Java™ in Electronic Design Automation

Peter Denyer
Technical Market Development
Sun Microsystems
Mountain View, CA 94043-1100
Tel: 415-786-3744
Fax: 415 786-3763
e-mail: peter.denyer@sun.com

Jean Brouwers
Java & EDA Consultant
Sun Microsystems
Mountain View, CA 94043-1100
Tel: 415-786-4979
Fax: 415 786-3763
e-mail: mrjean@best.com

Abstract– Increasing design complexity and the need for multi-disciplinary / multi-national design collaboration is causing a paradigm shift in the EDA application environment. This shift is necessary in order that time-to-profit goals are met in increasingly compressed market windows. The envisioned paradigm shift is enabled through Sun's Java™ technology. This technology will impact significantly the development, deployment, use and support of Electronic Design Automation (EDA) applications. This paper will examine some of the factors influencing this forthcoming EDA revolution and review some of the challenges yet to be resolved.

I. Introduction

Since it's introduction barely 600 days ago, nothing has captured the imagination and mind-share of the "high-tech" industry as Java. Already, Java-enabled applications are being deployed in Global Fortune 1000 companies; providing capabilities in a cost-effective manner never before seen in the corporate workplace.

These applications are not just simple Java applets for enhancing Web pages; rather, they are robust, mission-critical applications providing significant competitive advantage. National Semiconductor, for example, is using a Java application to keep the entire company, and its customers, on-line and up -to-date on its latest parts inventory status.

Significantly absent from this Java deployment is the EDA industry. Developers and consumers of EDA applications have yet to benefit in any significant way from this computing revolution. Why is this, and what are the prospects for change?

II. EDA Application Architectures

Before delving into how Java can impact the EDA community, it is instructive to view the current state of affairs in EDA application delivery and understand the driving forces that can make Java as ubiquitous in EDA as it is today in other industries.

A. Desktop-centric

Current EDA applications tend towards large, monolithic tool suites running on fast desktop workstations with networked file systems. A significant proportion of the applications used on a daily basis consume the resources of even the highest-end engineering desktop hardware. Applications requiring the fastest processors and 64 Megabytes of physical memory are not unusual. Systems configured with 128 to 256 Megabytes of memory are becoming commonplace.

This need for higher capability systems is especially evident in the IC design segment. The challenges brought about by deep sub-micron design are straining the limits of conventional desktop hardware. Applications such as deep sub-micron technology CAD, signal integrity analysis, chip and board-level simulation, design rule checking and layout fracturing are requiring multi-processor systems with Gigabytes of real memory for timely computation.

Clearly, desktop hardware is not the most cost effective way to deliver EDA computation cycles to the individual engineer. Even with the continually increasing price/performance rations for today's hardware, robustly configured hardware on each designers desk is not a preferred option.

B. Islands of Competence

Additionally, in this desktop-centric environment users and applications have remained in relative isolation on their own local networks. With this comes the attendant overheads of updating applications, managing data and other administrative tasks which compound the design task.

This relative isolation constrains design collaboration to the immediate workgroup. Members of the design team must know the status of the latest design data, the location and revision dates of the necessary applications, and have the ability to access easily the tools and data. This can occur with reasonable facility in the immediate workgroup. Design collaboration over widespread geographies is significantly more difficult.

This isolation leaves the design project vulnerable to single-point system failures. Unscheduled system down-time, disk crashes, power failures and the like can severely impact critical path design processes.

C. Client-server

The requirement for ever increasingly powerful hardware in support of the EDA function is fueling the move to a client-server paradigm. In this paradigm, engineering workgroups are supported by powerful compute servers, configured for the most demanding applications.

A classic two-tier client-server application environment is evolving into the environment of choice at many of today's leading electronic product manufacturers. The server becomes the central repository of core design content and distributed computing becomes the model of choice. Compute intensive tasks are performed on the server. Less intensive tasks are

ASP-DAC '97
0-89791-851-7$5.00

relegated to the desktop hardware; tasks such as supporting the application frameworks, data preparation, smaller applications, and data analysis tasks. The spiral of ever increasing compute power and its attendant cost at the desktop is diminished significantly as hardware funding is increasingly focused on the server configuration.

Less common, but perhaps more interesting in large-scale EDA environments, is a three-tier EDA client-server environment. In this scenario, the desktop hardware, communicates with an intermediate server which performs a load balancing function, distributing the request for computational resource out to specialized machines, or to other available computer hardware in the network.

The client-server paradigm goes a long way in addressing the underlying factors that can impact EDA productivity and overall time-to-profit goals. Overall hardware cost can be better contained. Acquisition and on-going maintenance of mission critical applications is easier as only one location, the server, needs to be updated. Additionally, the server, is configured as a mission-critical resource, protected in climate controlled rooms, configured for maximum up-time, and managed by professional Information Technology (IT) staff.

However, even in this client-server paradigm, nothing substantial has changed in the way that designers get their work done. Disparate EDA applications still have their unique user interfaces and data formats. Engineers still need to know where the latest version of the necessary applications reside. New application software releases are needed to correct the inevitable software bugs found during the design process. New mechanisms are needed to foster geographic collaboration. Herein lies the promise of Java and the Java Computing paradigm.

III. A Vision for the Future

The current EDA computation paradigm—whether supported by powerful networked desktop hardware, or the evolving client-server hardware configuration—must evolve to a new level if corporate time-to-profit goals are to be achieved. The evolution of the Internet, the prevalence of corporate Intranets and the growing influence of Java-based computing provides the infrastructure for this new paradigm.

A. Web-aware Applications

The first step in this new wave of EDA computation will be the evolution of Web-centric applications. Here, the power of the client-server configuration will be extended to take advantage of the expanding Inter/Intranet infrastructure.

Looking at EDA applications in the abstract, they can be decomposed into three components - the core algorithms, a communication channel and a user interface. There is no immediate purpose for expending effort on the core algorithms. These algorithms have been written, optimized for maximum performance and put to the test in countless design iterations. The compute intensive nature of these components require them to remain on the server. One change is that they will now be encapsulated by a Java shell. This encapsulation provides the mechanism by which users can access the application easily from anywhere in the network using any machine supporting the Java Virtual Machine (JVM) specification.

The communication channel is simply part of the existing Inter/Intranet infrastructure. The application user interface will co-exist with today's web browsers, and tomorrow's "web-top" user environments such as HotJava™ Views™. Browser plug-ins and Java applets will understand the unique data structures for design capture and navigation, application input, run-time control and results viewing. The web browser now becomes the application framework, which helps promote standardized and universal interfaces for both algorithms and design data.

This shift to a Web-centric view of EDA tools and tool usage further evolves the new-wave EDA paradigm. Collaborative design over diverse geographies is simplified as design team members access applications and project as commonly understood URL's as opposed to absolute locations which could easily change.

This shift also has a collateral benefit. The usefulness of legacy software can be expand by Java encapsulation. Such encapsulation can allow application access from any Java-enabled "web-top".

B. Web-centric Applications

The longer-term evolution of EDA applications in the network-centric environment will completely change the way these tools are used. The tools will be built from the ground up as distributed applications. The trend will be to smaller functional components interacting through clearly defined application program interfaces (API's) instead of the traditional monolithic EDA application. The ability to "mix-n-match" applications from multiple vendors will be facilitated and enhanced through these API's.

The main interface to this envisioned EDA design system is through Web-browsers or custom Web-oriented desktops with hyper-links to both data and applications. The notion of "hyper-spreadsheet" or forms-based data specification that will enable designers to access and manipulate information based on what it is, and not where it resides, may become the norm for such applications.

IV. Java Computing

Currently, the main focus of Java is on client-server computing within larger corporations. Java Computing proposes an open computing model consisting of networks of "thin" client stations and central servers. Tailored, single function applications are downloaded from the central server and run on the client desktop stations.

Java Computing is particularly attractive for enterprise applications like banking, on-line service centers, help desks, where a limited number of tasks are performed by a large number of users. In such environments personal computers are too costly to maintain and often too hard to use. Java Computing promises to significantly reduce the total cost of ownership per seat, mainly due to centralized system administration and zero client-side attendance.

The client-server model of Java Computing encourages smaller, single function applications which are easier to learn and use. This and the "write once, run anywhere" benefit of Java decreases the development and deployment effort for new applications. Java Computing combines the advantages of scalable network computing with the economics of mainframe computing. The Java Computing model is very attractive

across all departments of an organization including Engineering.

A. Key Java Language Features

Java is a programming language like C or C++ but different by design. The Java language is less complex and more robust than C++ but syntactically similar. By removing complexity (like memory management, multiple inheritance) it is easier and less error prone to write Java software. Rigorous typing (like string, objects, no pointers) makes Java applications more robust and allows programming errors to be detected early in the development cycle. Java encourages more modular and smaller programs (through classes, packages, interfaces). In addition, the Java language includes built-in support for multi-threading, synchronization, exception handling, network programming, security, graphical user interface and a wide range of standard Application Programming Interfaces across all Java platforms.

Probably the most significant feature of Java is the "write once, run anywhere" concept which removes the need for porting and maintaining different platform versions of applications. The Java compiler generates a single, neutral, binary format (Java bytecodes). The single binary runs on all Java environments, either inside Java-capable browsers, standalone Java Virtual Machines (JVM) on a host operating systems or on top of special Java devices based the JavaChip™ and JavaOS™.

The modularity, compactness and security of the Java language facilitate distribution across the Inter/Intranet—unlike any other language. Once written, Java applications run everywhere. Not only developing but in particular deploying and supporting applications written in Java is quicker and less costly than conventional methods.

B. Why Develop Software in Java?

Apart from network and web-based programming, there are several other compelling benefits to develop software in Java.

— Software developers are more productive in Java. Due to the single, uniform Java environment and the unambiguous language definitions across all platforms, developers spend less time debugging and dealing with low level details. Early experience with Java indicates development times cut by more than half.

— Java yields better quality and better documented software since it promotes good software engineering practices and imposes stricter "design rules" for software development. Rigorous typing allows detection of programming errors earlier in the development cycle. Automatic memory management (with garbage collection) and absence of pointers remove the most common cause of problems. Robust exception handling highlights unexpected and otherwise undetected conditions. Java includes conventions and tools to generate developer's documentation from the source code.

— Java lowers the software deployment and manitenance cost. "write once, run anywhere" eliminates repetitive porting of the software to various target platforms. Because of the single binary format, software can be delivered earlier and simultaneously for all platforms. Conversely, Java opens any hardware platform up to a larger range of available software.

— Java allows a gradual, incremental transition from stand-alone to multi-tier web-based applications. Existing C/C++ applications need not be re-implemented entirely in Java. Instead, application suites can be pure Java based or a mixture of Java and C/C++ code.

— Since web-centric applications provide universal access, Java-based design tools integrate and interoperate better with other applications on the desktop. In addition, Java is an ideal extension language to provide user-specific functionality since it is both platform and tool vendor neutral.

C. Java Development Tools

There are a number of Java development tools available from different vendors and for various platforms. Although the functionality of Java tools may vary substantially, all generate the standard Java bytecodes.

Individual Java tools are Integrated Development Environments similar to those for C and C++. The most common Java IDE's are Cafe and Visual Cafe (Symantec), J++ and Visual J++ (Microsoft), Latte (Borland), CodeWarrior (Metrowerks) and SuperCede (Asymmetrix).

JavaWorkshop™ and Visual Java™ (Sun) address larger-scale Java development projects. Java Workshop is written entirely in Java and is based on a web-centric user interface.

The Java development kit (JDK), from Sun, is available for several platforms. It includes the basic Java development tools like the Java Compiler, Java Debugger, Java Virtual Machine, AppletViewer and JavaDoc.

Unique to Java are so called Just-In-Time (JIT) compilers which convert neutral Java bytecodes in native machine instruction. JIT compiled Java runs more than an order of magnitude faster than interpreted Java. Typically, JIT compilers are included in the web-browser or the Java environment on the client-side.

Just like regular compilers, Java Native Compilers (Sun, Asymmetrix) translate and optimize Java programs directly to machine code for a specific platform. Natively compiled Java offers the best performance and is attractive where portability is not needed like server-based Java applications.

In addition to Java development tools, Java Components and Java Class Libraries are available from vendors such as RogueWave and the KL Group.

V. Issues Affection Java in EDA

At this time, there are a number of issues which hinder wide spread adoption of Java as primary development language for applications like EDA design tools.

The immaturity of the Java language, development tools, training and support are a temporary obstacle. The number of vendors offering Java tools and services is increasing rapidly and this situation is likely to improve further at a fast pace in the near future. As the Java language and Java API's are still evolving, major improvements and changes are still being made. Access to design specifications and early developer releases over the net address this issue to some extent.

Probably the main issue with Java is performance. Interpreting Java bytecodes is too slow for demanding applications like EDA tools. JIT and Java Native compilers will address the performance issue in the near future and are

expected to run Java applications at nearly the same speed as native code. In addition, Java hardware support like JavaChips will provide another performance boost.

Java includes a set of base API's and an increasing number of standard extension API's. Most of the extensions provide enterprise and network computing functionality. Some of the extension API's address design application requirements, for example the Java Media API will include better and faster 2D and 3D graphics support.

Missing at this time a licensing API. Newer licensing, metering and "pay-per-view" schemes are being discussed for Java. Until these licensing mechanisms are available and accepted, Java applications need to interface to existing licensing managers, such as FlexLM from Globetrotter Software.

VI. Java in EDA Today

While Java-based applications are not yet mainstream in the EDA community, a lot of research and development work taking place—demonstration application platforms being presented and small-to-medium scale commercial applications being announced.

A. PPP Project - Stanford

PPP is a web-based environment for Low-Power Design. Its graphical user interface is a set of dynamically generated HTML pages that can be accessed through any web-browser. Three sets of tools are available: Synthesis for low-power, Power Optimization and Power Simulation. File Transfer utilities are also available to upload input files and download results.

B. WELD Project - U.C. Berkeley

The WELD project aims to construct the first operational prototype of a national-scale CAD design environment enabling Internet-wide IC design for the U.S. electronics industry. WELD's goals are dual. In the small, WELD will empower individual American electronics designers by affording them efficient desktop access to, and seamless interoperability of, the numerous, heterogeneous resources forming a national scale electronics design system built upon the National Information Infrastructure.

In the large, WELD will reduce electronics industry market entry barriers to new entrepreneurs by providing a streamlined pay-per-use design development environment and a robust software distribution infrastructure. In reducing the costs, and shortening the time-to-market of new intellectual content, theWELD project expects to stimulate the U.S. electronics industry to dramatic new growth.

C. EDA Browser - Concurrent CAE Solutions

The EDA Browser from Concurrent CAE Solutions is the industry's first Java-based application for distribution of electronic design information. It utilizes standard Web based browsing technologies including Netscape™ 3.0 and HotJava™. The software is based on a client/server architecture, which allows for concurrent users to simultaneous visualize and distribute electronic design information across a local area network or the World Wide Web.

D. ProjectXView - Mentor Graphics

The ProjectXView application is an innovative new product that takes advantage of the network-aware and platform-independent computing paradigm made possible by Java.

The ProjectXView application compliments Mentor's WorkXpert Technical Project Management software. It provides web access to project status information, including roll-up reports across multiple projects, and also provides scheduling information that passes to and from commercially available scheduling tools.

VII. Conclusions

A paradigm shift in the EDA application environment as we know it today will happen. This shift is driven by the requirement for profitably addressing market windows with innovative new products. It is fueled by the pervasive Inter/Intranet infrastructure and the amazing market acceptance of the Java language and the Java Computing paradigm. The result of this shift will be an EDA environment that promotes a more collaborative, performance-oriented and cost-effective design capability. Mass deployment of the underlying Java technology in the EDA community is not without its own set of problems and issues that must be resolved. However, as we can see in the formative work from academia and the commercial world, it is only a matter of time.

References

A white paper on Java Computing is available from:
http://www.sun.com/961029/JES/whitepapers/

A brief summary of the Java language and some introductory Java examples can be found at:
http://www.sun.com/Solaris/products/javavm/prod_spec.html

The current set of Java API's together with the schedule and recent status is available at:
http://www.javasoft.com/products/apiOverview.html

For more information on the PPP synthesis tools, see:
http://akebono.stanford.edu/users/PPP/

For more information on the WELD project, see:
http://www-cad.EECS.Berkeley.EDU/Respep/Research/weld/

For more information on the EDA Browser from Concurrent CAE Systems, see:
http://www.ccaes.com

For more information on Mentor Graphics' WorkXpert product, see:
http://www.workxpert.com

Trademarks