OpenDesign: An Open User-Configurable Project Environment for Collaborative Design and Execution on the Internet

H. Lavana F. Brglez
CS Dept., NCSU
Raleigh, NC 27695

R. Reese
ECE Dept. MSU
Starkville, MS 39762

G. Konduri A. Chandrakasan
EECS Dept., MIT
Cambridge, MA 02139

http://www.cbl.ncsu.edu/OpenProjects

Abstract – OpenDesign *is an open user-configurable project environment that supports distributed collaborative design and execution on the Internet. The environment is created by* configuring *a generic client for a specific project. This is in contrast to an implementation of a project-specific client-server architecture.*

This paper introduces the OpenDesign environment in the context of a design process and project-specific tasks. An OpenDesign task *is defined as execution of one or more CAD point tools, whereas a* task flow *is a dependency graph of tasks and/or other task flows. Challenges arise when, within a single project, (1) tasks must be executed on remote hosts under different file systems, (2) data must be accessed, moved, modified, and archived with consistency, (3) tasks and task flows are assigned to more than one designer, and (4) designers are physically dispersed. In collaboration with peer institutions, a number of demo design projects demonstrate the features and the opportunities with the OpenDesign environment.*

Keywords: collaborative computing, design, Internet.

1 INTRODUCTION

The Internet is rapidly opening new opportunities not only in ways of how designers access tools and data but also how new design projects are defined and executed. While universities addressing the Internet design environments may lead in technical publications [1, 2, 3, 4], the special DAC99 issue of EE Times [5] lists 14 articles, contributed by industry leaders in EDA, that foretell of "the quiet revolution in the way the electronic products are designed – and best is yet to come." These predictions are on target: see [6] for an example of a web-site providing the pay-per-use service since March 2000.

The GUIs of the three clients in Figure 1 are representative of the current generation of Internet-based tools, and also provide the context for this paper:

- *JavaCADD client* has been developed at MSU [2]. It features a simple-to-use front end to submit various design representations (verilog, VHDL, ...) to a number of commercial tools (simulator, logic synthesizer, place & route tool, ...). The interface is simpler to learn and use when compared to interfaces of original tools. This client has been used successfully in a number of VLSI design classes at MSU.
- *WebTop client* is a hierarchical schematics editor developed at MIT. It evolved into CollabTop where two or more participants can collaboratively edit a schematics and use the file to drive other tools [4].
- *ToolWire client* represents a commercial product [6]. The free demo version of the tool paces the user to click through a sequence of tasks, using the four icons shown in the upper-left corner: (1) upload file (in VHDL), (2) analyze file, (3) synthesize an FPGA device (as per 'choice' window), (4) generate a report.

In the strict sense, only the *JavaCADD* is a point tool. Both *WebTop* and *ToolWire* allow the user to *manually* follow-up with another task without leaving the environment. In both cases, the choices have been predefined by the client's developer, *not* the client's user. Furthermore, none of the clients support automated chaining of tasks nor a collaborative environment to share and archive data at a common site – an important factor when a number of designers may be participating in a joint project.

The major goal of OpenDesign is to allow users to configure their environment:

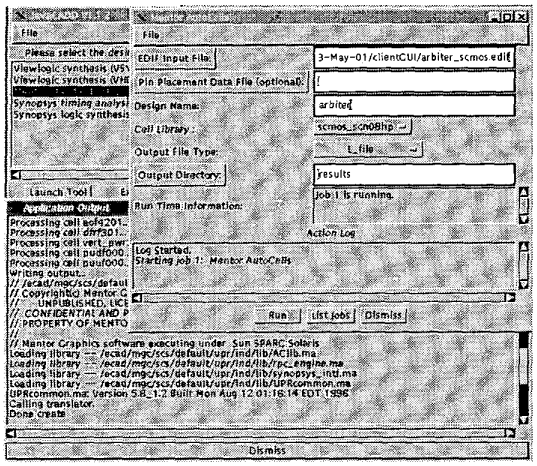
- by choosing the best or the most affordable tools for each design tasks – without having to install them on local host;
- by choosing the most effective sequences of tasks to be executed, not only for manual one-task-at-a-time execution but also for scheduling any number of tasks for automated execution;
- by choosing the hierarchy of data structures and revision control most appropriate for the project-at-hand – and having it readily accessible on the Web;
- by creating and storing preferences on the modes of collaboration among the members of a project team.

Our approach to devising the OpenDesign environment reflects its goal as listed above. Rather than implementing a dedicated client/server environment where software developers make most of the choices of how the environment is to be used by designers, we rely on *a generic client interfaced to a generic server* and let the designers/users implement the environment themselves

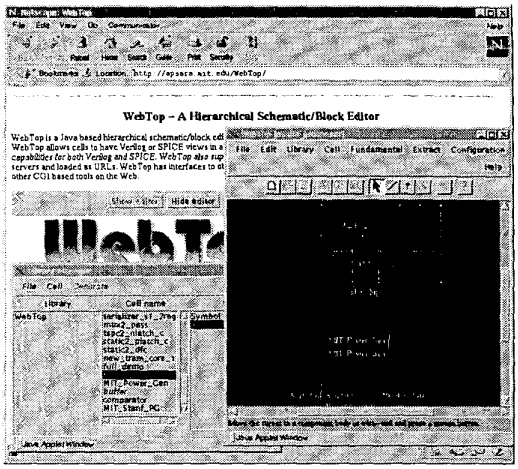
*Authors have in part been supported by contracts from DARPA/ARO (P-3316-EL/DAAH04-94-G-2080 and DAAG55-97-1-0345).

0-7695-0801-4/00 $10.00 © 2000 IEEE

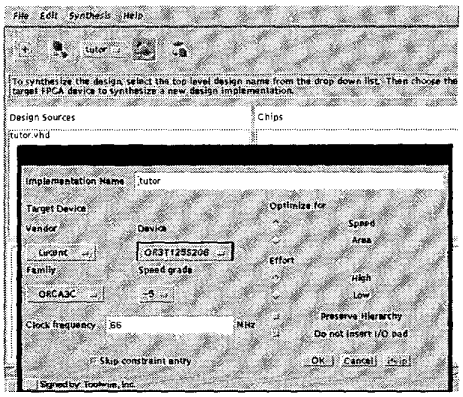
(a) JavaCADD client: accessing Mentor layout



(b) WebTop client: accessing PowerPlay analysis



(c) ToolWire client: accessing FPGA synthesis



(a) *JavaCADD client* has been developed at MSU [2]. It is basically a simple-to-use front end to submit various design representations (VHDL, verilog, edif, ...) to a number of commercial tools (simulator, logic synthesizer, place & route tool, ...).

(b) *WebTop client* is a hierarchical schematics editor developed at MIT. It evolved into CollabTop where two or more participants can collaboratively edit a schematics and save the file to drive other tools [4].

(c) *ToolWire client* represents a commercial product that has been accessible on the Web since March 2000 [6]. The free demo version of the tool paces the user to click through a sequence of tasks, using the four icons shown in the upper-left corner: (1) upload file (in VHDL), (2) analyze file, (3) synthesize an FPGA device (as per 'choice' window), (4) generate a report.

Fig. 1. State of-the-art point tool clients on the Internet.

by writing a simple configuration file and a set of encapsulation scripts. As long as each point tool *also* can be invoked remotely with a command-line script rather than the nominal GUI, our demos show that a number of project-specific environments can be readily created by re-configuration of the the generic client. The key elements of the generic client-server/peer-to-peer OmniFlow/OmniDesk architecture we now use were initiated during a class project and are described in [7, 8].

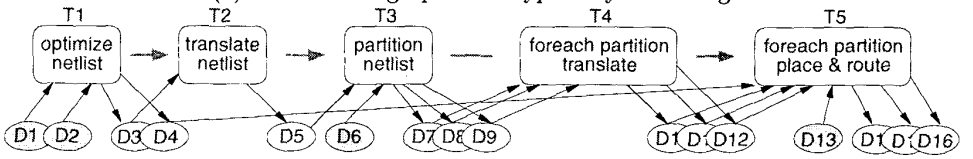
A number of complementary project environments are being created with OmniFlow/OmniDesk: OpenDesign as described in this paper as well as OpenExperiment and OpenWriter [9]. All environments have similarities in what we designate as *nominal view*, with data subdirectories explicitly linked to most if not all tasks. The nominal view is also convenient when exploring the project requirements in the start-up phase. However, during the course of most projects, version control becomes an important issue, a subject of on-going research. The next sections highlight the concepts and the implementation of the OpenDesign environment, and conclude with a brief outline of collaborative demos with peer institutions.

2 OpenDesign: Nominal Concepts

A *task-data graph* captures the tasks and data dependencies in a design. It is a directed bipartite graph such as shown in Figure 2a. For simplicity, we show an example where chaining all task dependencies creates a *single path.* The path may be broken into a number of *segments*; there are two segments in the example. The first segment consists of three *simple tasks: optimize netlist, translate netlist, partition netlist.* The second segment consists of two *iterative tasks: foreach partition translate, foreach partition place & route.* Note however, the data dependencies. Some tasks depend on primary input data only, some tasks depend on data generated by the preceding task only, but in general, tasks may depend on data generate by any task preceding the one being executed.

The task-data graph in Figure 2a is representative of a typical design flow. As such, it defines the objectives of a small multi-team *project.* Challenges that are faced by the team are: (1) resources to execute most tasks are available only on remote hosts under different file systems, (2) data must be accessed, moved, modified, and

(a) A task-data graph for a typical system design.



Note: D1, D2, D6, D13 are primary input files

(b) An example of OmniFlow/OmniDesk client/server configuration.

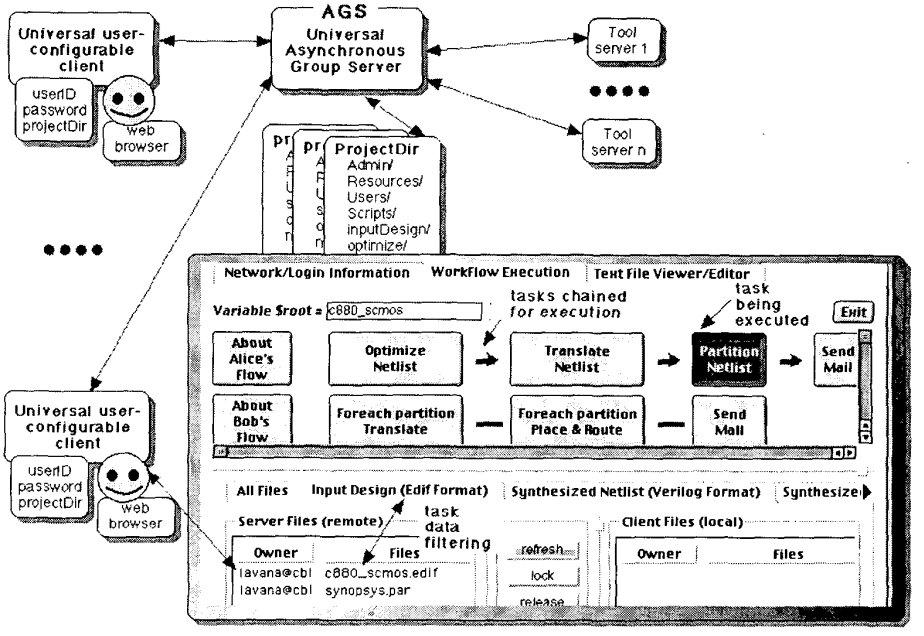


Fig. 2. A task-data graph example and its OpenDesign environment implementation.

archived with consistency, (3) tasks and task flows are assigned to more than one designer, and (4) designers are physically dispersed. At this point, project coordinator can use the client-server architecture of OmniFlow/OmniDesk and to collaboratively configure OmniFlow clients that best meet the objectives of the project.

The basic concepts of the OmniFlow/OmniDesk architecture are shown in Figure 2b: application-transparent universal and asynchronous group server (AGS), tool servers readily accessible from AGS, AGS-based project directories, each created initially by the project coordinator.The project directory (*projectDir* on the server) has a number of subdirectories that are project-specific and are devised by the team in the course of the project. Only project coordinator has read/write (r/w) permissions to Admin and Resources. Team members can read Resources as well as any Users subdirectory. Each team member has r/w permission in their own subdirectory.

The client received by each team member is enabled once the user enters *userID, password,* and *projectDir.* Once enabled, the client configures to a default view that may invoke a pre-configured *task flow* such as shown in Figure 2b. The facility to organize and configure a set of design objectives into a number of task flows is one of the important *generic features* of this client. For example, `Alice's Flow` involves three tasks that are to be executed in the following sequence:

```
optimize      translate      partition
 netlist  ->  netlist   ->   netlist
```

Clicking on each of the task buttons invokes a tool that may reside on a local host or a remote host. Files produced by one tool may or may not be used as inputs to the tool that is to be executed later. After completion of a task, user may click on the next button in the sequence shown. Alternatively, the interface allows user to configure on any of the *task connectors* that can be toggled between `--` and `->` such that a complete *task chain* gets invoked on the click of the first button in the chain. This is the state of `Alice's Flow` shown in Figure 2b; the sunken state of the third button (technology mapping) indicates that the task is being executed.

The files read and written by specific tasks can be filtered for display in the respective file boxes: one box

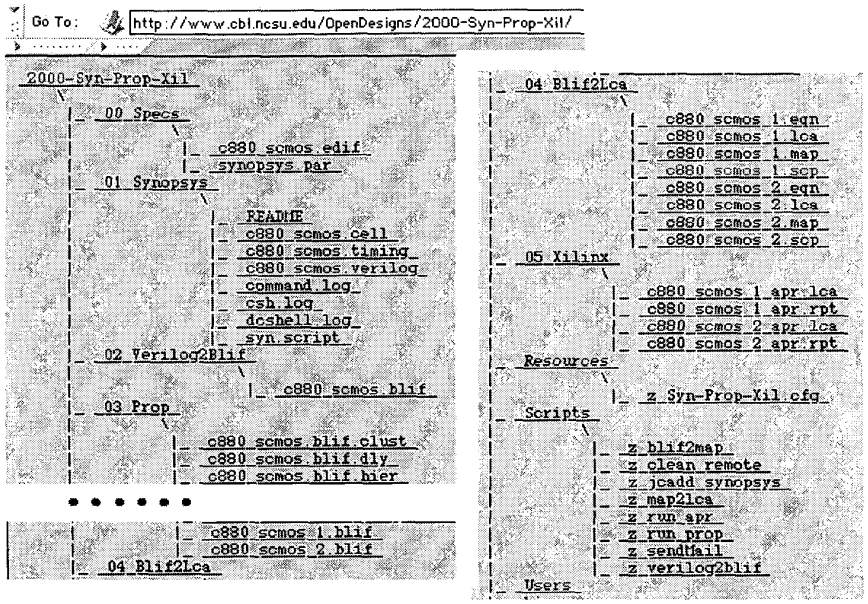


Fig. 3. Related to client in Fig. 2: an open project directory *after* completion of all tasks.

displays the working directory on the remote host, the other on the local host. Notably, if the file is owned by the server (remote host), ownership can be asserted for each file through the interface: either by clicking on the owner (listed as server) or by highlighting the file in the filebox and clicking on the *lock* utility button between the fileboxes. Similarly, only file owner can relinquish the ownership to the server, at which point it can be claimed and edited by another team participant.

3 OpenDesign: Experiments

The configuration file to render a project-specific interface such described in the previous section is typically less than one page long. It is written in a subset of Tcl [10]. This subset is simple and intuitive: no prior experience with Tcl is required to write such a file. A total of 4 lists are entered not only to create the entire client GUI but also to link executions of all tasks to the objects rendered by the GUI. For more details and related publications, see [9].

A number of collaborative design flows, similar to the one in Figure 2 have been tested between host servers running at MSU, MIT, and NCSU as part of the DARPA-sponsored Vela Project [11]. Demos with participating students took place in the University Booth during DAC'99 and DAC'00. A typical session would engage two students invoking the OmniFlow client on two workstations, capturing and jointly editing a netlist schematic using WebTop at MIT, optimizing the schematic with Synopsys tool at MSU, invoking partitioner on the netlist at NCSU and placing and routing all partitions using the Xilinx tool at NCSU. The directory structure that archives the results of the experiment is automatically posted on the Web, as shown in Figure 3.

References

[1] H. Lavana, A. Khetawat, F. Brglez, and K. Kozminski. Executable Workflows: A Paradigm for Collaborative Design on the Internet. In *Proc. of the 34th Design Autom. Conf.*, pages 553–558, June 1997. See `http://www.cbl.ncsu.edu/-publications/#1997-DAC-Lavana`.

[2] D. Linder, R. Reese, J. Robinson, and S. Russ. Java-CADD: A Java-based Server and GUI for Providing Distributed ECAD Services. Technical Report MSSU-COE-ERC-98-07, Microsystems Prototyping Laboratory, April 1998. See `http://WWW.ERC.MsState.Edu/mpl/-publications/papers/javacadd/JCaddTR.pdf`.

[3] F. Chan, M. Spiller, and R. Newton. WELD - An Environment for Web-Based Electronic Design. In *Proceedings of the 35th Design Autom. Conf.*, pages 146–152, June 1998.

[4] G. Konduri and A. Chandrakasan. A Framework for Collaborative and Distributed Web-Based Design. In *Proceedings of the 36th Design Autom. Conf.*, pages 898–903, June 1999.

[5] The Internets Impact on Engineering EE Times, special supplement on 'web-based engineering', Issue 1066, June 21, 1999, pages 85–112. See http://www.eet.com.

[6] ToolWire, The Electronic Design Workbench: Pay-per-use Online EDA - Anytime, Anywhere. March 10, 2000. For more information, see `http://www.ToolWire.com`.

[7] F. Brglez, H. Lavana, Z. Fu, D. Ghosh, L. I. Moffitt, S. Nelson, J. M. Smith, and J. Zhou. Collaborative Client-Server Architectures in Tcl/Tk: A Class Project Experiment and Experience. In *Seventh Annual Tcl/Tk Conference*. USENIX, February 2000. See `http://www.cbl.ncsu.edu/-publications/#2000-TclTk-Brglez`.

[8] F. Brglez and H. Lavana. CollabWiseTk: A Toolkit for Rendering Stand-alone Applications Collaborative. In *Seventh Annual Tcl/Tk Conf.*. USENIX, February 2000. See `http://www.cbl.ncsu.edu/publications/-#2000-TclTk-Lavana`.

[9] Home page for OpenProjects environments. See `http://www.cbl.ncsu.edu/OpenProjects` for more information. .

[10] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[11] Vela Project on Collaborative Distributed Design: New Client/Server Implementations EE Times, special supplement on 'web-based engineering', Issue 1066, June 21, 1999. See http://www.eet.com and http://www.cbl.ncsu.edu/-publications_misc/1999-EETimes-vela.pdf.