Web-based Distributed VLSI Design

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

Abstract

Emerging "systems-on-a-chip" will require a design environment that allows distributed access to libraries, models and design tools. In this paper we present a framework using the Object-Web technologies to implement Web-based CAD. The framework includes the infrastructure to store and manipulate design objects, protocols for tool communication and WebTop, a Java hierarchical schematic/block editor with interfaces to distributed Web tools and cell libraries.

1 Introduction

The design of future high-performance VLSI systems will require a distributed design and verification methodology due to the diverse expertise required at various levels of abstraction. These systems will require tools and generators that allow exploration of the design space at all level of abstraction. It will be desirable to provide access to these tools from a simple terminal with standard interfaces.

The advent of Internet has opened new opportunities in the areas of distributed design and 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 [1, 2, 3]. With Web based CAD, designers can utilize diverse utilities and expertise available all over the world.

VLSI system design typically involves design specification and verification, optimization and synthesis, generation of physical design and verification. A distributed microsystem design framework will facilitate seamless access to cell libraries and VLSI CAD tools like synthesizers, generators, optimizers and simulators distributed over the internet (Figure 1). The entry point in the framework is a hierarchical design editor which should be capable of accessing cell libraries located in different servers on the internet. The editor should be capable of extracting the design into netlists of various formats (e.g., SPICE, Verilog, etc.) and then interface with different CAD tools distributed over the Web.

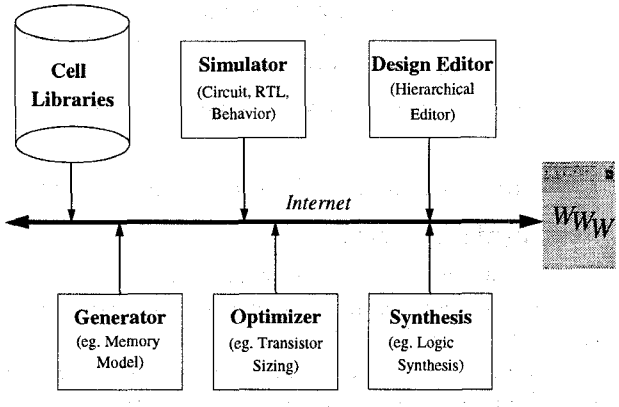

Figure 1: Distributed Microsystem Design Framework

The Web tools can themselves be distributed in the sense that each tool may invoke many other distributed tools. The example depicted in Figure 2, involving early exploration of power dissipation, demonstrates the usefulness of such a hierarchical framework. It is often desirable 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 (e.g., VHDL), parameterized library modules (e.g., ROM) and low-level schematics (e.g., transistors and gates). Several Web based tools exist to estimate power dissipation at different levels of abstraction (e.g., Pytha [9], PowerPlay [1], PPP [5]). Then a designer builds a tool *PowerZone* which given a 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, irrespective of the geographical location of the tool.

1063-9667/97 $10.00 © 1997 IEEE

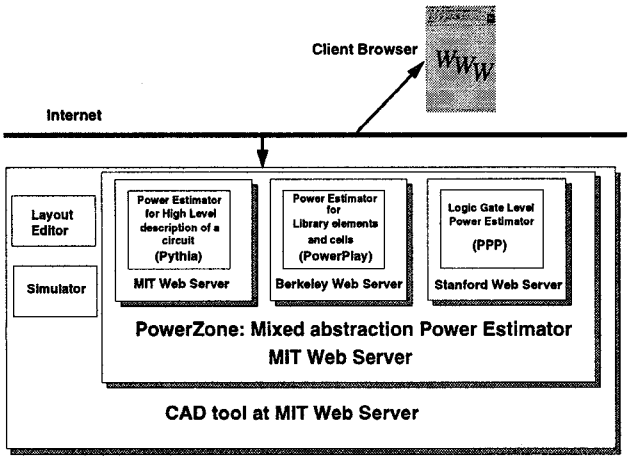

Figure 2: Hierarchical Web based design tools

2 Related Work

The popularity of the internet and the Web has lead to the development of several distributed design tools. An information based design environment, which helps users collect and manage information independent of the implementation platforms has been proposed [4]. PowerPlay, which helps in system level exploration of power consumption is available on the Web [1]. The WELD project[3], aims to construct a distributed CAD design environment enabling Internet wide IC design. A gate level synthesis and simulation tool, PPP, for power optimization can be accessed via a web interface [5]. There have also been efforts to provide Web-based Interfaces to executable CAD softwares[2, 6]. A network based simulation laboratory has been developed which can be accessed via a standard WWW browser [6]. Executable directed hypergraphs have been used to describe collaborative design activities on the internet [7]. Although there are numerous instances of design tools being available over the Web, there has been limited attention to the design of a framework which would allow the utilization different Web tools in a uniform and transparent fashion. In this paper we show how a distributed framework could be built, utilizing the core Web technologies, to support efficient communication and data exchange between different Web based tools.

3 Enabling Technologies for Web based CAD

In this section, we describe the technologies, mechanisms and protocols available that support distributed applications with the standard Web interface. Current tools support execution of applications at the server side using HTTP and CGI and use Java technology, scripts and plug-ins to create applications that are executed locally on the user's machine. Another approach is to use object technologies to define, locate and request computational services from participating applications, both remotely and locally. In this approach, the Web takes the role of providing uniform access and presentation mechanisms.

3.1 HTTP/CGI, Java and CORBA

CGI programs are usually referred to 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.

CGI programs and forms lack the interactivity and complex user interface. Java allows us to do complex client-side processing in a platform independent manner. Java *applets* can be embedded in HTML pages, which are loaded from the Web server and run on the client-side browser as a mini-application.

The Common Object Request Broker Architecture (CORBA) [10] 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. CORBA simplifies heterogeneous distributed computing and enables location transparency, activation transparency, language independence, and platform neutrality. CORBA is object-oriented, enabling many potential benefits such as reuse.

3.2 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 propose to use a open, interoperable, reliable, scalable Object-Web computing architecture.

The Object-Web architecture (Figure 3), is a three tier architecture consisting of the client tier, the application and data tier and the middle tier providing the middleware services. In this architecture, the client tier 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 can be either

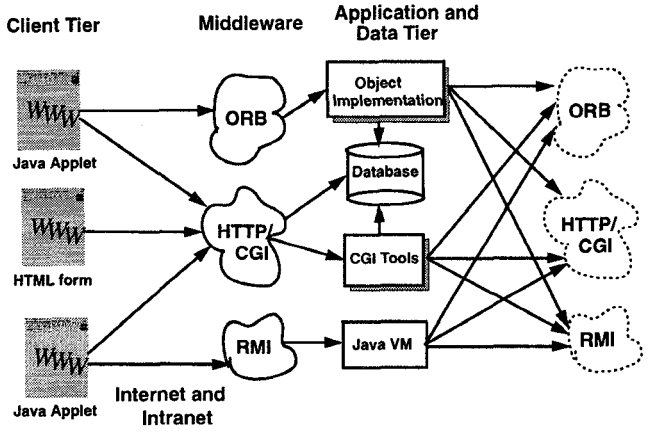

Figure 3: Object-Web Architecture for Distributed Computing

tools which are invoked through CGI protocols or can be objects providing their respective services. 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 architecture is flexible and enables tools and applications to effectively communicate with each other to build a distributed framework for Web based CAD.

4 Infrastructure for Hierarchical Web-based-CAD

In this section we present an infrastructure that enables hierarchical Web-based CAD. Any distributed application A can be sliced into different sub-applications of the form $< A_1, A_2, ..., A_n >$, where each A_i can be specified 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 input and output sets contains elements of the form,

```
<Parameter identifier>  <Parameter value>
<File identifier>       <File data>
```

It is sufficient to specify application's input and output in terms of parameters and files. Each Sub-application A_i is independent in the sense no other information 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. Although such a slicing of interactive tools may be obvious, such a formalism helps us design a distributed framework because 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.

We can easily see in a distributed environment that 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).

4.1 Web based Point tools

We first present the characteristics or capabilities of any tool in the framework and then we will go onto how to use these features to build an efficient hierarchical CAD framework accessible through the Web.

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)**. 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.

We propose to use the POST method and ENCTYPE multipart/form-data as standards for describing the BIS of 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. Also, the specification of the input and output sets as parameters and files can be adequately handled by the headers of multipart/form-data. Moreover, the media-type multipart/form-data follows the rules of all mul-

tipart MIME data streams, which is a standard.

Let us now 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". The input to this CGI program can be provided by a Java applet, or a HTML form. Both the input agents or input processes provide the data to the CGI program as multipart/form-data and they use the POST method to talk to the HTTP server, which would execute the CGI program. Any tool (back-end application) should be capable of decoding the input provided as multipart/form-data. The application then processes the input data and produces the output in a format specified by a **OutputType** parameter in the input set of the BIS of the tool. The OutputType parameter can take any of the following values: **HtmlType**, **BasicOutputType** or the **URLOutputType**.

If the display of the tool 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**. The OutputType of **URLOutputType** produces a URL which contains the output of the tool.

4.2 Hierarchical Web Tools

In this section we will show given the capabilities of all the HTTP address bound tools, how we can build tools which would utilize the capabilities of the existing tools and serve as a new tool in the framework. Consider a tool X on the Web bound to the HTTP address **http://x**, and another tool Y bound to the 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. It then calls the tool X and gets X's output as specified in its BIS. Z then proceeds to process the results and then calls the tool Y. It then gets Y's output and process all the available data to produce its own output. To the external world Z acts like an independent tool which has its own BIS. Therefore any client can make a connection to http://z and get the desired output of the tool, without being aware of the fact that Z actually calls two distributed tools on the Web. Figure 4 shows the dataflow involved in such a hierarchical tool. Now Z could be called from another CGI tool, therefore we can build a hierarchy of tools on top of the existing ones.

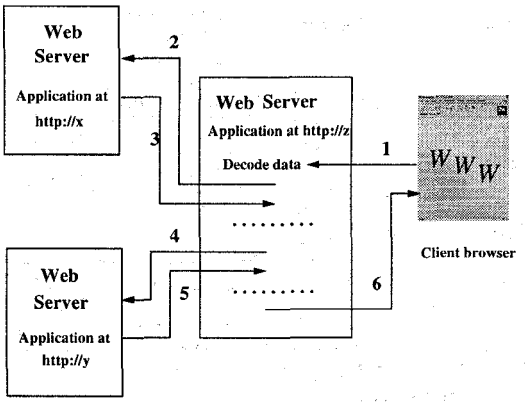

Figure 4: Data flow in the Hierarchical tools

4.3 Java and Web Tools

Java applets could be used in Web browsers instead of HTML forms to enter the input data of a tool and also do the interactive jobs mandated by many CAD tools. The applets POST any data to the server tools as multipart/form-data and the server tools write back the results and data to the applets again as multipart/form-data or as a simple URL, which the applet displays in the browser.

4.4 Example of Web-based Hierarchical Tool

In this example, we developed two point tools, Pythia and a graph plotting utility.

Pythia: *http://apsara.mit.edu/cgi-bin/pythia/pythia.pl.* 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]. 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. Pythia can be accessed from the HTML form at, *http://apsara.mit.edu/pythia-doc/pythia.html.*

The second tool which is a graph plotting utility, *http://apsara.mit.edu/cgi-bin/graph/drawgraph.pl*, takes in a file of X,Y data sets and produces a image of the plotted graph as a Jpeg image. The tool 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, 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 inter-

nally 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 4 where X is Pythia, Y is Graph Utility and Z is the new hierarchical web tool.

5 WebTop: Distributed Microsystem Design Center

In this section we describe WebTop, a Java based hierarchical schematic/block editor which is the entry point of the framework. WebTop supports hierarchical cells and multiple views of a cell. WebTop allows cells to have a behavioral view (Verilog) or a structural view (SPICE) in addition to the schematic views and can be extended to have physical (layout) views. WebTop has hierarchical netlisting capabilities for both Verilog and SPICE.

WebTop also supports distributed cell storage and access. Cells could be stored in different Web servers and loaded as URLs. This directly conforms to a distributed microsystem design framework where cells could be stored and accessed over the internet. A CellServer has been implemented which allows users to store cells remotely at the server's site using login and password verification. Users have the ability to access, modify and delete cells in their respective directories. All cells saved at the server's site are automatically linked as URLs and accessible for use for all other users. Only the owner of the cells has the permissions to modify or delete the cells. Figure 5 shows a screen dump of the editor, the library manager with the URL cell load dialog box.

WebTop uses the object-web architecture in Figure 3 for distributed tool integration. CAD tools in the application tier can be either CGI programs, Java objects or tools wrapped around as CORBA objects. The standard specification of RMI and CORBA makes it very easy to make transparent calls to tools which communicate using RMI or CORBA. Tools can also be CGI programs and could be integrated using the framework described in Section 4.

WebTop has interfaces to different 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:

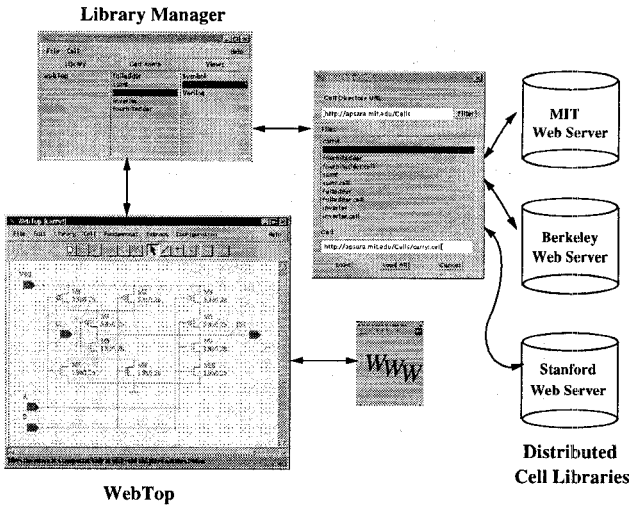

Figure 5: WebTop: Hierarchical Schematic Editor

http://apsara.mit.edu/spice.html. Using WebTop, the user can extract a design schematic into a SPICE netlist. WebTop, which is an applet 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 Pythia. The user extracts the design as a Verilog netlist and WebTop submits the netlist along with other parameters specified in the BIS of Pythia as multipart/form-data. Pythia which is a CGI program bound to a URL decodes the input multipart/form-data from WebTop, computes the power dissipation and produces a dynamic URL containing the results as a HTML file. Pythia sends back to WebTop the dynamic URL. WebTop then uses the browser context to display the dynamic HTML page.

We are currently working on interfacing WebTop to the UC Berkeley PowerPlay and other end-tools like MTCMOS sizing tools [8]. Work is also in progress to integrate different CAD tools using RMI and CORBA mechanisms. As we can see, 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 6 shows the tool integration architecture of the distributed framework.

WebTop demonstrates the feasibility of a Web-based VLSI design framework. The user opens up the hierarchical schematic editor, which is a Java applet from the Web browser. The user loads differ-

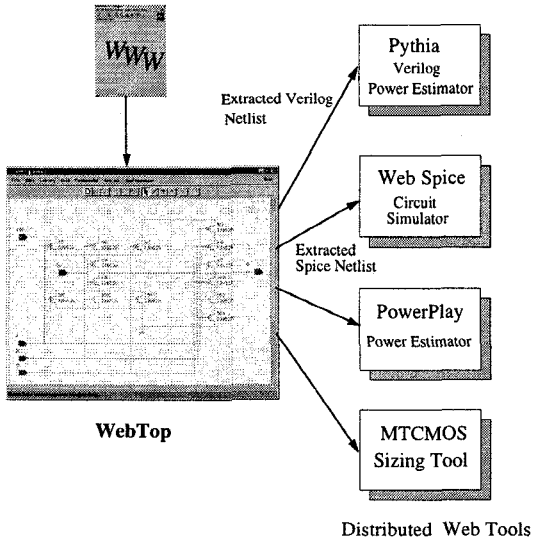

Figure 6: WebTop: Tool Integration

ent cells from different Web servers as URLs, edits the cell and then uses the CellServer to store cells. The user can then extract the design as a netlist and then makes use of the distributed CAD tools integrated with the editor in a transparent fashion. The framework supports a *plug-and-play* architecture for CAD tools which could eventually lead to a *pay-per-use* model for VLSI CAD tools and cell libraries.

6 Summary

The increasing complexity of microsystem design mandates a distributed and collaborative design environment and the World Wide Web serves as a desirable platform realizing these goals. The major concerns of doing CAD over the Web involves the issues of security, standardization and network limitations. The Web infrastructure should allow secure communications over the network. Most of the Web agent technologies are still in the early stages of maturity and standardization of protocols, security and communication mechanisms are necessary to fully utilize the potential of a distributed design environment. The current networks tend to be a major bottleneck to ensure fast transfer of large amounts of data across the network. However, the recent developments in high bandwidth networks are promising in alleviating this bottleneck in the near future. A careful mix of client end processing and server side programming is necessary to ensure fast response time.

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 implementation of the WebTop applet.

References

[1] D. Lidsky and J. M. Rabaey. Early Power Exploration - A World Wide Web Application. *Proc. Design Automation Conf, Las Vegas, NV*, pp. 27–32, June 1996.

[2] Microsystem Design Test Bed, http:// web_nt.sainc.com/ arpa/ msdproject/ testbed.htm.

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

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

[5] A. Boglio, L. Benini, G. D. Micheli, and B. Ricco. PPP: A Gate-level Power Estimator - A World Wide Web Application. *Stanford Technical Report No. CSL-TR-96-691*, 1996.

[6] N. H. Kapadia, M. S. Lundstrom, J. A. B. Fortes, and K. Roy. Network-based Simulation Laboratories for Microelectronics Systems Design and Education. *International Conference on Microelectronic Systems Education, Arlington, Virginia* (in press), July 1997.

[7] H. Lavana, K. Amit, F. Brglez, and K. Kozminski. Executable Workflows: A Paradigm for Collaborative Design on the Internet. *Proc. of the Design Automation Conference*, pp. 553–558, June 1997.

[8] J. Kao, A. Chandrakasan, and D. Antoniadis. Transistor Sizing Issues and Tool for Multi-threshold CMOS Technology. *Proc. of the Design Automation Conference*, pp. 409–414, June 1997.

[9] T. Xanthopoulos, Y. Yaoi, and A. Chandrakasan. Architectural Exploration Using Verilog-based Power Estimation: A Case Study of the IDCT. *Proc. of the Design Automation Conference*, pp. 415–420, June 1997.

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