WebSimMicro: Web-based Simulation of Microelectronic Devices*

Tao Tang and Chu R. Wie+

State University of New York at Buffalo, Dept. of Electrical Engineering, Bonner Hall,

Buffalo, NY 14260

Keywords Java, web-based, microelectronic device simulation, legacy simulator

Abstract

We have developed a web-based Java front-end for microelectronic device simulation applications. We used a framework, that we have previously proposed, for extending legacy simulation applications to the web. This we shall call WebSimMicro, the Web-based Simulation on Microelectronic devices. In this paper, we discuss the architecture of WebSimMicro and the design approach using the proposed framework. This WebSimMicro project contains a series of semiconductor device simulation applets, including applets for metal-oxide-semiconductor transistors and bipolar transistors which can be accessed on our website.

* Supported by National Science Foundation

+ Author to whom correspondence should be addressed: wie@eng.buffalo.edu

1 Introduction

Various online simulators have been built based on the frameworks proposed for extending the simulation applications to the web [Kapadia and Figueiredo 2000; Haupt et al. 1999; Romberg 1999]. For example, the Purdue University Network Computing Hubs (PUNCH) [Kapadia and Figueiredo 2000] is a large web-based simulation infrastructure which incorporates various legacy simulation tools and maintains a significant number of regular users worldwide. This text-based simulation tool was extended to the web via dynamically-generated user interfaces, and the graphical interface tools were extended with remote display management technologies [Kapadia and Fortes 2000] such as Broadway [X Consortium Inc. 1996] and VNC [Richardson 1998]. It implements virtual URL translation and the compilable HTML templates, and hides the remote application invocation details from the end user. During several years of growth, PUNCH has incorporated the simulations in the domains of nanotechnology, computer architecture, parallel programming, and VLSI design [Kapadia, Fortes, and Lundstrom 1997]. The infrastructure of PUNCH is a client/server based two-tier architecture, and mostly implemented in the platform-dependent languages [Kapadia and Fortes 2000]. Another of web-based simulation example is WebFlow [Akarsu 1999]. WebFlow implements such example projects as the Landscape Management System (LMS) and Quantum Simulations of condensed matter systems (QS), which used GAUSSIAN and GAMESS as the back-end applications [Akarsu 1999; Premchandran 1997]. Most simulation packages require some software installations on the client side and require maintaining a constant connection with the simulation server [LMS website 2001].

We have developed a framework for web-based simulation using Java [Tang 2001]. In this paper, we report an application of this framework to the web-based simulation of microelectronic

devices. We have constructed a series of web-based simulation applets, which can be used for educational and research purposes. We report the design and implementation of an architecture design for the web extension of two native simulation programs: the Florida Object-Oriented Device Simulator (FLOODS) [Liang 1994] and MINIMOS [Selberherr 1980], a popular simulation program of Metal-Oxide-Semiconductor Field-Effect Transistors (MOSFET).

FLOODS is a microelectronic process and device simulator, similar to the microelectronic devices and process simulators, PISCES and SUPREME [Stanford University TCAD website 2001], developed and disseminated by Professor Mark Law's group at University of Florida [FLOODS website 2001]. FLOODS is capable of simulating both fabrication processes and resulting device characteristics for Bipolar and Field Effect transistors. In this paper, we implemented several Bipolar Junction Transistor (BJT) simulation problems as web-accessible applets. MINIMOS is a popular simulation program, used both in research and in education, for many advanced device characteristics of MOSFETs. MINIMOS is developed and disseminated by Institute for Microelectronics at Technical University of Vienna [Institute for Microelectronics at Vienna website 2001]. MINIMOS is capable of two-dimensional MOSFET simulation. We have implemented several MOSFET simulation problems as applets in our WebSimMicro.

In order to integrate a legacy native application with a remote web-based user interface, we need to embed the application into our web-extension framework and define a new input and output interface using the reusable components provided by the framework [Tang 2001]. From the development perspective of a web-based simulation package, we need to extend the classes

in the presentation tier and the application tier of the framework, while the logic tier of the framework is used without further extension. The web-based microelectronic device simulator, WebSimMicro, can be easily extended (i) by adding new simulation topics based on already integrated simulation engines such as FLOODS and MINIMOS and (ii) by integrating a new simulation engine into WebSimMicro.

2 Design of Web-based Simulation

WebSimMicro integrates MINIMOS and FLOODS as the back-end simulation applications. Web-based simulators can feature a uniform user interface. A uniform user interface is useful for overcoming the learning curve by the beginning users. The uniform user interface is accomplished by unifying different simulation tools under the same front-end while hiding the software specific syntaxes. For example, in the microelectronic device simulation the user is provided with a uniform interface which always includes device image panel and a parameter design panel with appropriate default values. In addition, this hiding provides an isolation of the back-end resources from the end user. Therefore, the accessing level may be controlled by the framework, which can be useful for the application provider to offer only the necessary services. For the user, on the other hand, this can be advantageous for obtaining sufficient amount of service at a lower cost. However, this partial service access can be a drawback for the experienced and advanced users. The user still needs to resort to the traditional channel, for example, telnet, to get a full access to the simulation application.

The application domain of WebSimMicro is the microelectronic device modeling and simulation. Our framework was developed for extending the legacy simulation applications to

the web environment with reusable designs and components. WebSimMicro follows the framework implementation requirements and provides a web-based simulation environment for the microelectronic devices based on the back-end simulator MINIMOS and FLOODS.

2.1 Our framework

The framework was designed with the purpose of developing a Java web-based simulation application efficiently. Common features of simulations were abstracted and represented in the framework. High efficiency was achieved in developing the web-based simulation packages. The overall architecture of the framework is shown as in Figure 1. The three-tier architecture of our framework consists of reusable software components for presentation, logic and application plug-ins. The multiple tier architecture emphasizes the unique role of individual components by grouping them into a different tier. The presentation tier consists of input/output applet components and login/logout components, which is responsible for the user interface to the entire simulation system. By making most components in this tier configurable, we enable a fast development of user interfaces for different simulation problems or engines. Furthermore, by making them extensible we benefit from the possibility of varying the user interface styles in accordance with different applications which may be incorporated in the system at a later time. The logic tier consists of ready-to-use components and they perform middleware tasks. The Gateway Servlet component (GS) acts as a glue between the presentation tier and the logic tier and is in charge of collecting the user input and returning the application output. The entire logic tier works as a proxy between the remote user and the back-end application. Simulation Controller (SC) is the part of the framework that is in charge of job management. SC acts as the execution agent for the engine container. It arranges the simulation requests from different users

with the scheduling scheme and determines the proper call sequence to the simulation engines. The Engine Container (EC) component is responsible for holding the engines for *one* specific topic and running the engines on demand. There is one according EC for each simulation topic. Each EC is responsible for maintaining the request queue for the topic to which it is assigned. The EC may contain as many engines as necessary, which forms the engine list for all the simulation jobs at the moment. They are called one by one in a defined order and performs distinct tasks within the simulation towards the final completion. All database interactions from these components are managed by the Database Manager (DM) component, which wraps around the database with a connection interface. The DM caches and reuses the connections to the database among different requests to lower the resource cost and to increase the efficiency.

A security model built into our framework makes use of the user/password scheme for the user authentication and activity logging. This scheme is sufficient for most simulation requirements, which may only need to identify the user's working environment and provide the proper input and output for different users. For more critical security requirements, however, a more advanced model will be necessary.

The framework exposes the APIs in the presentation and application tiers for the simulation developer, while keeping the logic layer internal. To develop a web-based simulation for certain domain using our framework, the developer will be interested in the input/output plug-in API's of the presentation layer and the engine plug-in API's of the back-end application layer, which is further discussed in the following sections.

2.2 Input/output plug-in API

The input/output plug-in APIs are provided by two main components in the presentation tier: the Input Applet and the Output Applet. These components take into consideration the common input and output requirements such as the communication to the back-end application components and default user event responses. They can be used either directly by configuration or indirectly by extension for more complicated interface requirements. Since they are Java applets, the configuration is conducted through the HTML tags, by passing pre-defined parameter name/value pairs to the applet. This greatly simplifies the development work still with significant flexibility in the design.

2.3 Engine plug-in API

The new engine is plugged into the system via the engine plug-in API of the application tier. All native applications are abstracted to an Abstract Engine, further classified into Batch Engine component and Interactive Engine component. They serve as the base classes for new engines. To plug in a new simulation engine, a new engine class needs to inherit from either the Batch Engine class or the Interactive Engine class and provide the implementation details required by the base class. Some necessary default implementations are built in the base class, such as the connection to the components in the logic tier and the engine input/output management.

Templates represents an incomplete script for the pre-defined elements and directive tags for the user defined elements. They are also required to be designed while developing a new engine. This template is merged with the user defined elements during the job preparation phase

after the user submission. The added engines together with the template-generated scripts define the job itself.

3 Using the Framework for Development of Web-based Simulation

Applying the framework to a certain application domain is straightforward. Two categories need to be considered: adding new engines and developing new simulation topics.

3.1 Adding Engines

Adding a new engine can be realized by extending the appropriate engine class. If the engine to be added works in a batch mode, the *BatchEngine* class is extended and the abstract methods, such as the path to the simulation program, are implemented. The following code segment illustrates this:

```
public class FloodsBatchEngine extends BatchEngine {
   public FloodsBatchEngine() {}
   public String getProgramPath() {
      return "floods";
   }
.....
}
```

For a new interactive engine, implementation follows the same pattern but a slightly more work is needed. The *InteractiveEngine* class is extended and the methods for initializing and closing the program, as well as the methods for evaluating the user input commands and retrieving the response, are overridden. These methods map the corresponding operations provided by the engine's native API. A typical implementation on the interactive engine is like:

```
public class MatlabInteractivEngine extends InteractivEngine {
```

```
    public MatlabInteractivEngine() {}
    public String getProgramPath() {
       return "matlab";
    }
    public initEngine() { // call native init method }
    public closeEngine() { // call native close method }
    public evaluateCommand() { // call native evaluate method }
    public retrieveResponse() { // call native retrieve method }
.....
}
```

3.2 Adding New Simulation Topics

An example of adding a new simulation topic from the MINIMOS [Selberherr 1980] shall be presented. In this example, we will develop a simple simulation topic which calculates the I-V characteristics of a MOSFET.

3.2.1 User Interface Specification

This step utilizes the presentation tier of the framework and makes a direct use of the Java InputApplet class. By pushing most of the presentation logic to the framework, we only need to specify the UI attributes specific to our problem in the HTML page which will contain the input applet. The specification lies in the following categories:

- ID of the simulation topic. Can be any name that identifies this simulation uniquely.
- Image of the for device overview structure. This image is used for helping users in design of device structure for simulation.
- Device design parameter and default value pairs. Will be used in generating the input file for the simulator.
- Name of the input file template. Used for combining with the user-submitted parameters in order to generate the input file.

- Simulation engine list. Include the name and startup file for each engine used in the simulation. Engine is a wrapper for the native simulation software and must implement Java JNIEngine interface.
- Any other support files which are necessary for the simulation. Include any support scripts that the native simulation software needs.

The fragment of the HTML code is as follows:

```
<applet
  code=websim.in.InputApplet.class name=NMOSIV  width=490 height=390>
  <param name="simuID" value="nmosiv">
  <param name="paraName1" value="Thickness of oxide (cm):">
  <param name="paraValue1" value="5.0E-7">
  ……
  <param name="template" value="nmosiv.inp">
  <param name="deviceImage" value="nmosivview.gif">
  <param name="engine1" value="websim.engine.JNIMinimos">
  <param name="engineStart1" value="nmosiv.inp">
  <param name="engine2" value="websim.engine.JNIMatlab">
  <param name="engineStart2" value="nmosivMatlabTasks">
  <param name="supportFile1" value="nmosivMatlabTasks.m">
  <param name="supportFile2" value="nmosiv.m">
 </applet>
```

After specifying the parameters for the input applet, we finish the input user interface for the simulation as shown in Figure 2.

3.2.2 Prepare Simulation Files

This step does not include an extension from the framework but only deals with the transformation of the script files from the *local* simulation. If the simulation has been done locally, then most of the script files are ready for use in the framework without any modification. The task involves:

- Prepare the device image file specified in the previous step, *User Interface Specification*.

- Prepare the template file specified in the previous step.
- Prepare the startup files for each engine specified in the previous step.

The device image file can be prepared with any graphical software. A template file contains the specified tags which are recognized by the framework. The parameter tags follow the pattern *@paraN@*, where N is a positive integer that denotes the parameter order. A sample template file for the MINIMOS engine used is shown below:

```
*  MINIMOS input file "nmosiv.inp "
*  MOSFET - 0.25um NMOS IV

DEVICE   CHANNEL=N GATE=NPOLY TINS=@para1@ W=@para2@ L=@para3@

PROFILE  NB=@para4@  ELEM=AS  DOSE=@para5@  AKEV=@para6@
PROFILE  ELEM=AS  DOSE=@para7@  AKEV=@para8@  XOFF=120E-7
+        TIME=@para9@  TEMP=@para10@
.........
```

In this example, the startup file for the MINIMOS engine is a file generated automatically when the WebSimMicro merges the template file with the user submitted parameter values. Therefore, we just direct the MINIMOS engine to use *nmosiv.inp* as the startup file in the HTML code, as shown in *User Interface Specification*. If we want the simulation result plotted by Matlab in addition to the raw data, we just make use of Matlab script locally and specify the Matlab startup script name in the HTML code as well. The HTML code in *User Interface Specification* expressed this.

3.2.3 Output Interface Specification

The last step is to specify an output interface. It does not need to be implemented if we do not want a graphical output. A raw data output, such as packing and downloading the data, is built into the framework.

The output interface specification makes use of the class OutputApplet in the framework (extend this class when more specific output is required). The tasks involved are quite similar to the input interface specification that is in the HTML style (it is written as JSP file to utilize the security model provide in the framework and to reduce the implementation chores):

- Specify the user identity and job identity parameters. This is shown in the following code segments (lines 3 and 4). By using the JSP, the user name and job ID are filled automatically and passed in the query strings to the JSP.
- Specify the desired output graph. Use the name pattern *outputN* where N is counted incrementally starting from 1. In this example, we only have one I-V output.
- Specify the simulation ID. This should be identical to the ID specified in the input interface.

Here is the segment of the JSP file for the output, the UI is shown in Figure 3:

```
<applet
   code=websim.out.OutputApplet.class name=OutputApplet width=400 height=330>
   <param name="user" value="<%= username %>">
   <param name="id" value="<%= jobid %>">
   <param name="output1" value="IV" >
   <param name="simuID" value="nmosiv" >
  </applet>
```

The above three steps have specified all the necessary elements, required by the framework. The required elements are the input UI in the presentation tier, the simulation

directives in the back-end application tier, and the output UI in the presentation tier. The simulation logic in middle tier is handled entirely by the framework with no need for additional work. For a final view of the simulation example we have just built, see Figure 4.

4 NMOS I-V Characteristic Simulation

A sample NMOS I-V characteristic simulation was implemented according to the phases described in section 2 above. It consists of the input and output interfaces shown in Figure 2 and Figure 3, plus the utilities uniformly provided by the logic tier of the framework (Figure 4), all of which forms a web-based simulation on the specified topic and meets our initial requirements. The input applet provides the interface for working with the NMOS device structure and the process parameters, specified by the end user with assistance of the device overview image. The parameters such as the channel dimension and doping are presented with their default values. In addition, the syntax of the engine, MINIMOS in this example, is hidden from the end user so that the user's attention may be focused mainly on the device design itself.

After submitting the parameters, the simulation script is generated and executed. User is notified of any invalid or physically meaningless parameters submitted by the error output file, and an outline I-V plot is provided for an easy view of the simulation output. For a more detailed and complete analysis of the simulation results, all output files are made available to the user.

5 Discussion

In this paper, we applied our previously reported framework to the microelectronic device simulation application and built a simple NMOS simulation. For the sample we just provided, we

made use of the reusable components provided by the framework either by extension or by configuration. Quality of the application may be improved by using another feature from the framework: reuse by extension. For example, in the example above, the design interface, which is implemented by configuring the input applet component provided by the framework, only provides a static device image to help the device design by the user. More powerful and dynamic assistance can be achieved by adding a user interaction with the device image, therefore reducing the design difficulty for beginners. The framework does not provide a ready-to-use implementation for this kind of dynamic input, simply because such interactions vary widely from problem to problem and there is no easy way to generalize. However, this function can be implemented in individual simulation problems by extending the input applet class. Actual implementation will be similar to those in other client driven UI designs, provided by other online simulation services [Wie and Yuan 2000; Educational Java Applet website 2001].

6 Summary

A web-extension framework was used to implement a web-based simulation service in microelectronic devices. This approach saves the development time and improves the quality and efficiency of the developed product. By incorporating the Java technology, WWW, and Object-Oriented Design, the framework consists of several reusable components and a reusable architectural design for the web-extension of a legacy simulation program. It allows a remote access to the powerful microelectronic engines such as FLOODS and MINIMOS. Finally, we have set up a website that provides the remote simulation in our microelectronic device education using the framework [WebSimMicro website 2001].

7 Acknowledgements

We would like to acknowledge the financial support by National Science Foundations through the grant numbers DUE9752316, DUE9950794 and a partial support from the University of Virginia through a subcontract number 526007.

References

AKARSU, E., FOX, G., HAUPT, T., KALINICHENKO, A., KIM, K. S., SHEETHAALNATH, P., AND YOUN, C. H. 1999. Using gateway system to provide a desktop access to high performance computational resources. *International Workshop on Java for Parallel and Distributed Computing* (in conjunction with IPPS/SPDP '99). San Juan, Puerto Rico.

Educational Java Applet Service website. 2001. http://jas.eng.buffalo.edu.

FLOODS website. 2001. http://www.tec.ufl.edu/~flooxs/.

HAUPT, T., AKARSU, E., FOX, G., KALINICHENKO, A., KIM, K. S., SHEETHALNATH, P., YOUN, C. H. 1999. The Gateway system: uniform web based access to remote resources. *ACM 1999 Java Grande Conference*. San Francisco, CA, June 12-14.

Institute for Microelectronics at Technical University of Vienna website. 2001. http://www.iue.tuwien.ac.at/.

KAPADIA, N. H., FIGUEIREDO, R. J., FORTES, J. A. B. 2000. PUNCH: web portal for running tools. *IEEE Micro. 20*, 3. 38-47.

KAPADIA, N. H., FORTES, J. A. B., AND LUNDSTROM, M. S. 2000. The Purdue University network-computing hubs: running unmodified simulation tools via the WWW. *ACM Transactions on Modeling and Computer Simulation 10*, 1 (Jan.), 39-57.

KAPADIA, N. H., FORTES, J. A. B., AND LUNDSTROM, M. S. 1997. The semiconductor simulation hub: a network-based microelectronics simulation laboratory. *12th Biennial University Government Industry Microelectronics Symposium* (Jul.). Rochester, New York.

Landscape Management System (LMS) website. 2001. http://www.npac.syr.edu/users/haupt/WebFlow/projects/LMS/Snapshots/index.html.

LIANG, M., AND LAW, M. 1994. An object-oriented approach to device simulation - FLOODS. *IEEE Transactions on CAD, 13*, 10. 1235-1240.

PREMCHANDRAN, G., AND FURMANSKI, W. 1997. WebFlow: a visual programming paradigm for web java-based coarse grain distributed computing. *Concurrency: Practice and Experience, 9*, 6. 555-577.

RICHARDSON, T., STAFFORD-FRASER, Q., WOOD, K. AND HOPPER, A. 1998. Virtual network computing. *IEEE Internet Computing, 2*, 1. 33-38.

ROMBERG, M. 1999. The UNICORE architecture: seamless access to distributed resources. *Proceedings of the eighth IEEE International Symposium on High Performance Distributed Computing* (HPDC-8) IEEE Computer Society (Aug.). Los Alamitos, CA. 287-293.

SELBERHERR, S., SCHUTZ, A., AND POTZL, H. W. 1980. MINIMOS - a two-dimensional MOS transistor analyzer. *IEEE Transactions on Electron Devices, 27*, 8. 1540-1550.

Stanford University TCAD website. 2001. http://www-tcad.stanford.edu

TANG, T., AND WIE, C. R. 2001. A Java extension framework for web-based simulation. Submitted to *ACM Transactions on Modeling and Computer Simulation*.

WebSimMicro website. 2001. http://jas7.eng.buffalo.edu.

WIE, C. R., YUAN Z., LIU N. 2000. Educational Java applets: object-oriented development for reuse and maintenance. *International Conference on Simulation and Multimedia in Engineering Education* (Jan.). 59-64.

X Consortium, Inc. 1996. X window system version 11 release 6.3: release notes.

Figure Captions

Figure 1. Web-based simulation framework overview

Figure 2. Screenshot of input UI

Figure 3. Screen shot of output UI

Figure 4. Screen shot of WebSimMicro

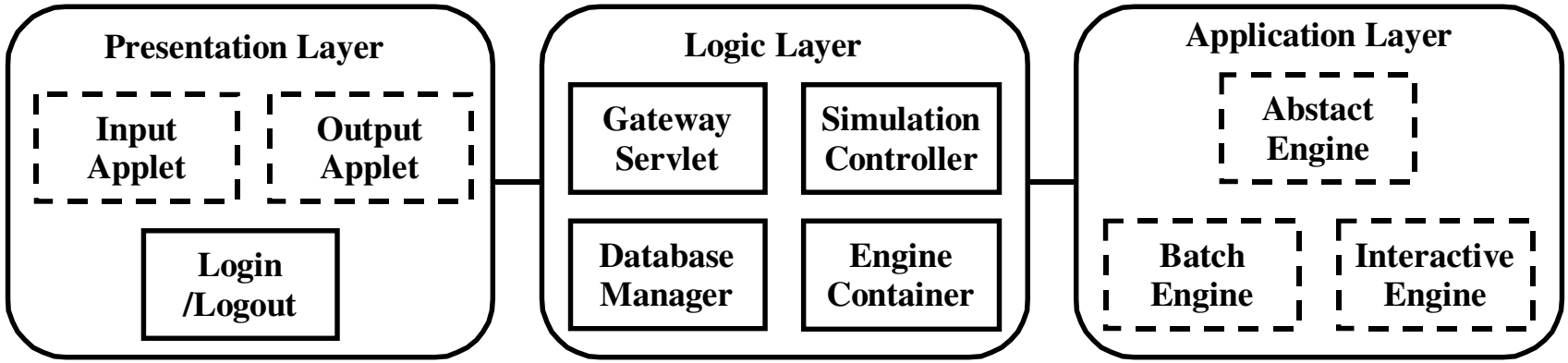
Presentation Layer
Input Applet
Output Applet
Login /Logout
Logic Layer
Gateway Servlet
Simulation Controller
Database Manager
Engine Container
Application Layer
Abstact Engine
Batch Engine
Interactive Engine

Figure 1

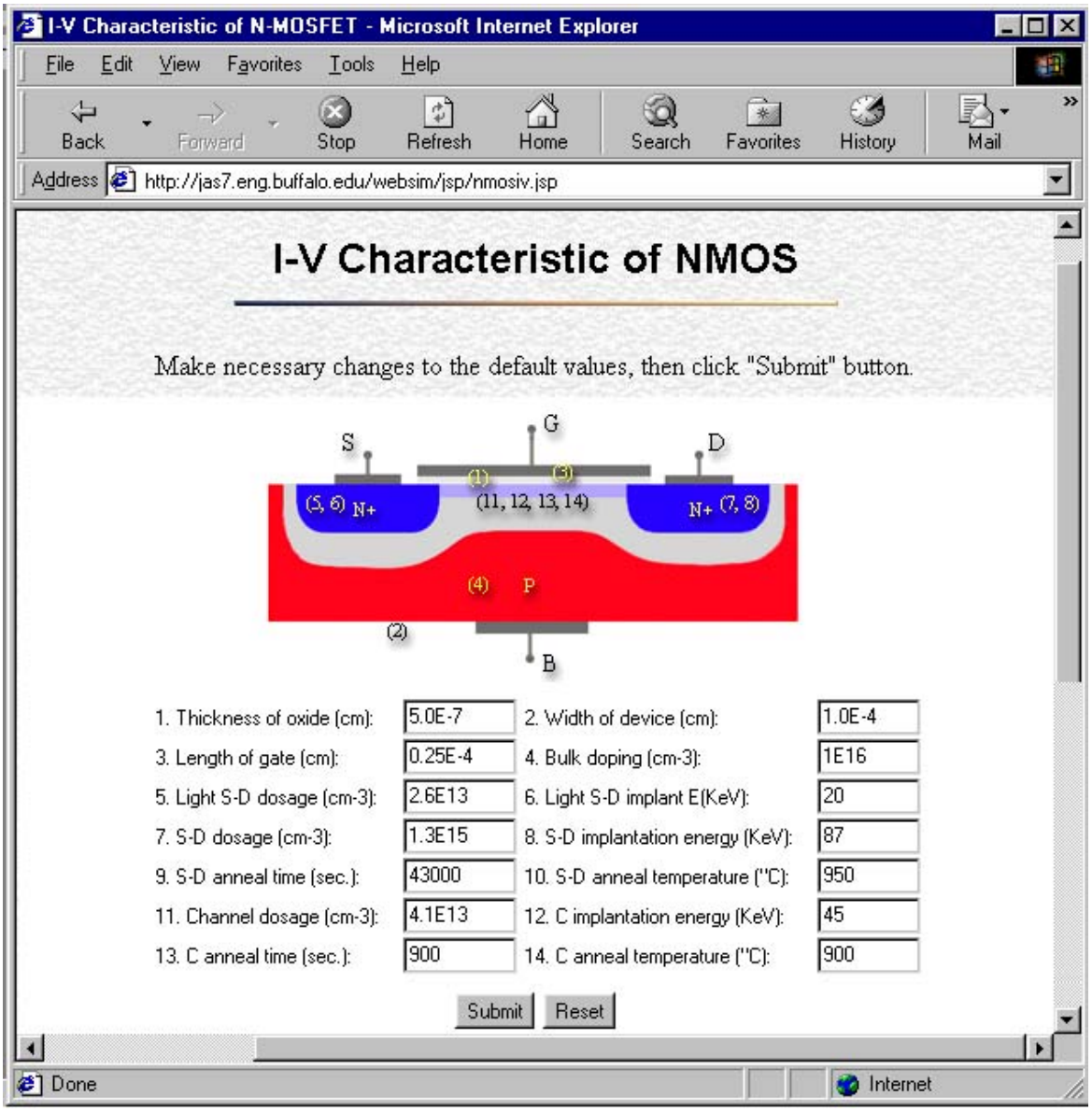
I-V Characteristic of N-MOSFET - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail
Address http://jas7.eng.buffalo.edu/websim/jsp/nmosiv.jsp
I-V Characteristic of NMOS
Make necessary changes to the default values, then click "Submit" button.
S
G
D
(1)
(3)
(5, 6) N+
(11, 12, 13, 14)
N+ (7, 8)
(4) P
(2)
B
1. Thickness of oxide (cm): 5.0E-7
2. Width of device (cm): 1.0E-4
3. Length of gate (cm): 0.25E-4
4. Bulk doping (cm-3): 1E16
5. Light S-D dosage (cm-3): 2.6E13
6. Light S-D implant E(KeV): 20
7. S-D dosage (cm-3): 1.3E15
8. S-D implantation energy (KeV): 87
9. S-D anneal time (sec.): 43000
10. S-D anneal temperature (°C): 950
11. Channel dosage (cm-3): 4.1E13
12. C implantation energy (KeV): 45
13. C anneal time (sec.): 900
14. C anneal temperature (°C): 900
Submit
Reset
Done
Internet

Figure 2

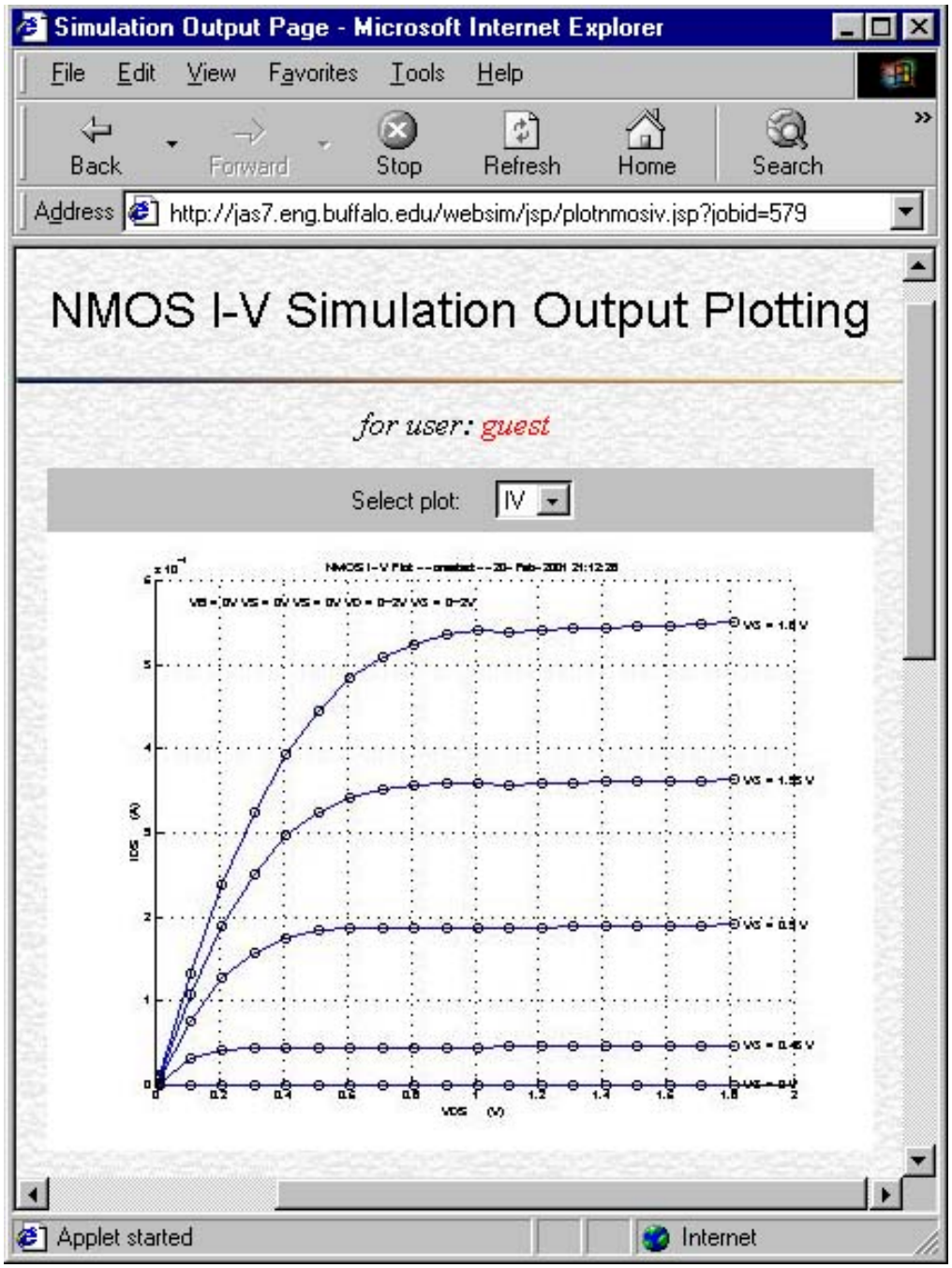
Simulation Output Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search
Address http://jas7.eng.buffalo.edu/websim/jsp/plotnmosiv.jsp?jobid=579
NMOS I-V Simulation Output Plotting
for user: guest
Select plot: IV
Applet started
Internet

Figure 3

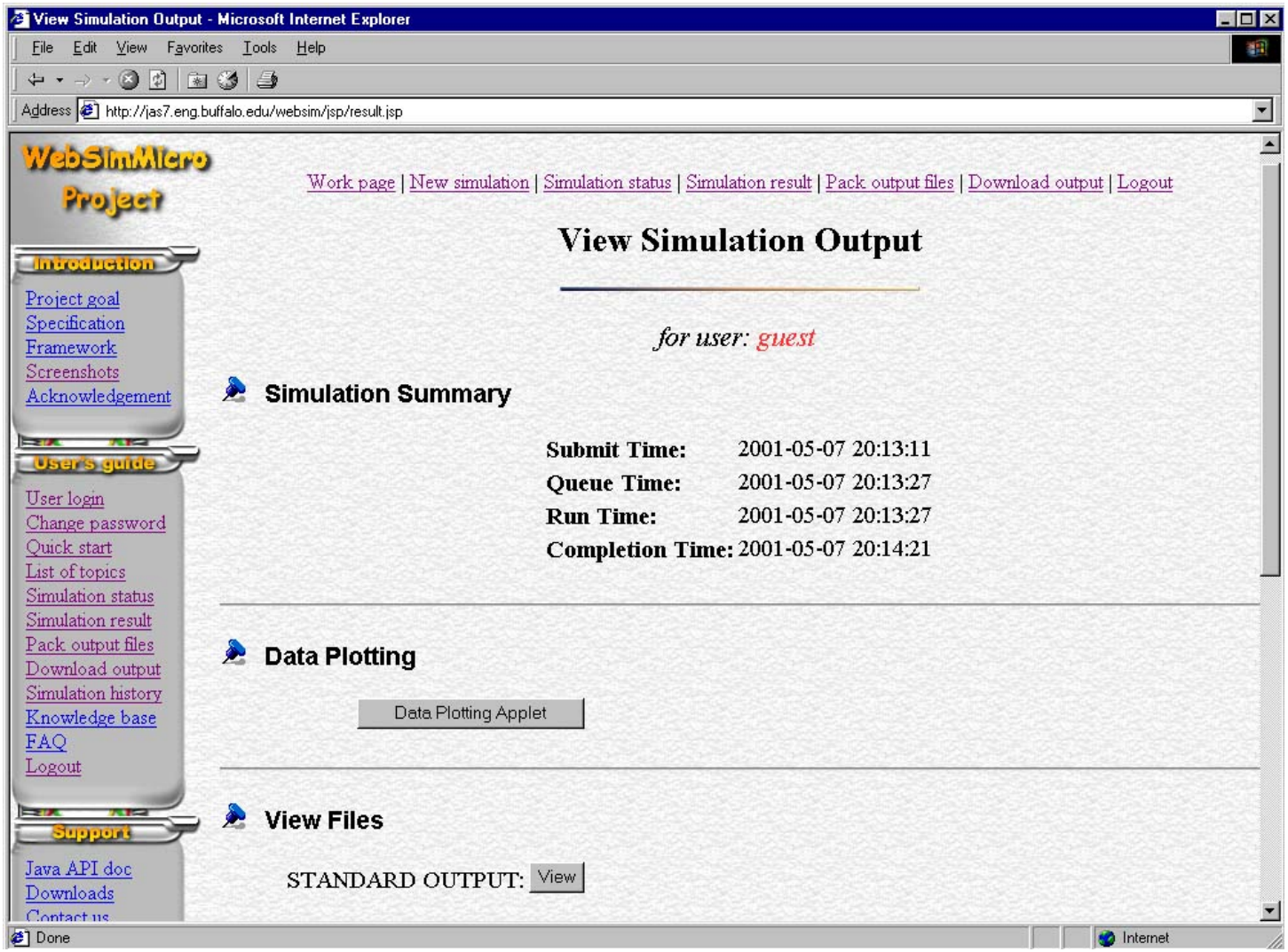
View Simulation Output - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://jas7.eng.buffalo.edu/websim/jsp/result.jsp
WebSimMicro Project
Work page | New simulation | Simulation status | Simulation result | Pack output files | Download output | Logout
View Simulation Output
for user: guest
Introduction
Project goal
Specification
Framework
Screenshots
Acknowledgement
User's guide
User login
Change password
Quick start
List of topics
Simulation status
Simulation result
Pack output files
Download output
Simulation history
Knowledge base
FAQ
Logout
Support
Java API doc
Downloads
Simulation Summary
Submit Time: 2001-05-07 20:13:11
Queue Time: 2001-05-07 20:13:27
Run Time: 2001-05-07 20:13:27
Completion Time: 2001-05-07 20:14:21
Data Plotting
Data Plotting Applet
View Files
STANDARD OUTPUT: View
Done
Internet

Figure 4