A Framework for Distributed Web-based Microsystem Design

Debashis Saha Anantha P. Chandrakasan
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Abstract

The increasing complexity of microsystem design mandates a distributed and collaborative design environment. The World Wide Web serves as a desirable platform for distributed access to information and design tools. In this paper we present a framework that enables distributed Web based CAD, in which Web based design tools can efficiently utilize the capabilities of existing design tools on the Web to build hierarchical Web tools.

1 Introduction

The emergence of *"system-on-a-chip"* with more than a billion transistors on a single chip calls for a distributed and collaborative design framework which would facilitate easy and fast information flow. Designers are increasingly seeking for tools and expertise that allow rapid exploration of the design space irrespective of the geographical or physical availability of the design tools. VLSI design could be facilitated by giving the designer access to design tools and different library modules and their specification in a distributed format from a simple user interface. All these requirements motivate us to build a distributed framework where designers would be able to access and utilize diverse resources from the desktop. The framework should allow designs tools to communicate with other in a standard integrated way and allow for other tools to utilize the capabilities of existing tools.

The advent of Internet has opened new vistas in the areas of distributed design. The World Wide Web has emerged as the most desirable platform for distributed access to information. In the last few years the Web has also emerged as a strong platform for remote access to tools using a simple Web browser [1, 2, 3]. The limitations of only server side computation of the Web using form based CGI programming has been alleviated with the emergence of a platform independent programming language Java. The ultimate goal of Web based CAD is to have the designer have access to tools, irrespective of the tool's location and platform. In such a Web-based framework the tools should have the capability to exchange information efficiently such that the designer could make effective use of the wide variety of available tools.

2 Related Work

The popularity of the internet and the Web has lead to many distributed design efforts. In [4], Bentz et al. have proposed an information based design environment. They describe an environment which helps users collect and manage information in a uniform fashion, independent of the abstraction levels or implementation platforms. In [1], Lidsky and Rabaey present a World Wide Web based prototype tool, PowerPlay which helps in system level exploration of power consumption. The WELD project [5], aims to construct a distributed CAD design environment enabling Internet wide IC design for the US electronic industry. At the 1996 Design Automation Conference (DAC), the WELD group demonstrated a prototype WWW-based design environment. There have been efforts to provide Web-based Interfaces to executable CAD/CAM softwares. Links to different Web based tools could be found at [2]. Several CAD/CAM tools like SUPREM, SAMPLE-2D (Simulation And Modeling of Profiles in Lithography and Etching) and SPLAT (Simulation of Projection Lens Aberrations via TCCs) are available on the Web [3].

The explosive growth of the World Wide Web has lead to a need for support for distributed and scalable applications. WebOS [7] (Software Support for Scalable Internet Services) focuses on the infrastructure necessary to build distributed Web application. In [8], Lavana et al. use executable directed hypergraphs to describe collaborative design activities on the internet. Although there are numerous instances to design tools being available over the Web there have been limited attention to design a framework which would allow us to utilize the different tools over the Web. In this paper we show how a distributed framework could be built over the Web which would support efficient communication and data exchange between different Web based tools.

0-8186-7967-0/97 $10.00

3 Technologies For Web based CAD

3.1 HTTP and CGI

Common gateway Interface (CGI) has enabled a new class of specialized Web applications to exploit Internet's interactive nature. CGI programs are usually referred as scripts which run on the server machine and produce the output (usually HTML output) to be displayed on the clients browser. The Hypertext Transfer Protocol (HTTP) and CGI are the protocols that govern interactions between the client, server and script. Under this protocol, a client sends a *request* to the server to retrieve a document or execute a script. The server complies and sends back a response containing the requested data.

Currently, HTML forms allow the producer of a form to request information from the user reading the form. The traditional forms encode the user input data with encryption "application/x-www-form-urlencoded" type. A more versatile and useful encoding type "multipart/form-data" has been recently introduced and this can handle arbitrary file uploads. We propose to use the POST method and ENCTYPE multipart/form-data as standards for Web based CAD applications. This is because the POST method and ENCTYPE multipart/form-data is capable of handling any arbitrary data transfer between clients and servers. The POST method uses the method body to send large data, typical of a CAD environment, to the server. Moreover, after studying a large number of input and output specification of various CAD tools we found out that it is sufficient to specify the inputs and outputs of tools with only two tags. These two are $< Parameter >$ and $< Files >$. The semantics of multipart/form-data is best suited to capture both of these tags. It is useful to keep these tags as simple as possible because that would facilitate communications between different heterogeneous.

3.2 Java based Client-side processing

CGI programs and forms lacked the interactivity and complex user interface. No designer would be satisfied with the limited display of a CGI based program with a HTML output. Designers need to modify and view different designs in real time. Java allows us to do complex client-side processing. Therefore applications which do not need extensive computational power could be handled well with Java based interface. But leaving all the work to Java based client-side processing would hinder our attempts to do CAD with a simple desktop. Therefore what we need is a partitioning of an application into Java based client processing and traditional remote computation.

3.3 CORBA

The Common Object Request Broker Architecture (CORBA) [12] is a standard for distributed computing that has gained widespread acceptance. CORBA provides an infrastructure which enables invocations of operations on objects located anywhere in the network as if they were local to the application using them. Together with the Interface Definition Language (IDL) and Application Programming Interfaces (API) that were defined by the Object Management Group (OMG), CORBA 2.0 enables client/server object interaction and interoperability by specifying how Object Request Brokers (ORB) from different vendors can interoperate.

3.4 Object-Web Architecture

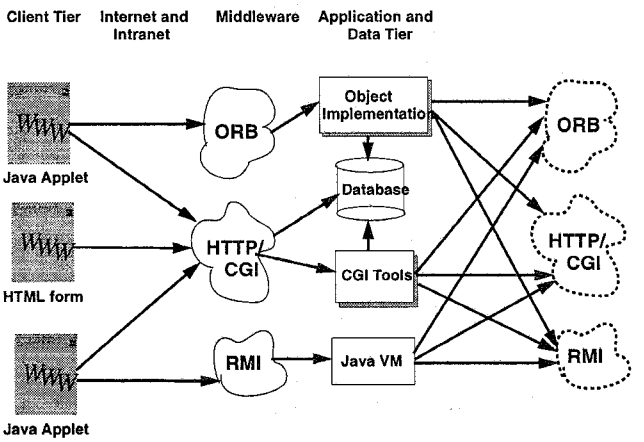

Figure 1. Object-Web Architecture

With the emergence and acceptance of industry-defined standards such as CORBA, Web standards such as HTTP and Java and the world-wide distribution medium of the internet, we can build a open, interoperable, reliable, scalable Object-Web computing architecture.

In the Object-Web architecture (Figure 1), the client tier is is built on Web browsers to provide a standard graphical interface, through which users can access tools and information via the internet and the intranets. Lightweight Java client applications can run in Java-enabled browsers. Application can also be accessed using the HTML form based interface. The application and data tier consists of different application tools and databases distributed across the network. The tools are heterogeneous and run on diverse platforms. The application tools have well defined interfaces and they cooperate to build distributed applications through the middleware services. The middleware services such as CORBA ORBs, HTTP Common Gateway Interfaces or the Java Remote Method Invocation (RMI) connect client users to resources and applications in the application tier. Client

and server objects alike can send messages to other objects throughout the network using ORBs or invoke remote applications using CGI or RMI. Web servers also provide access to HTML documents and Java applets. Overall, the Object-Web achitecture is flexible and enables tools and applications to effectively communicate with each other to build a distributed framework for Web based CAD.

4 Hierarchical Web based CAD

Once we have many tools on the Web, it would be very useful to utilize the tools available on the Web to build new tools. A new tool could remotely execute different tools and produce its necessary outputs. The following example demonstrates the usefulness of such a hierarchical CAD framework.

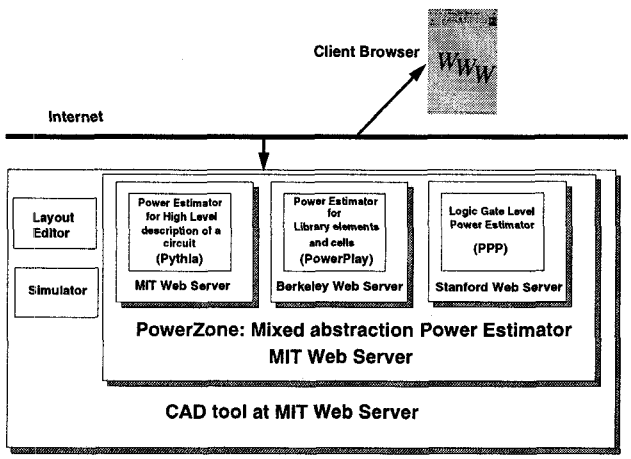

Figure 2. Hierarchical Web based design tools

Early exploration of power dissipation of circuits is becoming important in VLSI design. With a multitude of different technologies and power estimation tools available as depicted in Figure 2, designers would like to estimate the power dissipation for different architectures and technologies. Designers would also like to specify their design in mixed levels of abstraction. For example the designer may want to specify the design as a combination of high level description (C or VHDL program), library modules (RAM, ROM, adders, multipliers) and low-level schematics (transistors and gates). Several Web based tools exist to estimate power dissipation at different levels of abstraction (for e.g., Pytha - Verilog based Power Estimation Tool at MIT [9], PowerPlay of UC, Berkeley [1], PPP - Power Estimation and Synthesis of Low-Power Circuits at Stanford [6]). Then a designer at MIT builds a tool *PowerZone* which given a mixed level circuit design (specified in various levels of abstraction) would generate the estimated power dissipation of that circuit. PowerZone calls appropriate tools on the Web for the specific technology for appropriate abstraction of the design and generates the total power dissipation of the circuit. Once PowerZone is available on the Web, it could be used by other integrated CAD environments to estimate the power dissipation. Therefore we have a hierarchy of tools which encapsulates other tools on the Web to build a new tool on the Web. Such a hierarchical framework on the Web facilitates distributed design and efficient exchange of information. Figure 2 shows the distributed hierarchy of tools.

5 Infrastructure for Hierarchical Web based CAD

We would like to present a formalism of applications which would help us to integrate tools in the distributed framework. Any application can be sliced into different independent applications based on its inputs and outputs. Therefore any application A can be sliced into different sub-application in the form $< A_1, A_2, ..., A_n >$, where each A_i can be visualized in the form of $< Input_i, Process_i, Output_i >$, where $Input_i$ is the set of inputs to sub-application A_i and $Output_i$ is the set of outputs of A_i. $Process_i$ is the set of steps (the algorithm and the program) used to process $Input_i$ to produce $Output_i$. The slicing is based on the interrelationship between the Output sets and the Input sets. The $Input_j$ sets depend only on some $Output_k$ sets such that $k < j$.

Let us consider the following CAD tool which allows the user to draw the schematic of the circuit, then extracts the schematic to produces a spice netlist and then simulates the circuit and produces an output plot of the simulation result. Such an application can be thought as a collection of applications like $<$ *Schematic Editor, Spice extractor, Simulator, Graph Utility* $>$. The Schematic Editor uses GUI to enter a schematic and produces a representation of the schematic. The Spice Extractor takes the output of the Schematic Editor and produces a spice netlist. The Simulator takes input the Spice netlist and a set of input vectors and produces a data set of the output. The Graph Utility takes in a data set and the output format (Postscript, Jpeg or Giff) and produces an image file. We note each of these tools are independent and can produce their own output. Our main application acts as an agent to properly invoke each tool with the output of the previous tool.

The input and output sets are specified in the form described previously, which are sets of the elements of the form, $< Parameter \quad id >$ and $< File \quad file - id >$. As it was argued any application's input and output can be specified with the help of these two elements. Each Sub-application A_i is independent in the sense no other infor-

mation other than its Input set is required to execute the application. The division is also more of a functional division, where each sub-application performs a specific function. Such a standard description of applications helps us to standardize the specification of an application in a distributed environment. Now, each of the independent sub-applications could be developed separately in a distributed manner and the clear specification of the applications in terms of inputs and outputs provides a first step to easily integrate them in a distributed framework. In a later section we would build on this concept to describe a hierarchical Web based CAD framework.

We can easily see in a distributed environment such an application A could be a design agent and the sub-applications are distributed point tools. The design agent interacts with the designer, invoking tools and accessing data on the designer's behalf [4]. Such a design agent could be a static one in which the agent is responsible for interpretation of outputs and appropriately generating the inputs for the chain of sub-applications. Or the design agent could be a dynamic one which dynamically configures itself (in terms of the sub applications that constitute itself).

5.1 Web based Point tools

Any point tool or application on the Web can be specified as, $< Input_{tool}, Output_{tool} >$, where $Input_{tool}$ is the input set and $Output_{tool}$ is the output set expressed in the standard form as described in the previous section. We call this set **Basic Interface Specification (BIS)**. Now the $Input_{tool}$ can be obtained by a CGI form or a Java based GUI. The input is then given to the tool to be processed and produce the $Output_{tool}$. The tool itself can be a CGI program, linked to a HTTP address or the same Java applet or a different Java applet, which processes the input.

5.2 CGI back-end Tools

Let us consider that our back-end processing is done by a CGI program which is bound to a HTTP address like, "http://apsara.mit.edu/cgi-bin/pythia/pythia.pl". By HTTP address bound tools we mean that if a socket connection is made to the web server and the input data as specified in the BIS of the tool is POSTed as multipart/form-data to that address, the HTTP server executes the tool and writes back the output to the socket as specified in the BIS. The input to this CGI program can be provided by a Java applet or a HTML form. Both the input agents or processes provide the data in the multipart/form-data and they use the POST method to talk to the HTTP server, which would execute the CGI program. All CGI back-end application tools should be capable of decoding this multipart/form-data process the data and produce the output. In addition to the BIS we to add another parameter to the input set of the tool : $< Parameter \quad OutputType >$. The **OutputType** parameter can take any of the following values: **HtmlType**, **BasicOutputType** or the **URLOutputType**.

If the tool was used as an end tool, that is the display is intended for the user it may have the **OutputType** parameter set as **HtmlType** in which case the tool produces a dynamic customized HTML page of the output. In addition to that every tool in our framework should be able to produce the output in multipart/form-data when the **OutputType** parameter is set as **BasicOutputType**. This is necessary because we would like to build a hierarchy of tools using this existing tool. At least there should be the potential to integrate this tool with other tools on the Web. The OutputType of **URLOutputType** produces a URL which contains the output of the tool. This is necessary because applets may not have the capability of parsing a HTML file, but it can prompt the browser to load any URL which parses the HTML file.

Therefore the HTTP address bound tools should be capable of decoding the input multipart/form-data of the input set of the BIS to its internal representation of data. It has the capability to produce a custom HTML output page or the output set of the BIS as multipart/form-data or a URL output according to the value of the OutputType parameter in its BIS. In its BasicOutput the tool could have any headers which the HTTP server generates. In addition to those it also generates a header containing the information of the output generated, which at present contains just three tags, BasicOutputType (which specifies the address of the program generating the output) and Content-length (which specifies the content length of the multipart data in bytes) and Content-type (which tells that the data type is multipart/form-data).

5.3 Hierarchical Web Tools

In this section we would show how the point tools described in the previous sections could be used to build a hierarchical Web based CAD framework. Given the capabilities of all the HTTP address bound tools, we would show how we can build tools which would utilize the capabilities of the existing tools and serve as a new tool on the framework.

Let us have a tool X on the Web which is bound to the HTTP address **http://x**. And let us have another tool Y on the Web on the address **http://y**. Given these two tools and their BIS we can easily construct another tool Z bound to **http://z**. Tool Z gets its own input according to its BIS, then process the data and produce an appropriate BIS specified multipart/form-data. The series of action are described in details below and is shown in Figure 3.

1. We have a CGI program Z bound to http://z which

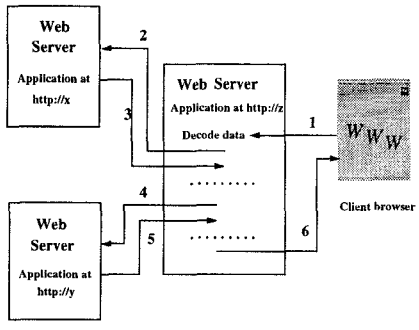

Figure 3. Data flow in the Hierarchical tools

gets its own input from an applet or a HTML form.

2. It decodes the input which is in form of multipart-form data and is POSTed to the Web server. It processes the input and makes a socket connection to the Web server at http://x. It packs all the input data specified in the BIS of X and writes it to the socket and waits for the output.

3. The server at http://x executes X. X produces the Output as specified by the output set, and the server writes back the data to the socket.

4. Z then decodes X's output, processes the data and then makes a similar call to http://y, sends the inputs as specified in the BIS of Y and waits for the reply.

5. The server at http://y executes Y. Y produces the Output as specified by the output set, and the server writes back the data to the socket.

6. Z produces its own output as specified by its BIS and sends it to the client browser.

It can be seen the only thing that needs to be known is the BIS of the individual tools. And since all other protocols are well defined communication in this distributed framework can be done in a uniform and unambiguous way. Now Z could be called from another CGI tool, therefore we can build a hierarchy of tools on top of the existing ones. As we speculate an enormous increase in the Web access to different CAD tools in the coming years such a distributed framework could help utilizing different existing tools on the Web. The framework provides the protocols and standards needed to effectively do distributed design on the Web.

5.4 Java and Web Tools

As we have mentioned before Java applets could be used instead of forms to enter the input data of a tool. Java applets could also act as agents which would help use the capabilities of different Web tools. The present day security restrictions do not allow applets to open a socket with any machine other than the base machine (from which the classes were downloaded) or perform a read/write on local file system. Therefore to efficiently utilize any tools anywhere on the Web it is necessary to have proxy servers linked with the server with which the applet communicates. We use the same encryption of multipart/form-data for the applets to communicate with tools and vice versa, which provides a standard packing of data for any Web based applications. The applets POST any data to the server tool as multipart/form-data and the server tools write back the results and data to the applets again as multipart/form-data. We present a prototype of such communications in the Section 6.

6 Example of Web based Hierarchical Tools

We present two prototypes which demonstrate the ideas discussed for the Web based framework.

Example 1: In the first example, we developed two point tools:

1. **Pythia:** This is a power estimation tool, which takes a verilog netlist and other parameters (Technology type, Cell models) and generates energy and power information [9] and is bound to the URL *http://apsara.mit.edu/cgi-bin/pythia/pythia.pl*. It also generates a data set of the current drawn and the power consumed at different time intervals. In the Output specification in the BIS of Pythia, we have *pythiafile*, which is a file containing the data sets of the current drawn versus time as a list of X and Y co-ordinates.

   ```
   Output: <File pythiafile>, ...
   ```

2. **Graph Plotting Utility:** *http://apsara.mit.edu/cgi-bin/graph/drawgraph.pl*. This tool takes in a file of X,Y data sets and produces a image of the plotted graph as a Jpeg image. In the BIS of the tool we have *datafile* which is a file containing the data set and *graphfile* is the Jpeg image file.

   ```
   Input: <File datafile>
   Output: <File graphfile>
   ```

   The graph plotting can be accessed at: *http://apsara.mit.edu/graph-images/drawgraph.html*

Now that we have these tools on the Web we make a new tool **HierDem**, which takes in a verilog netlist and simulates the current drawn by the circuit. It produces a plot of the current versus time. This tool again is an independent tool with its own BIS, but internally it

calls Pythia to get the data set of the current drawn by the circuit and then calls the Graph Utility to plot the data set. This new hierarchical tool can be accessed at: *http://apsara.mit.edu/demo1.html* and is bound to the URL *http://apsara.mit.edu/cgi-bin/demo1/demo1v2.pl*. The flow of information is similar to that depicted in Figure 3 where X is Pythia, Y is Graph Utility and Z is the hierarchical web tool HierDem.

Example 2: In this example we describe WebTop, a Java block/schematic editor WebTop [10]. WebTop reuses some code of a Digital Simulator Java applet [11] written by Iwan van Rienen for the graphical user interface. WebTop also supports distributed cell access. Cells could be stored in different Web servers and loaded as URLs.

WebTop interfaces to other Web tools and demonstrates how an Java applet tool could utilize another CGI based tool on the Web. We have developed a CGI based point tool **WebSpice** at *http://apsara.mit.edu/cgi-bin/spice/spice.pl*, which provides Web access to the circuit simulator Hspice. WebSpice is an independent web tool and can be accessed through the HTML form at: *http://apsara.mit.edu/spice.html*. Using WebTop, the user can extract a design schematic into a SPICE netlist. WebTop, then submits the netlist to WebSpice as multipart/form-data conforming to the BIS of WebSpice. WebSpice simulates the spice netlist and sends back the results of simulation to the applet as URL. The applet then displays the resulting URL in the browser. WebTop also interfaces to the Web-tool Pythia [9] in a similar fashion. The inter-tool communication is generic and can be used for any tool accessible over the internet. Each server side tool could itself call other Web tools in our hierarchical framework. Figure 4 shows a snap shot of WebTop, its distributed cell access and tool integration architecture.

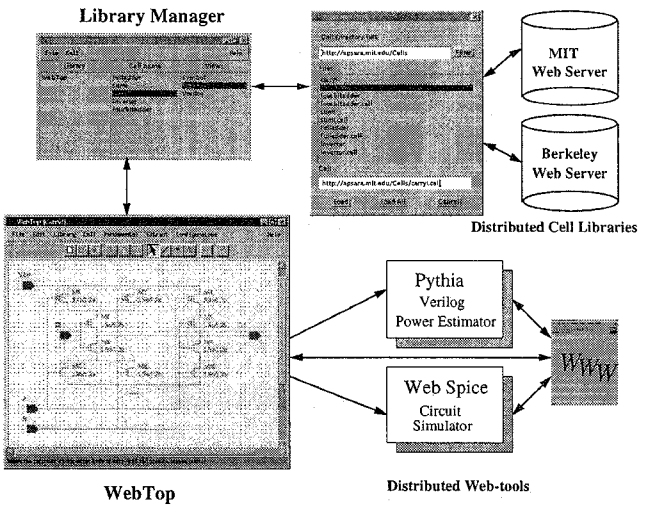

Figure 4. WebTop: A schematic/block editor

7 Conclusions

Once we have a multitude of tools over the Web, there will be an increasing need for sophisticated design agent which will allow us to build new tools. This requires for standardization of communication and data exchange protocols between the different tools. Although the Web delivery provides some standardization as regards the protocols of communications, the inputs and results of the tools should be specified in a more formal and rigid manner to help information interchange. Also, the issues of security, standarization and bandwidth limitations should be addressed adequately. We presented a standard slicing of an application with respect to its inputs and outputs. We then presented a framework for Web based CAD with a standard way of information interchange between the tools using CGI as an effective middleware to build hierarchical web tools.

Acknowledgments

This work was supported by the Defense Advanced Research Projects Agency (DARPA Contract DABT63-95-C-0088). The authors would like to thank David Liebson for his assistance in modification of the DigSim applet code.

References

[1] D. Lidsky and R. J. M. Early power exploration - a world wide web application. *Proc. Design Automation Conf, Las Vegas, NV*, June 1996.

[2] Microsystem design test bed, http://web_nt.sainc.com/arpa/msdproject/testbed.htm

[3] WebCAD: Web-based interfaces to CAD/CAM softwares http://www-bsac.eecs.berkeley.edu/ jjudy/ntu/simulations/

[4] O. Bentz, D. Lidsky, and J. M. Rabaey. Information-based design environment. *IEEE VLSI Signal Processing VIII*, pages 237–246, Nov 1995.

[5] The WELD Project, http://www-cad.eecs.berkeley.edu/Respep/Research/weld/

[6] PPP, http://akebono.stanford.edu/users/PPP/

[7] WebOS: http://now.CS.Berkeley.edu/WebOS/

[8] H. Lavana, K. Amit, F. Brglez, and K. Kozminski. Executable workflows: A paradigm for collaborative design on the internet. *Proceedings of the Design Automation Conference*, June 1997.

[9] Pythia, A Verilog based power estimation tool, http://apsara.mit.edu/pythia-doc/pythia.html

[10] WebTop: http://apsara.mit.edu/WebTop/

[11] Digsim: Digital Simulator http://www.lookup.com/homepages/96457/digsim/index.html

[12] OMG. *The Common Object Request Broker: Architecture and Specifications*. John Wiley and Sons, New York, 1993.