Solaris OpenWindows

ToolTalk in Electronic Design Automation

A White Paper

Table of Contents

ToolTalk in Electronic Design Automation

Ricki Frankel

Executive Summary

Faced with increasingly complex design challenges, Electronic Design Automation (EDA) users are demanding greater cooperation among tools from multiple EDA vendors. At the heart of the framework technologies emerging to support these needs, is a flexible inter-tool communication (ITC) mechanism. ITC capabilities will become increasingly important to EDA tool developers, framework vendors and EDA users alike. ToolTalk™ is part of Solaris™ OpenWindows™, from SunSoft™. Solaris is a single, integrated distributed computing solution. It is comprised of SunOS™, ONC™, OpenWindows, DeskSet™, and OPEN LOOK®.

ToolTalk provides such a mechanism and introduces a new concept of support for network-wide application services. This support will become increasingly vital in the 1990s for the development, integration and use of tools in highly coordinated, multivendor tool sets. ToolTalk was demonstrated in the CAD Framework Initiative's 1991 Integration Project. It supports critical ITC requirements today and will allow the creation of highly interactive, multivendor tool environments based on the standard framework technologies of tomorrow.

Introduction

Increasing design complexity continually augments engineers' needs for comprehensive EDA solutions running on more powerful platforms. Yet, no single EDA vendor can serve the full breadth and depth of a design organization's tool requirements, much less satisfy their unique demands. Lacking common intertool mechanisms, corporate CAD departments and tool integrators have consequently been forced to use inflexible, tool-specific mechanisms to integrate and coordinate multivendor tool sets. SunSoft's ToolTalk provides a more general alternative for tool integration and interoperability — one able to support distributed client/server computing environments. ToolTalk is a highly flexible interapplication communication mechanism that lets individual tools request and provide services in the design environment without specific knowledge of the types or locations of other tools in the computing environment. ToolTalk thus serves as the foundation for multivendor, client/server EDA framework environments, as demonstrated in the CAD Framework Initiative (CFI) 1991 Integration Project.

The ability of different tools to exchange data and generally cooperate on broad design tasks has preoccupied the EDA industry for years. Rather than lacking available mechanisms, EDA tool developers have been faced with too many tool-specific mechanisms. Unfortunately, the available mechanisms support tool interaction at a very low level, with little of the application-level support required to create coordinated multivendor tool environments. In contrast, ToolTalk takes a dramatically different approach for tool interaction — one that raises tool developers' focus above low-level communication mechanisms and lets them concentrate instead on using and delivering application-level services.

ToolTalk Services

ToolTalk offers a high degree of flexibility for application interoperability not unlike the flexibility for personal communications provided by a public telephone network. Instead of tying two locations together with a physical wire, a public telephone network provides a "virtual connection" between telephone sets; an individual user "connects" with the public network through a particular telephone number that is registered with the network. Any calls placed from any other user in the network to that number are automatically routed to the appropriate telephone — all without any special need for action on the part of the receiver.

Existing methods for supporting tool interoperability require tool developers to agree on the details of their tool interfaces, as if they had to hook up a wire between their locations to communicate by telephone. Just as "virtual connections" between telephones have greatly simplified personal communications, ToolTalk greatly simplifies interapplication communications.

ToolTalk, however, dramatically extends this concept of a "virtual connection" between applications. Instead of building a specific link with a particular EDA tool, a ToolTalk-based application simply registers its interest in particular types of messages with the ToolTalk server — and then continues executing.

When messages of that type occur, the ToolTalk server notifies the application program, which can then take appropriate action. Furthermore, an application can send a ToolTalk message to request particular services without knowing the provider of those services, because the ToolTalk server will automatically notify the provider.

Application-Specific Protocols

ToolTalk is the vehicle to facilitate tool interaction — much as a public telephone is a vehicle for interpersonal communications. It's up to the users to determine what to say.

Thus, to create interoperable tools, application developers need only agree on standard message protocols — a set of high-level, well-defined, application-specific messages based on ToolTalk intertool communications mechanisms and services. A tool registers its interest in particular messages by giving the ToolTalk server a set of "attributes" that define the characteristics of messages of interest. When a message occurs, the ToolTalk server compares the attributes of the message with the attributes of interest to individual applications and notifies applications whose attributes match those of the message.

Attributes provide a fine-grained mechanism for defining, selecting and using highly specific application-level services. For example, an application server can go online by sending a message with attributes that define its interest in particular types of events. When client applications send messages with similar attributes, the application server will be notified, deliver its services and send a message indicating completion. The particular services, operations and notification procedures depend entirely on the message protocols defined by tool developers.

By using standard EDA-specific message protocols, EDA tool developers can support designers' specific application needs involving interactions between various classes of tools such as schematic editors, logic simulators, physical design tools, and others. EDA developers can facilitate engineers' tasks of creating timing-correct circuits, for example, by using a public message protocol that identifies circuit nets with timing violations. Here, a schematic editor could register an interest in information about nets with timing violations and continue supporting an engineer in modifying a schematic. Separately, a concurrently executing timing analysis program would issue messages identifying timing violations on particular nets in the engineer's design. With each message, the schematic editor could highlight the corresponding circuit nets on the schematic. The benefit of this type of intertool coordination is a more immediate feedback mechanism to the engineer, which decreases design errors and speeds the overall development process.

ToolTalk Benefits

ToolTalk's highly flexible, generalized approach to intertool communication offers multiple benefits to EDA tool developers and users alike. Because ToolTalk eliminates the need for specific, "hard-wired" connections between applications, tool developers can write applications that coordinate their operation with arbitrary tool sets. Instead of rewriting programs to support specific links with other applications, tool developers can create tools that more easily coordinate their activities in the users' environment.

For their part, users gain the power of integrated tool sets and a more flexible ability to select tools. Instead of choosing a tool on the basis of its specific links with other applications, users can select a tool based on its functional capability. The ToolTalk mechanisms will let the tool automatically interoperate with the other ToolTalk-based tools in the design environment.

As organizations face even more complex design challenges, uniform inter-tool communications mechanisms such as ToolTalk will prove critical in supporting distributed computing environments comprising multiple workstations. Using these scalable program execution resources, development organizations will be able to draw on multiple systems to support computationally intensive applications or scale them down to support multiple engineers with their individual needs. Furthermore, as next-generation, object-oriented architectures such as those proposed by the Object Management Group emerge, ToolTalk's internal object-oriented architecture will provide a smooth migration path to these sophisticated environments.

CFI 1991 Integration Project

ToolTalk's ability to support tool interoperability has already proven to be important in EDA tool integration. ToolTalk served as the key intertool communication (ITC) mechanism required to support the CFI 1991 Integration Project — the CAD Framework Initiative's demonstration of the use of standard interface specifications for multivendor tool integration and interoperability. Using the CFI's standard interfaces, multiple tool developers integrated their tools in a heterogeneous workstation environment and demonstrated their ability to share data, exchange control information and cooperate during various design activities.

Project Architecture

In the CFI 1991 Integration Project, a common framework uses a client/server architecture to support interactions between two types of workstation-based EDA tools — producers and consumers. Producer tools request services from a separate design representation (DR) server to create new data, while consumer tools use these same services to read data for analysis and display. This DR server was separate from but complementary with the CFI ITC mechanism.

At the heart of the integration project architecture, a ToolTalk-based ITC mechanism provides a virtual communications "bus" that lets developers "plug" their tools into the demonstration. (See Figure 1.) Users communicate with tools through the framework cockpit user interface, which invoke tools by transmitting messages via the ITC mechanism. For the purposes of the demonstration, a tool-control program, called the Surrogate Methodology Manager (SMM), transmits and receives messages to coordinate producer and consumer tools in an artificial sequence — just as a production control-flow management tool might orchestrate tool sequences in support of a design organization's actual development methodology. In addition, an ITC daemon monitor and design representation server monitor each record ITC and DR transactions, respectively, for later inspection.

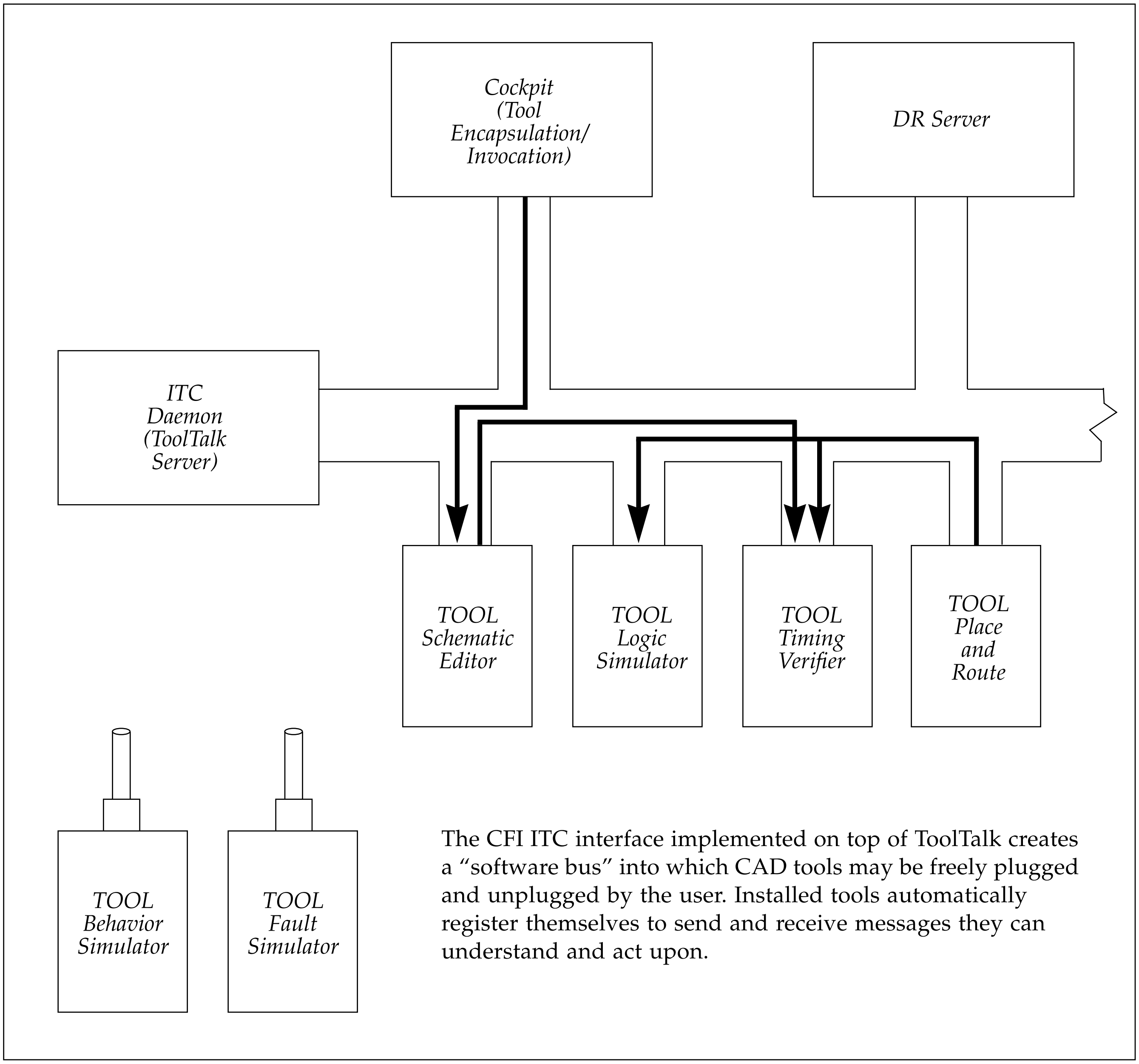

Figure 1 CFI ‘91 Integration Project

The CFI demonstration highlights the complementary nature of ToolTalk ITC capabilities with design representation services, which let applications share data in a repository such as a disk file or database management system (DBMS). While repositories let EDA tools work with the same data to complete design tasks, an ITC mechanism is necessary to support cooperation between the tools and to orchestrate their separate activities in complex tasks. For example, if the schematic editor example described previously did not have an ITC mechanism to notify it of nodes with timing violations, the schematic editor would need to scan some external file (or even the design database) to search for timing violation flags placed there by the timing analysis tool. At best, such a technique is cumbersome; at worst, its inflexibility would make it difficult to support designers' evolving needs. Thus, an ITC mechanism serves a critical role in coordinating flexible interactions among multiple tools in the EDA environment.

ToolTalk-Based CFI ITC Services

Because intertool communications are so fundamental to cooperative multi-tool environments, the supporting ITC mechanisms should be easy to use by tool developers. ToolTalk's simplicity permitted the rapid development of interapplication communications services compliant with CFI ITC specifications. From the time the CFI selected SunSoft's ITC technology, software engineers required only about six weeks to implement ToolTalk-based ITC services on the CFI demonstration's five different platforms: DECStation™ 5000/200, HP 433™, IBM RISC/System 6000™, Intergraph Model 6000™ and SPARCstation® 330.

Part of this process involved the development of an intermediate software layer that implemented the CFI ITC interface library used by EDA vendors in the CFI demonstration. In fact, this intermediate software layer proved to be a minimal development effort, because SunSoft's active participation in CFI activities helped ensure that ToolTalk mechanisms were fundamentally in agreement with CFI ITC interface specifications as they evolved.

As part of the CFI demonstration, SunSoft also worked with MCC's (Microelectronics and Computer Technology Corporation) original ideas and cooperation to develop a separate application called the ITC monitor. The ITC monitor audited ITC messages and provided a real-time display of messaging traffic to help attendees visualize the CFI demonstration's underlying ITC activity.

The final piece of the ToolTalk-based CFI ITC mechanism was the definition of a set of message protocols to support the particular needs of the CFI demonstration. Developed by the CFI ITC Technical Subcommittee (TSC), the CFI ITC message dictionary defines a set of message protocols required to support specific intertool operations, such as starting and stopping tools, reading and writing design data, highlighting nets, and more. (See Table 1.)

Table 1 Message Dictionary for DAC Demonstration

CFI 1991 Integration Project

Message	*Description*
ToolStarted	Informs client tools that a particular producer or consumer tool started running.
ToolFinished	Informs client tools that a particular producer or consumer tool stopped running.
ReplyStatus	Informs client tools that a particular consumer has successfully initiated execution.
DesignOpened	Informs client tools that a particular design data set was opened for access.
DesignChanged	Informs client tools that a particular design data set was modified.
DesignWrite	Requests a producer to begin processing (writing to) a particular design data set.
DesignWriteDone	Informs client tools that a particular design-write operation has been completed.
DesignRead	Requests a consumer to begin processing (reading from) a particular design data set.
DesignReadDone	Informs client tools that a particular design-read operation has been completed.
Highlight	Requests client tools to highlight a particular net.
HighlightOff	Requests client tools to end highlighting of a particular net.

CFI ITC Deployment

Starting in mid-February, 1991, SunSoft engineers began delivering the actual ToolTalk ITC interface library code to participating EDA developers. In the first actual integration effort, two engineers required only two days to integrate CFI ITC capabilities into the European Jessi CADLAB framework. Subsequent integration efforts at CFI headquarters in Austin, Texas, also proceeded smoothly. EDA developers participating in the CFI demonstration required little effort to use the ToolTalk-based CFI ITC mechanisms.

In the CFI demonstration, a particular EDA tool uses a two-stage process to interact with the ToolTalk ITC service. In the initial stage, the EDA tool registers its interest in events in the environment. Here, the tool uses the ITC library to pass a list of attributes that characterize the tool's interest in ToolTalk messages that will occur during the demonstration. The next stage comprises the tool's specific interactions with other tools. Here, the CFI ITC library provides information on message activity forwarded by the ITC server.

CFI Scenarios

The CFI 1991 Integration Project demonstration provides various runtime scenarios to illustrate typical tool interactions in an EDA environment. In each case, the ToolTalk ITC mechanism plays a pivotal role in supporting interoperability among the multiple EDA tools from different vendors.

Among the various scenarios, one illustrates the complementary nature of ITC and remote procedure call (RPC) mechanisms. Here, individual producer and consumer tools use RPC mechanisms to access the separate DR server, which in turn reads and writes *netlist* data on behalf of the requestor. RPC effectively extends the CFI DR procedural interface (PI) across the network, permitting high-speed access to the dedicated DR server. Thus, these RPC mechanisms form a tight link between the client and server halves of the same interface.

ToolTalk ITC mechanisms come into play by providing a highly flexible communications link between participating applications. In this same scenario, the ITC mechanisms specifically permit the SMM (Surrogate Methodology Manager) to orchestrate the orderly access of individual producers and consumers to the netlist data. A consumer tool could coordinate its action with the producer directly by registering for the message (*DesignWriteDone*) that the producer sends when it completes. This approach would, however, embed some knowledge about the producer in the consumer, creating some degree of direct coupling between the particular consumer and corresponding producer tool. In effect, the SMM further decouples the consumer from knowledge about the producer.

With ToolTalk, EDA tool integrators can easily extend this approach to support more complex tool workflow sequences. With conventional ITC approaches, tool integrators would be forced to use specialized interface mechanisms to hard-wire specific tool interactions in a multivendor tool environment. ToolTalk, however, enables a broad set of tool interactions, and the user is permitted to choose which specific tool interactions to invoke — all within the broad context of an overall workflow. Thus, ToolTalk's generalized intertool communications services provide users with maximum flexibility in the use of multivendor tools, while permitting CAD managers to exercise the level of control they require over the overall workflow.

Another scenario highlights the specific benefits of inter-tool communications in coordinating "real-time" cooperation among multiple tools from different vendors. Here, tools capable of highlighting circuit nets register their interest in the Highlight message. When the user selects a circuit net for highlighting, a Highlight message alerts all interested tools, which in turn use their internal capabilities to highlight the appropriate circuit net on the display.

ToolTalk's flexible notification mechanism permits the activities behind this scenario to remain intuitively simple. In contrast, conventional integration mechanisms would require specialized, tool-specific interface mechanisms to support this conceptually simple highlighting operation. Similarly, integration approaches that tried to pass this information through shared data representations would easily become bogged down, since the tools looking to highlight the user-selected nets would be forced to scan large databases or external files, looking for special flags indicating the nets of interest.

Conclusion

Designers' greater reliance on multivendor tool environments dictates the need for powerful, yet flexible, intertool communications services to facilitate tool interoperability. ToolTalk responds to this need with mechanisms that let diverse EDA tools interoperate through high-level, application-specific messaging capabilities. ToolTalk eases tool integration, because its simplicity lets EDA tool developers exploit its services with minimal code changes.

Furthermore, ToolTalk lets users move easily to more powerful computing environments and computational paradigms. As design organizations move to more powerful distributed computing client/server environments, ToolTalk's inherent client/server architecture easily spans networks. Thus, design environments can scale up to distributed computing environments, yet retain their investment in ToolTalk-based tools. ToolTalk servers automatically extend their scope to all clients in the network that register interest in particular events. No code modification is necessary because the ToolTalk mechanisms let applications interoperate without regard to the specific network configuration — or even to the specific application tools with which they cooperate.

Similarly, ToolTalk simplifies the migration of design organizations to more powerful computing paradigms such as object-oriented, message-based methods. ToolTalk's internal message-based architecture facilitates migration to object-oriented methods as they emerge from organizations such as the Object Management Group.

As shown in multivendor tool interoperability demonstrations such as the CFI 1991 Integration Project, ToolTalk simplifies tool integration and interoperability. For EDA users, the availability of this interapplication communications mechanism promises a freer choice of design tools and more effective use of individual tools within well-orchestrated tool sets.

SunSoft, Inc.
2550 Garcia Avenue
Mountain View, CA 94043

For more information, call 1 800 227-9227.

Printed in USA 9/91 91032-0 1.5K