

WELD – An Environment for Web-Based Electronic Design

Francis L. Chan
fchan@eecs.berkeley.edu

Mark D. Spiller
mds@eecs.berkeley.edu

A. Richard Newton
rnewton@eecs.berkeley.edu

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley

Abstract

Increasing size and geographical separation of design data and teams has created a need for a network-based electronic design environment that is scalable, adaptable, secure, highly available, and cost effective. In the WELD project we are evaluating aspects of the network integration and communication infrastructure needed to enable such a distributed design environment. The architecture of WELD and the components developed to implement the system, together with performance results, are described and evaluated.

1 Introduction

Advances in areas such as software methodology, operating systems, storage systems, and programming languages have often had an enormous impact on EDA. The explosive growth in the development of wide-area network infrastructure over the past few years indicates an opportunity for the industry and the field of computer science in general to make a leap to a new generation of capabilities. Specifically, we envision the entire EDA community organized as an integrated distributed environment [39] that offers users the ability to create an evolvable, customizable and adaptable “virtual” design system that can couple tools, libraries, design, and validation services. Beyond that, the system could also provide manufacturing, consulting, component acquisition, and product distribution, encompassing the developments of companies, universities, and individuals throughout the world.

In this paper we present an approach to the network infrastructure developed for such a distributed design system. The motivation and description of the WELD project [47] is given in Section 2. Comparisons to related work are documented in Section 3. A high level view of the WELD architecture and its key components is introduced in Section 4 and results and experiences from our efforts are described in Section 5.

2 The WELD Project

The goal of the WELD project is to pull together emerging technologies, such as network communications, visualization [45], alternative interfaces, and new algorithmic approaches to design, to provide the basis for a next-generation EDA system. Working sets of modern VLSI designs, which approach two gigabytes of data today and will continue to grow, are forcing design teams to become larger and more distributed [35], indicating a great need for a collaborative, distributed design system that is scalable, adaptable, secure, highly available, and cost effective.

The World Wide Web [49] today can be viewed as a “bad multicomputer”, which has all the possibilities for parallelization of a normal multiprocessor, but a great deal more uncertainty in the latency, performance, and consistency guarantees that are offered. By carefully analyzing the usage of today’s users and tools, it may be possible to somewhat relax the traditional ACID properties (Atomicity, Consistency, Isolation, and Durability) [15] required for reliable data management and exchange and make use of the network for distributed collaborative design. A new set of semantics referred to as BASE [13] (Basically Available, Soft state, Eventual consistency) has been proposed that stresses availability and low latency, instead of the traditional transactional capabilities of ACID that have often proven to be too high in overhead and complexity. In BASE, a fast, approximate answer may be more valuable initially than a late-but-correct response. Part of the design challenge facing the WELD group was to find a general, flexible architecture that features both properties and addresses as wide a range of EDA requirements as possible.

To be successful, the WELD system must be built upon an open design environment that is *scalable and adaptive*, where system service and performance are not sacrificed with the addition of tools, users and different technologies, and the use of components guarantees flexibility and extensibility. In the current implementation, users can easily extend features or incorporate into the system existing tools and technologies, regardless of programming language, operating system base, or development environment. A *wide range of computing resources*, from high-end workstations to mobile PDAs, are supported in WELD through the use of servers and proxies located both locally and in the network.

Although the Web potentially offers substantial performance improvements for suitably partitioned or iterative designs [4], additional infrastructure will be required to assure *robustness*, especially in the area of data management. Intermediate state of long processes must be stored so that design states can be

recovered efficiently and incrementally after any hardware, software, or network failure. *Security* measures¹ must also be available so that data security is never comprised.

Applications and services in a distributed environment should manifest *high availability*, providing users with consistent on-demand access regardless of time or location. The Web offers several built-in advantages to a distributed design system. The wide distribution of Java-enabled browsers enables the system to be readily accessible to a *large user base*, making it fertile ground for innovation and wide-area collaboration. In addition, Java clients allow for *consistent and intuitive user interfaces* and *platform independence*, thus reducing start-up and training time.

3 Related Work

Our architecture and infrastructure for a distributed design environment can be compared to CAD frameworks [17, 48]. Comparable features include:

- A design database – distributed data manager and client object management package
- A design data manager – versioning, security and meta-data handling capabilities
- A design process manager – server wrapper and distributed workflow system.

However, our approach differs from others' work² in a number of fundamental ways. Most of the past efforts in the area of CAD frameworks are involved in introducing new systems, techniques or extensions in specific areas of design data management [33, 46], design meta-data management [14, 24, 34], and flow, process and tool management [12, 16, 18, 25, 43]. WELD, however, concentrates on providing reliable, scalable connection and communication mechanisms for distributed users, tools and services. It also extends the traditional client/server model by treating both tools and data as potentially mobile components. This component-based approach facilitates both the fragmentation of large data sets into manageable and transferable segments as well as the separation of sophisticated tools into logical components that can be run on either clients, servers, or proxies. While we are also involved in the development of EDA applications [47], the goal of the work presented in this paper is to provide the enabling and enhancing mechanisms that leverage existing systems and toolkits. The infrastructure was deliberately engineered such that no restrictions or assumptions are placed on data representation, design methodologies or data and tool usage, instead allowing policy choices to be built on top of the infrastructure. Finally, the WELD project avoids the tight coupling found between many frameworks and their operating environments (i.e. UNIX and NFS) by using platform-independent standards such as Java, sockets, proxies, and generic string-based communication protocols.

¹ Security features can be layered upon the base WELD architecture.

² A number of external efforts that are relevant to our research include the REUBEN system [26], CREW [8], the Design Agent system [3], and Active Documents [38].

4 System Architecture

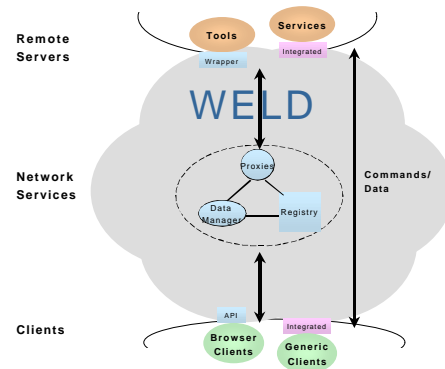


Figure 1: WELD System Architecture.

The three-tier WELD architecture³ (Figure 1) consists of:

- *Remote Servers* – network-accessible tools or services. These can be either command-line tools encapsulated by server wrappers or tools with built-in support for socket connections and WELD communication protocols.
- *Network services* – infrastructure components found in the network that are available on demand to applications. Examples include the *Distributed Data Manager*, *Proxies* and *Registry Service*.
- *Clients* – applications that use the WELD infrastructure to access network resources. These can be either Java *Browser Clients*, which incorporate the WELD Java client packages and are run via Java-enabled browsers, or *Generic Clients*, which are developed in socket-enabled languages such as Java, C, C++, or PERL and use the client protocol(s).

The *Client-Server Communication Protocol* and *Client-Database Communication Protocol* are the mechanisms that enable these network entities to communicate with each other.

4.1 Server Infrastructure

A *Server Wrapper* package [41] was implemented to provide an interface between normal UNIX applications and the WELD system, abstracting away the system internals from application integrators. Its network code allows programs to switch between the role of a client, server, and proxy in the WELD system. Also included is the capability to send heartbeats and periodic updates to remote systems, providing status and failure information.

4.2 Network Services

The *Distributed Data Manager* provides data services to remote applications via generic network socket connections. It utilizes the WELD client-database communication protocol to serialize and mirror Java object hierarchies to an object-oriented database server in the network. The data manager facilitates

³ The WELD architecture is covered in greater detail in [5, 40].

customizability in the addition of new message types as well as the transfer and translation of data between different tools. It is also accessible via multiple interfaces, including the WELD protocols and HTTP.

In the WELD architecture, *Proxies* are software programs that reside in the network and act as transparent, “intelligent” agents between clients, network servers, and other proxies. Although originally used as a means to circumvent the Netscape Java security model⁴, proxies offer a wide range of other possibilities, which include translating between data formats, filtering and redirecting of data flow, tracking and monitoring network tool usage, caching, and security.

The *Registry Service* server dynamically maintains information on the availability of network resources and their status, including uptime, history, failure rate, etc.

4.3 Client Infrastructure

The *Java Client Persistent Object Management Package* [6] works in tandem with the data manager. It enables Java (client) objects to be managed and manipulated by a network data server. In addition to object storage and retrieval⁵, it extends remote data backend capabilities, such as querying and versioning, to client users, through a set of extensible APIs.

A *Java Network Client Package* [6] was also developed to allow Java applets and applications to communicate with WELD network resources.

5 Results and Experiences

5.1 Java-Based OCT - A CAD Data Manager

The first application of the data server and persistent object management package was a Java-based version [6] of the Berkeley OCT Tools. OCT serves as the data manager for a large number of VLSI/CAD applications following the Berkeley CAD framework [30]. The Java persistent object package provides a layer of abstraction for the data server, presenting capabilities including object saving and loading, as well as object traversal with an interface similar to that of the OCT generator. A data schema was created that follows the OCT convention, in which objects can be arbitrarily attached to each other, providing a general mechanism for data organization upon which developers can impose specific policies. Several applications that involved the construction, display, and partitioning of netlists were built upon this infrastructure.⁶

5.2 The Distributed Workflow System

The *Distributed Workflow System* [41] pulls together many of the WELD components developed, including the server wrapper, client packages, proxies, registry server, and data manager. The

result is a cohesive, distributed design system that uses and tests the limits of the network infrastructure, as well as sheds light on many of the criteria discussed in Section 2, such as ACID/BASE properties, flexibility, and ease of use. Components (see Figure 2) include a Java front-end (Workflow Monitor, see Figure 3) in which workflows are created, displayed, and tracked, (2) tool servers, which actually perform operations upon data, (3) a registry that tracks distributed resources, (4) a database server that stores the final and intermediate results, and (5) a central control proxy (workflow server) that receives the workflow graph from the client and then handles workflow execution. Tools in the system in the 1997 DAC demonstration [7] included (1) simple programs, such as *rwho*, *finger*, *email* [27], (2) UC Berkeley CAD tools, including the Nova state optimization tool [30], *VIS* [44], *SIS* [36], *POLIS* [32], and (3) a commercial CAD tool, the Synopsys Design Compiler [10].

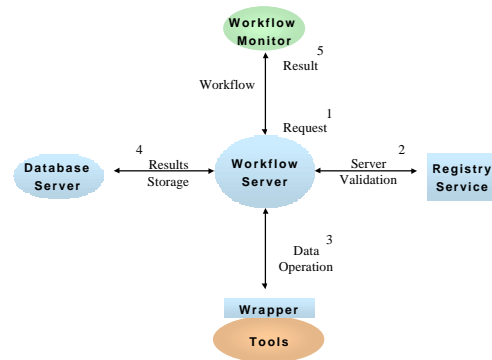


Figure 2: Architecture of the Distributed Workflow System. The numbers indicate the flow of events during a workflow execution.

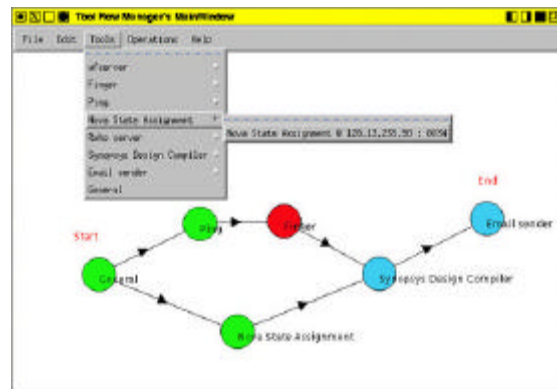


Figure 3: Workflow Monitor of the Distributed Workflow System.

⁴ Java applets can only make socket connections to the server from which they were downloaded.

⁵ Other efforts in the area of Java object and database connectivity include the Java Database Connectivity API [22] and various commercial persistent Java object systems [20, 28, 29].

⁶ EE 244, a graduate-level course on CAD in UC Berkeley Fall 1996, used the Java-based OCT package for several class assignments [11].

5.3 Experiences and Measurements

5.3.1 Data Rates

Most of the WELD components were developed and tested on a high-speed local area network more representative of a company Intranet than a widely distributed design environment. Several measurements were made to gauge the effects of wide area networks on the system using a design process requiring interaction between a machine at Berkeley and a machine at North Carolina State University (NCSU). Although these measurements are not comprehensive, they provide useful order-of-magnitude estimates for performance that indicate that the length of the link is not a current system bottleneck.

Figure 4 shows a set of data rate comparisons for both local and distant data transfers in C++. While the results (Table 1) for local server with large buffer transfer, local server with byte parsing, and remote server with large buffer transfer, scale linearly, the fact that byte parsing, even in C++, delayed the system more than the wide-area network was surprising.

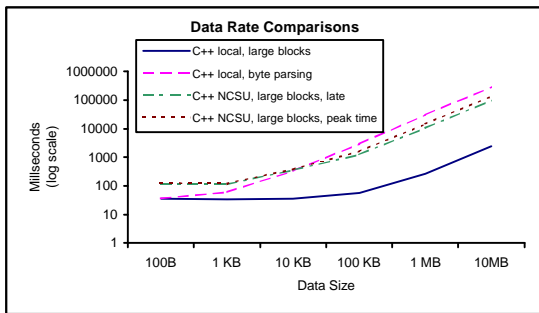


Figure 4: Data Rate Comparisons

| Data Rate Comparisons | | | | | | | | | | | | |
|-----------------------|---------------------|------|------|----------------------------------|--------|--------|--------------------|-------|-------|--------|-------|------|
| Data Size | Local, large buffer | | | Local, byte parsing (Time in ms) | | | NCSU, large buffer | | | Ratios | | |
| | Avg. | Min. | Max. | Avg. | Min. | Max. | Avg. | Min. | Max. | Local | Parse | NCSU |
| 100 B | 36 | 24 | 51 | 37 | 27 | 49 | 114 | 101 | 130 | 1 | 1 | 3 |
| 1 KB | 35 | 24 | 46 | 61 | 50 | 71 | 114 | 101 | 130 | 1 | 2 | 3 |
| 10 KB | 36 | 26 | 48 | 332 | 323 | 345 | 359 | 347 | 374 | 1 | 9 | 10 |
| 100 KB | 56 | 44 | 66 | 2917 | 2895 | 2951 | 1243 | 1209 | 1286 | 1 | 52 | 22 |
| 1 MB | 270 | 238 | 359 | 30510 | 29736 | 31789 | 10482 | 9959 | 11718 | 1 | 113 | 39 |
| 10 MB | 2446 | 2303 | 2717 | 284207 | 282755 | 286085 | 97506 | 97332 | 97745 | 1 | 116 | 40 |

Table 1: Data Rate Comparisons

Figures 5 and 6 compare the network performance of C++ and various Java implementations. These results indicate that, for the same hardware platform, the basic network capabilities for Java applications (Sun Java Development Kit (JDK) 1.02 [21]) and Java applets (Netscape 3.01 on a Sun Sparc 20) are comparable to that of C++.

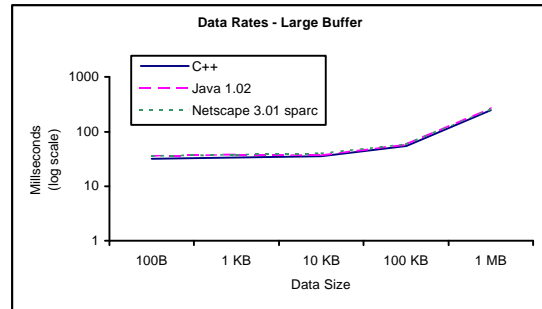


Figure 5: Data Rates - Large Buffer

5.3.2 Java

Prototyping Java applications provided a number of insights with regards to using Java and a distributed server system for computing-intensive (EDA) applications. Java and Java-based OCT were well-received within a group of users (UC Berkeley CAD Group) currently developing applications primarily in C. This was due to Java's object-oriented nature, ease of programming for both general programming logic and user interfaces, good abstraction of I/O and networking, useful documentation, library functions, and reasonable performance.

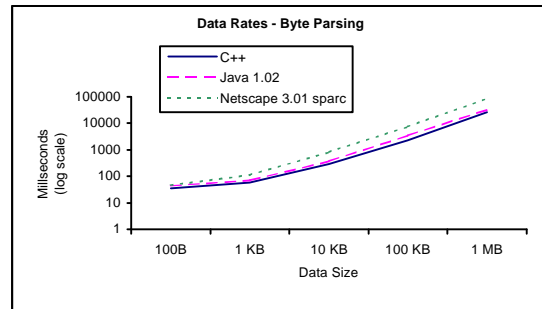


Figure 6: Data Rates - Byte Parsing

Most of WELD group's Java development was implemented using the JDK, which offered favorable performance and promised a portable Java front-end. Java performance was initially anticipated to be a system bottleneck, but extensive application usage indicated that it was reasonable, and will only improve as just-in-time compiler [23] technology advances. In addition, the focus on a proxy-based architecture allows the off-loading of work onto proxies, and can thus reduce many (Java) clients to lightweight communications and user interface tools. Fine object granularity, and the resulting large number of objects transmitted, turned out to be more of an issue than Java performance.

In order to reduce the resulting network and processing overhead, *DataObjects*, generic persistent objects specializing in string-based data storage, were implemented. Despite this work,

Proceedings of the 27th ACM/IEEE Design Automation Conference, pp. 136-141, 1990.

- [5] F. Chan, "Architecture and Infrastructure for a Distributed Design Environment – A Client Perspective," MS Report, UC Berkeley, May 1997.
- [6] Francis Chan's research page,
<http://www-cad.eecs.berkeley.edu/~fchan/research>.
- [7] P. Clark and R. Goering, "Web-based design hoists new sail," *Electronic Engineering Times*, June 16, 1997, issue 958. Demonstration materials at
<http://www-cad.eecs.berkeley.edu/weld/dac97>.
- [8] CREW, The Collaboratory for Research on Electronic Work, <http://www.crew.umich.edu>.
- [9] A. Denning, *ActiveX Controls Inside Out*, Microsoft Press, Redmond, 1997.
- [10] The Design Compiler Family Datasheet, http://www.synopsys.com/products/logic/design_comp_ds.html.
- [11] EECS 244 Fall 96 home page,
<http://www-cad/~newton/courses/>.

- Proceedings of VLDB*, pp.144-154, Cannes, France, September 1981.
- [16] J. W. Hagerman and S. W. Director, "Improved tool and data selection in task management," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 181-184, June 1996.
- [17] D. S. Harrison, et al., "Electronic CAD Frameworks," *Proceedings of the IEEE*, vol. 78, no. 2, pp. 393-417, February 1990.
- [18] A. Hoeven, et al., "A flexible access control mechanism for CAD frameworks," *European Design Automation Conference with EURO-VHDL*, pp. 188-193, September 1994.
- [19] The Informix's Universal Web Architecture, <http://www.informix.com/informix/bussol/uwa/uwa.htm>.
- [20] Infoscape, <http://www.infoscape.com>.
- [21] Java Development Kit, <http://java.sun.com/products/jdk/1.1/index.html>
- [22] The JDBC page, <http://java.sun.com/products/jdbc/index.html>.
- [23] Symantec Just In Time Compiler Performance Analysis, http://www.symantec.com/jit/jit_pa.html.
- [24] E. W. Johnson and J. B. Brockman. "Incorporating design schedule management into a flow management system," *Proceedings of the 32nd ACM/IEEE Design Automation Conference*, pp. 88-93, June 1995.
- [25] S. Kleinfeldt, et al., "Design methodology management," *Proceedings of the IEEE*, vol. 82, no. 2, pp. 231-250, February 1994.
- [26] H. Lavana, et al., "Executable Workflows, A Paradigm for Collaborative Design on the Internet," *Proceedings of the 34th ACM/IEEE Design Automation Conference*, June 1997.
- [27] Descriptions of finger, mail, and rwho are available from section one of the online UNIX Programmer's Manual.
- [28] NetDynamics, <http://www.netdynamics.com>.
- [29] "Objectivity for Java," *The Insider: News and Views for Objectivity Users*, July 1997.
- [30] OCTTOOLS-5.2 Reference Manual, UC Berkeley, 1993.
- [31] D. Platt, *The Essence of COM and ActiveX*, Prentice Hall Computer Books, Jan. 1998.
- [32] The Polis Project, <http://www-cad.eecs.berkeley.edu/Respep/Research/hsc>.
- [33] W. Schettler and S. Heymann, "Towards support for design description languages in EDA frameworks," *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 762-767, November 1994.
- [34] J. Schubert, A. Kunzmann and W. Rosentiel, "Reduced design time by load distribution with CAD framework and methodology information," *European Design Automation Conference with EURO-VHDL*, pp. 314-319, September 1995.
- [35] SEMATECH, <http://www.sematech.org/member/division/dsgn/ecad/home.htm>.
- [36] E. M. Sentovich et al., "SIS: A System for Sequential Circuit Synthesis," Technical Report UCB/ERL M92/41, May 1992.
- [37] The Silicon Integration Initiative (formerly the CAD Framework Initiative, CFI), <http://www.si2.org>.
- [38] M. J. Silva, "Active Documentation for VLSI design," Ph.D. dissertation, available as Technical Report UCB/CSD-94-843, UC Berkeley, December 1994.
- [39] M. D. Spiller and A. R. Newton, "EDA and the Network", *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 470-476, November 1997.
- [40] M. D. Spiller, "Architecture and Infrastructure for a Distributed Design Environment – A Server Perspective," MS Report, UC Berkeley, November 1997.
- [41] Mark D. Spiller's research page, <http://www-cad.eecs.berkeley.edu/~mds/research>.
- [42] The Vela Project, <http://www.cbl.ncsu.edu/vela/>.
- [43] I. Videira, P. Verissimo and H. Sarmiento, "Efficient communication in a design environment," *Proceedings of the 33rd ACM/IEEE Design Automation Conference*, pp. 169-174, June 1996.
- [44] The VIS Group, "VIS: A system for verification and synthesis," *Proceedings of the International Conference on Computer Aided Verifications*, pp. 428-432, November 1996.
- [45] The DARPA Intelligent Collaboration & Visualization program (IC&V), <http://www.ito.darpa.mil/ResearchAreas96/IntellCollabVisual.html>.
- [46] F. R. Wagner, L. Golendziner, and M. R. Fornari, "A tightly coupled approach to design and data management," *European Design Automation Conference with EURO-VHDL*, pp. 194-199, September 1994.
- [47] The WELD Project, at <http://www-cad.eecs.berkeley.edu/weld>.
- [48] P. van der Wolf, *CAD Frameworks Principles and Architecture*, Kluwer Publishers, Boston, 1994.
- [49] The WWW consortium, <http://www.w3.org/WWW/>.