569 44.7180160233491 47.469405384509 </datavector>

<datavector name='Frequency' units='Hz' scalable='true' type='vector'>10 15.8489319246111
25.1188643150958 39.8107170553497 63.0957344480193 100 158.489319246111 251.188643150958
398.107170553497 630.957344480193 1000 1584.89319246111 2511.88643150958 3981.07170553497
 </datavector>

<datavector name='DOUTTIME' units='bool' scalable='true' type='vector'>0 1 1 2 2 3 3 4 0
1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2
3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0
1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2 3 3 4 0 1 1 2 2
3 3 4</datavector>
```

```
<datavector name='DOUT' units='bool' scalable='true' type='vector'>0 0 0 0 0 0 0 0 1 1 1
1 0 0 0 0 2 2 2 2 2 2 2 2 3 3 2 2 2 2 3 3 4 4 4 4 4 4 4 4 5 5 5 5 4 4 4 4 6 6 6 6 6 6 6
6 7 7 6 6 6 6 7 7 8 8 8 8 8 8 8 8 9 9 9 9 8 8 8 8 10 10 10 10 10 10 10 10 11 11 10 10 10
10 11 11 12 12 12 12 12 12 12 12 12 12 13 13 13 13 13 13 14 14 14 14 14 14 14 14 15 15
14 14 14 14 14 14 16 16 16 16 16 16 16 16 16 16 16 16 17 17 16 16</datavector>
</experimentResult>
```

Appendix D

LabConfiguration.dtd

```
<!ELEMENT labConfiguration (setup*)>
<!ATTLIST labConfiguration lab CDATA #REQUIRED
specversion CDATA #REQUIRED>
<!ELEMENT setup (name, description, imageURL, mode+)>
<!ATTLIST setup id CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT imageURL (#PCDATA)>
<!ELEMENT mode (terminal*)>
<!ATTLIST mode  type (TD | FD) #REQUIRED
 enabled (true | false) #REQUIRED>
<!ELEMENT terminal (label, pixelLocation?,subCOM*, ((source+, postProcessOptions?)
|(enabledModes, file*)))>
<!ATTLIST terminal  instrumentType (FGEN | SCOPE | ARBO | VPSPos | VPSNeg | DOUT | COM | BODE)
#REQUIRED
    instrumentClass (input | output | control | switch) #IMPLIED
    instrumentNumber CDATA   #REQUIRED
    setupTermID         CDATA    #REQUIRED>
<!ATTLIST subCOM subCOMType  CDATA #REQUIRED
 instrumentNumber CDATA   #REQUIRED
 setupTermID CDATA   #REQUIRED>
<!ELEMENT subCOM (label)>
<!ELEMENT label (#PCDATA)>
<!ELEMENT pixelLocation (x, y)>
```

```
<!ELEMENT x (#PCDATA)>
<!ELEMENT y (#PCDATA)>
<!ELEMENT source (pixelLocation)>
<!ATTLIST source  name CDATA #REQUIRED
    channel CDATA #REQUIRED>
<!ELEMENT postProcessOptions (#PCDATA)>
<!ELEMENT enabledModes (#PCDATA)>
<!ELEMENT file (name, description, URL, length, recSamplingRate, recTotalSamples)>
<!ATTLIST file  WAVID CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT length (#PCDATA)>
<!ELEMENT recSamplingRate (#PCDATA)>
<!ELEMENT recTotalSamples (#PCDATA)>
```

Appendix E

ExperimentSpecification.dtd

```
<!ELEMENT experimentSpecification (setupID, terminal+)>
<!ATTLIST experimentSpecification lab CDATA #REQUIRED
specversion CDATA #REQUIRED>
<!ELEMENT setupID (#PCDATA)>
<!ATTLIST setupID mode (TD | FD) #REQUIRED>
<!ELEMENT terminal (function+)>
<!ATTLIST terminal instrumentType (FGEN | SCOPE | ARBO | VPSPos | VPSNeg | DOUT | BODE)
#REQUIRED
instrumentClass (INPUT | OUTPUT | CONTROL) #IMPLIED
instrumentNumber CDATA #REQUIRED>
<!ELEMENT function ((waveformType, frequency, amplitude, offset) |
(scope+, samplingRate, samples, trigger) |
(mode, arbSamplingRate, arbSamples, ((frequency, amplitude, offset, phase, dutyCycle?) |
        (waveform) |
             (fileID))) |
    (value) |
   (byte) |
(startFreq, stopFreq, stepsPerDec, inputAmp))>
<!ATTLIST function type (WAVEFORM | SAMPLING | ARB | BODE | VPSFunction | DOUTFunction) #REQUIRED>
<!ELEMENT waveformType (#PCDATA)>
<!ELEMENT frequency (#PCDATA)>
<!ELEMENT amplitude (#PCDATA)>
<!ELEMENT offset (#PCDATA)>
```

```
<!ELEMENT scope (name?, source, paSpec, paDist)>
<!ATTLIST scope channel (A | B) #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT source (#PCDATA)>
<!ELEMENT paSpec EMPTY>
<!ATTLIST paSpec perform (true | false) #REQUIRED>
<!ELEMENT paDist EMPTY>
<!ATTLIST paDist perform (true | false) #REQUIRED>
<!ELEMENT samplingRate (#PCDATA)>
<!ELEMENT samples (#PCDATA)>
<!ELEMENT trigger (source, slope?, level?)>
<!ELEMENT source (#PCDATA)>
<!ELEMENT slope (#PCDATA)>
<!ELEMENT level (#PCDATA)>
<!ELEMENT mode (#PCDATA)>
<!ELEMENT arbSamplingRate (#PCDATA)>
<!ELEMENT arbSamples (#PCDATA)>
<!ELEMENT frequency (#PCDATA)>
<!ELEMENT amplitude (#PCDATA)>
<!ELEMENT phase (#PCDATA)>
<!ELEMENT dutycycle (#PCDATA)>
<!ELEMENT waveform (#PCDATA)>
<!ATTLIST waveform dt CDATA #REQUIRED>
<!ELEMENT fileID (#PCDATA)>
<!ELEMENT value (#PCDATA)>
<!ELEMENT byte (#PCDATA)>
<!ELEMENT startFreq (#PCDATA)>
<!ELEMENT stopFreq (#PCDATA)>
<!ELEMENT stepsPerDec (#PCDATA)>
<!ELEMENT inputAmp (#PCDATA)>
```

Appendix F

ExperimentSpecification.dtd

```
<!ELEMENT experimentResult (datavector+)>
<!ELEMENT datavector (#PCDATA)>
<!ATTLIST experimentResult lab CDATA #REQUIRED
specversion CDATA #REQUIRED>
<!ATTLIST datavector name CDATA #REQUIRED
units CDATA #REQUIRED
scalable (true | false) #IMPLIED
type (vector | scalar) #IMPLIED>
```

Bibliography

[1] Thomas E. Brewer. Georgia institute of technology simplifies teaching circuit design with ni elvis, ni labview, and ni multisim. http://www.ni.com/academic/ni_elvis/universities_using_nielvis.htm.

[2] Jesus del Alamo. Realizing the Potential of iLabs in sub-Sahara Africa, 2005. http://www-mtl.mit.edu/ alamo/del%20Alamo.pdf.

[3] Samuel Gikandi. Elvis ilab: A flexible platform for online laboratory experiments in electrical engineering. Master's thesis, Massachusetts Institute of Technology, 2006.

[4] Bryant J. Harrison. Expanding the capabilities of the elvis ilab using component switching. Master's thesis, Massachusetts Institute of Technology, 2008.

[5] Judson Harward. Service broker to lab server api. http://icampus.mit.edu/iLabs/Architecture/downloads/protectedfiles/Service0Broker

[6] iCampus: the MIT-Microsoft Alliance. ilab: Remote online laboratories. http://icampus.mit.edu/projects/ilabs.shtml.

[7] National Instruments. *Getting Started With LabVIEW*, 2007. http://www.ni.com/pdf/manuals/373427c.pdf.

[8] Adnaan Jiwaji. Modular development of an educational remote laboratory platform for electrical engineering: the elvis ilab. Master's thesis, Massachusetts Institute of Technology, 2009.

[9] Steve Lerman and Jesus del Alamo. ilab: Remote online laboratories. 2000. http://icampus.mit.edu/projects/ilabs.shtml.

[10] National Instruments. *NI Educational Laboratory Virtual Instrumentation Suite (NI ELVIS)*, 2006. http://zone.ni.com/devzone/cda/tut/p/id/3711.

[11] Carter Macready Snowden. The ilab project. 1986. https://wikis.mit.edu/confluence/display/ILAB2/Home.

[12] David Zych. Client to service broker api. http://icampus.mit.edu/iLabs/Architecture/downloads/protectedfiles/Clientto