WADE : A Web-based Automated electronic Design Environment

Dhruva R. Chakrabarti *Pramod G. Joisha* *Prithviraj Banerjee*

Technical Report No. CPDC-TR-97-08

Center for Parallel and Distributed Computing
Department of Electrical & Computer Engineering
Northwestern University
Technological Institute
2145 Sheridan Road, Evanston IL-60208

WADE : A Web-based Automated electronic Design Environment

Dhruva R. Chakrabarti (dhruva@ece.nwu.edu)

Pramod G. Joisha (pjoisha@ece.nwu.edu)

Prithviraj Banerjee (banerjee@ece.nwu.edu)

Center for Parallel and Distributed Computing
Electrical and Computer Engg. Department
Northwestern University
2145 Sheridan Road
Evanston, Illinois 60208

Abstract

In this paper, we have presented a novel framework of a Web-based automated electronic design environment. The design files of a user working in a remote machine are transparently shipped to our local Compute Center, the relevant computations are performed in a parallel environment and then the results are returned back to the user. The whole process is done efficiently and transparently without the user having to bother about any low-level details. The goal of this project is to make available to the CAD tool users some of the core parallel applications like ProperVHDL, ProperSIS and ProperPLACE. At present, the ProperPLACE tool has been made available on the Internet for use by the CAD community. Results from a preliminary implementation of a parallel placement tool are impressive and show the feasibility of the approach.

This research was supported in part by the Semiconductor Research Corporation under Contract SRC 95-DP-109, the Advanced Research Projects Agency under contract DAA-H04-94-G-0273 administered by the Army Research Office and the National Science Foundation under contract NSF MPI 93-20854.

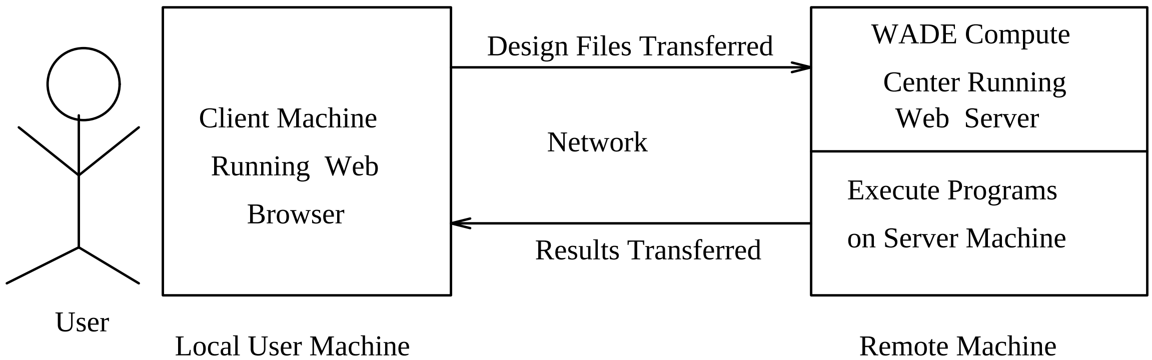

Figure 1: Approach 1

1 Introduction

Computer-aided design (CAD) tools have become a crucial technology for the design of electronic systems. Owing to the complexity of current electronic systems, the computational requirements for these *CAD* tools have also increased. This makes techniques for the acceleration of the electronic design process a significant and growing concern for digital circuit design. In addition, increasingly large complex designs are being done by groups of people physically located in distant places. At the same time, as *CAD* tools are becoming a critical portion of circuit design methodologies, CAD companies are seeking newer pricing models for providing their software. Instead of selling CAD software for a per-seat or per-license basis, newer pricing techniques like a per-use policy are being considered. A pricing technique like per-use will need methods that can guarantee delivery of service to users while keeping track of the count of actual use. Since this implies that the tool cannot be delivered to users on a time-bound basis, users should be able to access the software remotely and efficiently when the need arises. With the growth in high-speed networks, this concept is becoming more and more relevant as access to computing power and information resources scattered physically are now available at people's desktop.

The goal of the WADE [Web-based Automated electronic Design Environment] project is to develop a prototype of a self-contained, automated CAD Computing Center accessible transparently to everyone on the Internet. Users can specify the target technology, design constraints and upload their design files to the WADE Compute Center. Software at the Compute Center can apply parallel and distributed computing resources automatically to the CAD task, producing a circuit implementation as quickly as possible. The resulting circuits can then be returned back to the user. This approach is illustrated in Figure 1. Alternatively, users can choose to run an existing software from the WADE Compute Center on his design files on his own machine. This approach is illustrated in Figure 2. On the user's request, applications that reside in centralized servers in the Compute Center will be shipped transparently through the Internet as Java applets, perform the development at the user's machine and terminate. Such applications can be developed which will work on any machine without any modification whatsoever. Both the above methods support inherently the per-use pricing policy since powerful authentication mechanisms can be built into the application which will authenticate and allow access to only qualified users. The present situation of this ongoing work supports the first scenario. A framework has been developed through which users authenticate themselves, upload their design files to our Compute Center and receive the results back. All these are done transparently and efficiently over the Internet. This approach is appropriate when programs are considerably complex and difficult to write in Java and when execution requires specialized hardware such as parallel machines.

The aim of this work is to make available to the CAD tool users some of the core parallel applications [3, 5]. They include **ProperVHDL** [6], **ProperSIS** [7] and **ProperPLACE** [1]. *ProperVHDL* is a parallel version of a behavioral simulation tool which reads in a design using VHDL descriptions at the RTL and behavioral levels, partitions the design automatically among different processors and then simulates the design. *ProperSIS* consists of parallel algorithms for combinational and sequential logic synthesis based on **SIS**, which takes logic designs at the VHDL RTL level and generates a netlist of gates mapped to a particular technology. *ProperPLACE* consists of various versions of parallel cell placement using simulated annealing based on **TimberWolf6.0** which takes either the output of logic synthesis or a design in *EDIF* or TimberWolf netlist format and produces a layout.

Section 2 outlines some of the related work based on this idea. Section 3 briefly explains an example application based on *properPLACE*. This example will be used throughout the paper to explain the various issues involved. Section 4 gives an overview of the Web interface while Section 5 explains the implementation issues in more detail. Section 6 presents conclusions.

2 Related Work

Several test beds for distributed design tools and algorithms have recently been proposed. These include the *WELD* project [20] which seeks to enable electronic design over the Internet. By providing to the EDA community toolkits, protocols and interfaces, the *WELD* team is trying to establish a common, open framework which designers and tool suppliers can easily enhance and add to. A *Cell Characterization* software from *Mississippi University* characterizes a cell over varying slew rates and output capacitive loads. It returns an accurate macromodel. A Web-based environment for low-power design [21] has been reported by Stanford University. Its graphical user interface is a set of dynamically generated HTML pages that can be accessed through a Web browser. Other relevant test beds are *PowerPlay* [17], *Validity Checker* [22], *Speedie* [23], Biology Workbench [18] etc.

While previous research tries to integrate the Web-based Internet technology with conventional CAD tools running on conventional machines, we are the first to propose Web-based tools for *PARALLEL CAD*. This is especially relevant since not many organizations have access to dedicated parallel machines. Our tools will provide a general infrastructure that is adaptable to various IC design applications. Furthermore, the goal of the project is to provide an automatic software to evaluate the tasks to be performed and decide what mix of tools and algorithms on a particular platform will produce the desired results efficiently and successfully. Another advantage of such a tool is that a new per-use pricing model

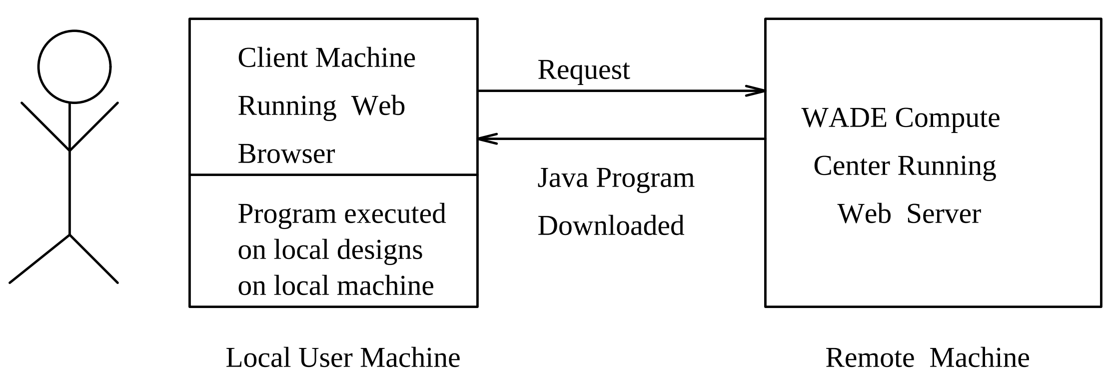

Figure 2: Approach 2

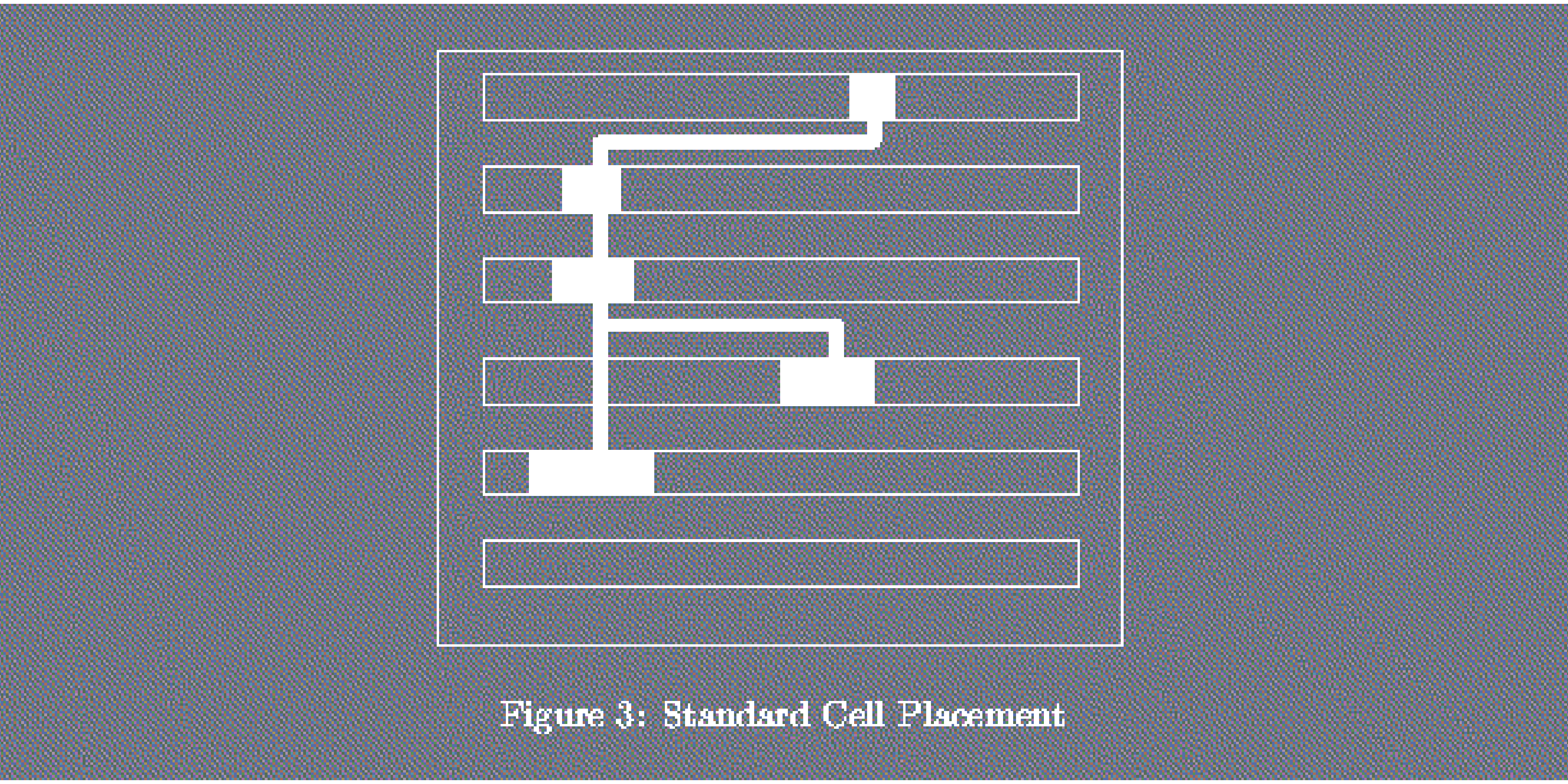

Figure 3: Standard Cell Placement

Figure 3: Standard Cell Placement

can be incorporated within it.

3 An Example Application

Standard cell based design methodology allows a designer to build his or her design from a library of predefined modules or cells. The placement problem involves placing these cells on a VLSI layout, given a netlist that provides the connectivity between each cell and a library containing layout information for each type of cell. This layout information contains the width and height of the cell, the location of each pin, the presence of equivalent pins and the possible presence of feed through paths within the cell. The primary goal of cell placement is to determine the best location of each cell so as to minimize the total area of the layout and the length of the nets connecting the cells together. With standard cell design, the layout is organized into equal height rows, and the desired placement should have equal length rows, as shown in Figure 3. One of the powerful algorithms for standard cell placement is simulated annealing. Since VLSI cell placement lacks good heuristic algorithms, it is a good application for simulated annealing. Simulated annealing is an iterative optimization software strategy that starts with a system in a disordered state, and through perturbations of the state, brings the system gradually to a low energy and thus optimal state [8, 9].

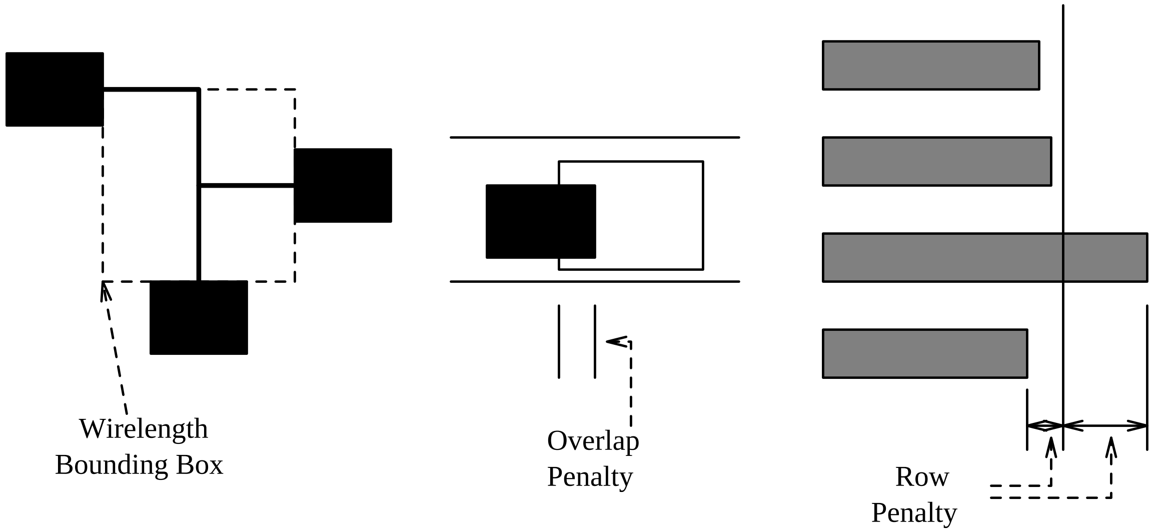

Figure 4: TimberWolf Cost Function Components

3.1 TimberWolfSC

The *TimberWolfSC* cell placement tool [12] uses simulated annealing. The TimberWolfSC cost function is defined below.

$$C = W + \mu P_0 + \lambda P_R$$

W is the bounding box half parameter estimate of the wirelength of all nets, P_0 is the penalty for the area overlap between cells in the same row, and P_R is the penalty for the difference between the actual row length and the desired row length. These cost parameters are shown graphically in Figure 4.

Moves are generated by choosing a random cell and then displacing it to a random location on the layout. If a cell is already present at the new location, the two cells are exchanged. A temperature dependent range limiter is used to limit the distance over which a cell can move. Initially, the span of the range limiter is set such that a cell can move anywhere on the layout. Subsequently, the span is decreased logarithmically with temperature. These range limited updates are made at the end of each of the 160 iterations into which TimberWolfSC segments the simulated annealing procedure. As the algorithm progresses, the temperature is gradually decreased by forcing the acceptance rate to follow a theoretically derived schedule that attempts to keep the acceptance rate close to 44% during the middle region of annealing [13]. TimberWolfSC6.0 also uses row bins to aid in the computation of overlap and row penalties, and early rejection methods are used to speed up the decision process [14].

3.2 ProperPLACE : A Set of Parallel Implementations of Placement

Because of the inherent computational costs associated with simulated annealing, several methods have been proposed for the parallelization of this procedure. The goal of *ProperCAD*, of which *ProperPLACE* is a part, is to develop portable parallel algorithms for VLSI CAD applications that will run on a range of parallel machines including shared memory multiprocessors such as the Sun SparcServer 1000E and the SGI Challenge, distributed memory multicomputers such as the Intel Paragon, IBM SP-2, Thinking Machines CM-5 and networks of workstations.

We have developed four different algorithms [16] for standard cell placement using simulated annealing all of which are based on the Timberwolf 6.0 placement program. The first algorithm called

ProperPLACE-PM uses a parallel moves approach where each processor gets a copy of the entire circuit but they perform moves of cells independently. Speedups of 3.1 to 3.7 have been reported on 8 processors of an Intel Paragon, with quality degradation of about 5 to 9%. The second algorithm called ProperPLACE-MMC uses a multiple Markov chain approach which is based on the observation that the search path of a simulated annealing strategy is basically a Markov Chain. In the parallel algorithm, each processor gets a copy of the entire circuit, but each processor performs an independent search of different Markov chains starting with different seeds, with 1/P number of moves performed on each of the P processors, with periodic update of the best placement information among the processors. This algorithm gives nearly linear speedups with very little loss of quality. Speedups of 6.1 to 6.3 have been reported on 8 processors of an Intel Paragon with qualities that are within 3 to 5% of the serial algorithm. This approach is suitable for small scale multiprocessors and for circuits that are small enough to fit in the memory of a single node. The third algorithm called ProperPLACE-SC uses a speculative computation approach. One can view simulated annealing as a decision tree of moves, and in this approach, two different sets of processors are assigned to speculatively work on two different possibilities on each decision level. Even though the qualities of the results are identical to the serial algorithm, the speedups were sublinear, i.e. 0.2 on 4 processors of a SUN4 shared memory multiprocessor. The fourth and final algorithm called ProperPLACE-PART addresses the memory scalability problems by partitioning the circuit across the nodes. Each processor performs annealing based moves on cells in its current set of cells but moves them all across the circuit. Periodically, the cell partitioning across the processors is changed. This circuit-partitioned approach provides speedups to larger number of processors with little loss of quality with excellent memory scalability. Speedups of about 6 on 8 processors of a SPARCSERVER-1000 and about 11 on 32 processors of a CM-5 have been measured for the algorithm. The quality of the solutions of the parallel algorithm are within 5% of the sequential algorithms. The strength of this memory scalable algorithm is that one can run placement on circuits that do not fit on the memory of a single processor. All the algorithms have been implemented using the ProperCAD environment, and the circuit-partitioned work has also been implemented using the Message Passing Interface (MPI).

4 Overview of the Web Interface

The present implementation of the *Web Interface* involves three distinct machines, a *client or user*, a *Web Server* and a *dedicated parallel machine*. The user requests for a document [at present, this document is at the address http://cpdcser.ece.nwu.edu:8080/WADE/wade.html] from the Web Server and gets it displayed on his own machine. In this case, this document introduces the user to the Web Interface, the way it works and the choices offered to the user. Starting from this page, the user may choose to open a new account or access his old account, if any. User identification is necessary to provide privacy, protection of data and resources and to enable the software to track each individual's usage of the software. Once the user authenticates himself, he is allowed to proceed to the next document which allows him to upload his files to the design center. He is also allowed to change the user data parameters and program control parameters in order to suit his requirements. This document is a simple *HTML* document which gives an extremely easy interface to the user to upload his files by just mentioning the file names from his local machine. Once his design files are submitted, the user is given a unique job number so that he can access the results of the particular submission later. We assume that the job requested by the user is of considerable computational complexity and may take a sizeable time to complete. The Web server collects the data submitted by the user, transfers it to the server on the dedicated parallel machine and returns. The server process on the parallel machine receives the data and spawns off a child process to handle the execution of the program. Once the program completes, the results are sent back to the Web Server in a particular secure area accessible only to the corresponding user. Figure 5 shows the interaction between the different processes while Figures 6, 7, 8 show a particular sequence of steps through which a user may progress.

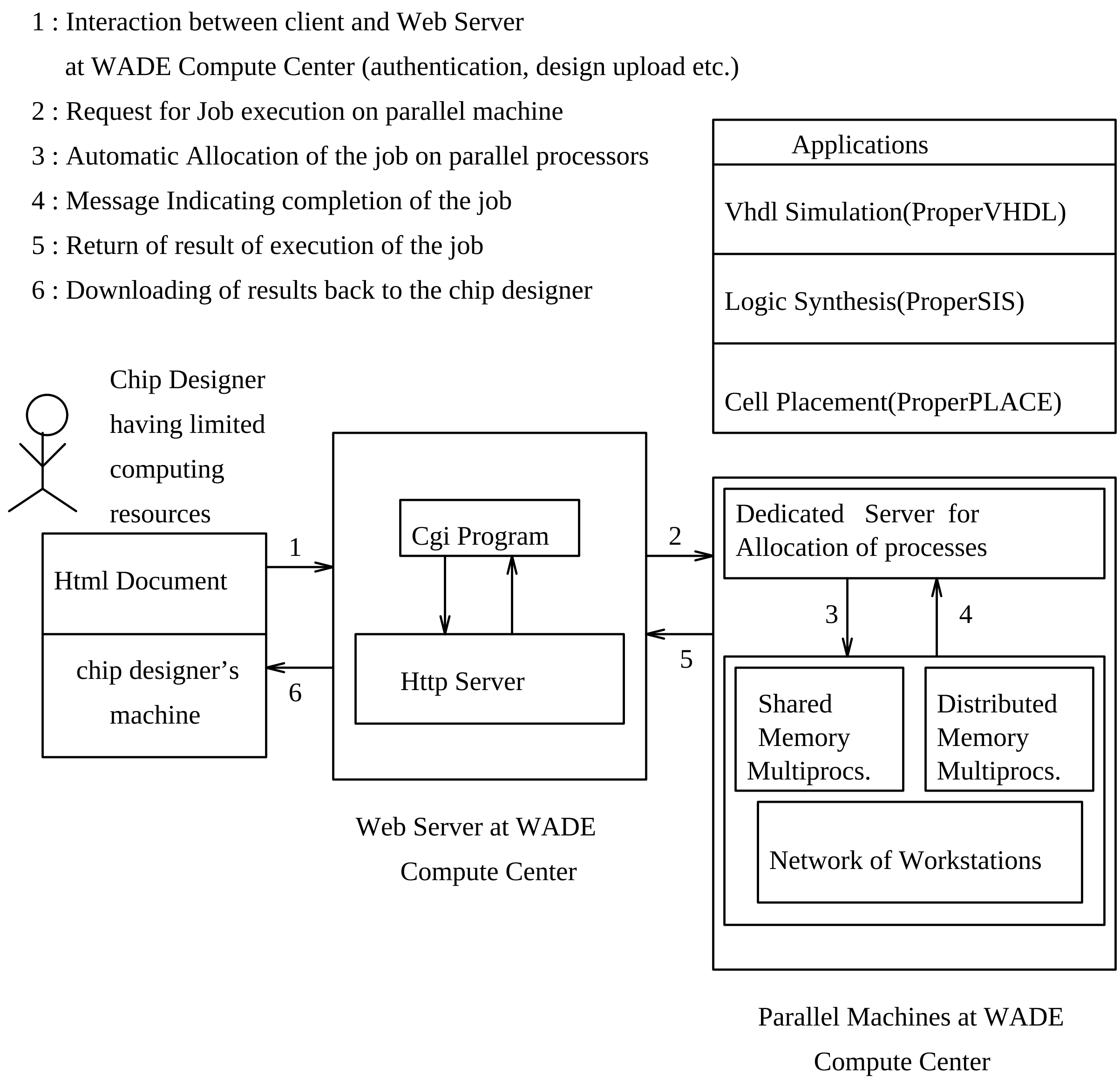

Figure 5: Overview of the WADE project

At the time of writing the paper, the *ProperPLACE-PART* software has been made available to the user and so the sequence of steps are specific to *ProperPLACE-PART* and very similar to the inputs required by *TimberWolfSC6.0*. The ProperPLACE tool accepts 3 files as input : a *par* file, a *cel* file and a *blk* file. The *par* file contains parameter specifications for the ProperPLACE system. This parameter file consists of two parts : the design rule parameters and the program control parameters. The design rule parameters contain definitions of layers, the associated parasitic resistance and capacitance, the connections between layers, minimum routing width of the layers, the

Figure 6: Authentication

minimum distance between any defined layers etc. The second part of the parameter file is devoted to program control parameters and has the format *progname*parameter : value*. The parameters are usually *vertical_wire_height*, *vertical_path_weight*, *rowSep*, *padspacing*, *graphics.wait*, *random.seed*, *chip.aspect.ratio*, *default.tracks.per.channel*, *feedThruWidth* etc. The *cel* file contains the descriptions of the standard cells (row-based-cells), macro cells and pads and the netlist. Standard cells precede the macro cells which precede the pads. The *blk* file contains the block class definitions. The complete BNF for the system can be found in [19].

5 Implementation Details

The implementation makes use of *HTML*, *CGI scripts* and *Java scripts*. The CGI scripts are written in the programming language *C*. The following subsections enumerate the primary issues and the way they have been handled in the present implementation.

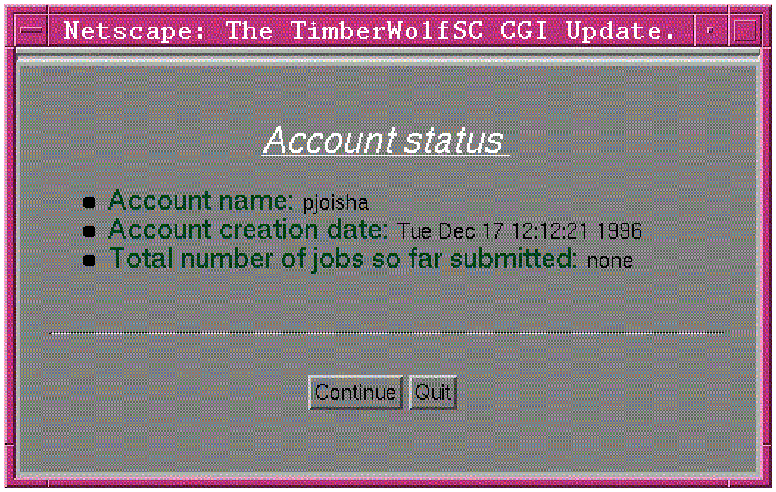

Figure 7: Status Display

5.1 Security

Authentication of the user is necessary so as to prevent misuse of resources and protection of user data. This is also in keeping with the per-use pricing policy discussed in the previous sections. In the present implementation, users are required to authenticate themselves before they are able to access either the software or existing data. It is ensured that users acquire access to data only if they enter the correct password. Each user's data is stored in an area to which no one else can have access. Furthermore, each user is assigned a job number so that he can submit a number of designs to the same software. However, each such access is tracked and controlled by custom CGI scripts.

5.2 File Uploading

An easy, user-friendly and transparent way to transfer the design files from the client machine to the *Web server* in the *Compute Center* is necessary. Since current *Netscape* browsers don't allow files to be read or written from client machines through Java applets, we have used the *HTML file upload* facility to accomplish this task. Most current browsers, with the exception of Netscape 2.0 and above, do not provide a mechanism by which a user agent can communicate a file as input to the remote CGI script [24]. However, the Netscape HTML extensions observed in Netscape 2.0 and above, allow for the inclusion of file uploading as an input in forms. This is accomplished by setting the *TYPE* attribute of the *INPUT* tag in forms to *FILE* and by specifying the encoding type to a format relevant to file submission. This feature has been standardized in HTML+, HTML 3.0 and HTML 3.2. By indicating the encoding type to *multi-part/form-data* (enctype = multipart/form-data), the user agent (WWW browser in this case) transfers the data across to the remote server using a suitable MIME representation. The file upload widget is displayed by the browser as a window containing previously selected filename, if any, and a *Browse* button for new selection. Selecting the browse button causes the browser to enter into a file selection mode appropriate for the platform. Window-based browsers pop up a file-selection window where the user has the option of replacing the current selection or choosing a new selection. It is possible to constrain the file types allowed to be uploaded. Specifically for the ProperPLACE-PART program, which currently accepts data in TimberWolf6.0 format, there are 3 input files, a *cel* file, a *blk* file and a *par* file. All these are ASCII files. These files can reside in a local area of the user's machine.

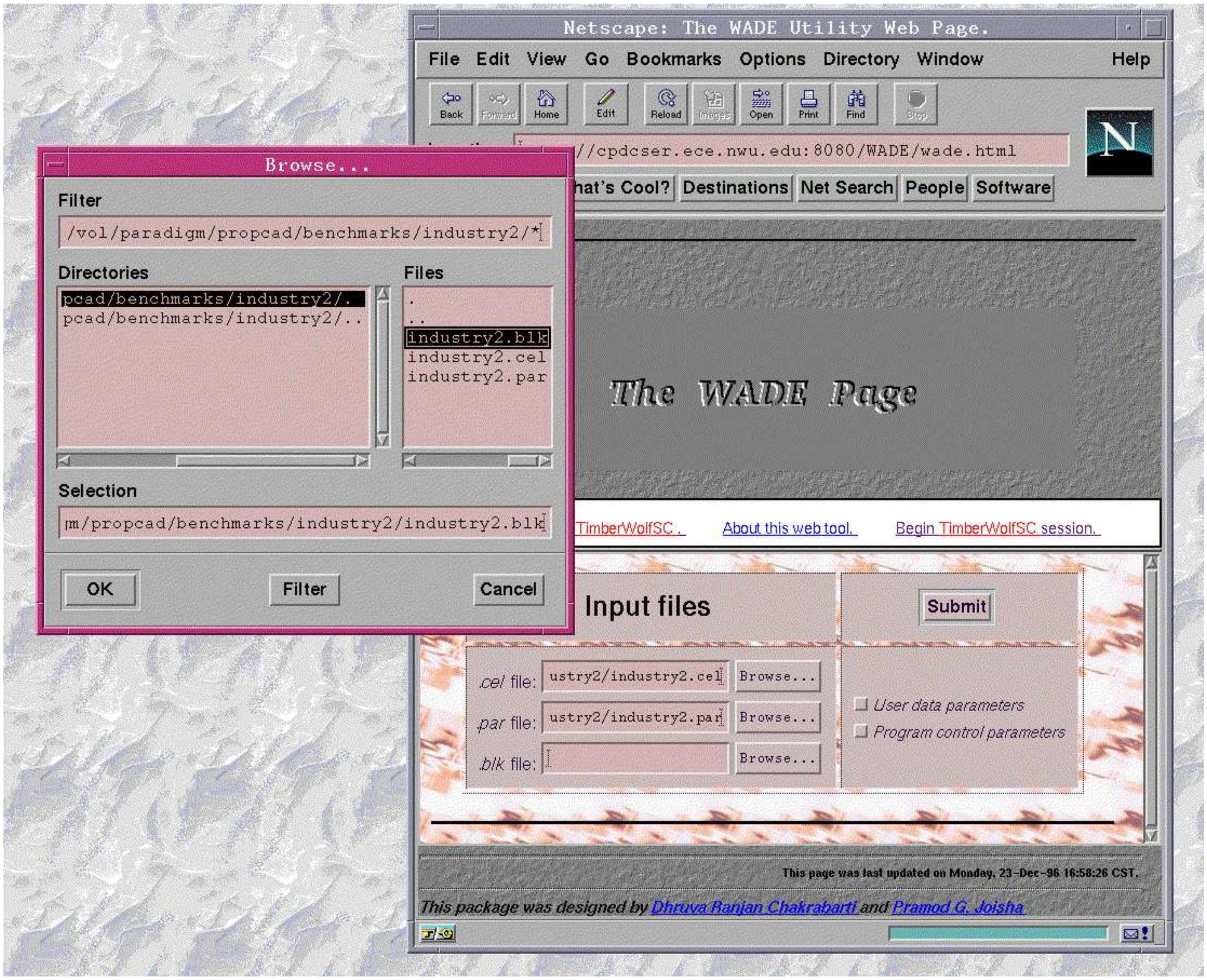

Figure 8: File Upload

When the user completes the form and selects the *Submit* button, the browser sends the form data and the content of the selected files to a specified CGI script. The CGI script does the appropriate parsing to strip off the headers and boundaries from the received data and retrieves the appropriate files.

This scheme has been chosen not only for its simplicity but also for the flexibility it gives to the client in the number and types of files it can send to the server. At the same time, it gives the server the the authority to decide which files to accept. This is a natural way to upload files transparently over the Internet from the client to the server. Security issues like snooping on the net by a third party to extract the file contents have been investigated but not yet been implemented. Appropriate measures to encrypt data will be considered later.

5.3 The Web Server

A *CGI program* residing on the *Web server* [cpdcser.ece.nwu.edu] handles the data once it is received from the user. It does appropriate processing of the data to retrieve the actual file contents. Then it opens a socket connection to the server waiting on the dedicated parallel machine. The *Web server* acts as a client to the server on the parallel machine. Once the server on the parallel machine finishes execution, the results, including any error conditions are returned to the Web server. The state of the execution is maintained in the Web server and is accessible to the corresponding client.

5.4 The Server on the Parallel Machine

The present implementation employs a concurrent server running on an 8 processor IBM J-40 shared memory multiprocessor [blueser.cpdc.ece.nwu.edu]. The server process remains running all the time waiting for new connection requests from the Web server. The server forks off a child process to handle each new connection and allows the child process to handle execution as well as communication with the Web server. A concurrent implementation of the server is essential in order to handle arbitrary number of client requests at the same time and also to preserve acceptable performance in events of complex computation and large data transfer. The server is *stateless* since it does not store the result or status of a transaction with the client. While this may affect performance adversely, this is essential since the parallel machines are dedicated for computation and it is a separate machine, the Web server, which interacts with the outside world. The server automatically places the job requested on the 8 processor IBM J-40 [blueser.cpdc.ece.nwu.edu] and a 16 processor SP-2 [speth**.cpdc.ece.nwu.edu] distributed memory multicomputer as it deems necessary.

5.5 Sequence of steps in a transaction

- Authentication
  - In case of access of an existing account, validate password
  - In case of new user, open a new account for the user
- Assignment of a unique id. to the current access of the software
  - A unique id. is generated using the login name and the access count of the concerned account
  - A secure area on the filesystem is alloted to the user which can be accessed only by the corresponding user. This portion of the filesystem is not visible to the outside world and can be accessed only via custom CGI scripts.
- Upload design files
  - The CGI script at the Compute Center reads from standard input to receive the files.
  - The script reads in the input length from the environment variable content_length.
  - Along with the data, a boundary, which does not occur in the data is received at the server.
  - Each field of the form is received in the order in which it occurs in the form, as a part of the multipart/form-data stream.
  - Each part received contains the name of the file and the appropriate content-type. Since the input files for the CAD applications supported need to be ascii files, only ascii files are accepted by the CGI program.

- The CGI script opens a socket connection to the server residing on the front-end of the parallel machine, transfers the unique id. for the current request and the data received as a stream of bytes.
- In case the user wants to override the design parameters specified in the files by some other data, those overriding parameters are also passed on to the server on the parallel machine.

- Server Processing for automatic allocation of jobs on multiple processors
  - Once a request arrives, the server forks off a child, allows the child to handle the request and returns, again waiting for another request
  - The child process receives the stream of bytes from the Web server, extracts the files depending on the boundary received, does appropriate processing to strip off the headers and stores the files using the unique id. received from the Web server.
  - The job is then allocated on the available processors automatically by a loadleveler.
  - Once execution is complete, the result files are copied over to the Web server in the area assigned to the corresponding user

- Return of Results
  - The user may choose to be notified of the completion of his job or he can access his account to check for the completion of his job.
  - The user authenticates himself, accesses his old account, views the results and downloads the result files.

5.6 Performance and Results

The question of performance and efficiency naturally arises owing to remote file transfer and remote execution. But considering the complexity of CAD tools, it is evident that the network delay won't be a crucial factor. This is why use of multiple processors to solve CAD problems acquires a greater importance in this scenario. The present implementation makes use of the tool *ProperPLACE* with inputs in the form of three files, the actual cell description file, the parameter file and the block class definition file. The user is also allowed to change the data and control parameters. The present implementation of the interface has been tested using standard benchmark circuits (Table 1) with the user in the **University of Illinois at Urbana-Champaign** and the Web server and parallel machines at **Northwestern University**. The results are illustrated in Table 2. The results amply show that the network delay is negligible compared to the total run time and a sizeable speedup is maintained even for remote execution.

6 Conclusions

In this paper, we have presented a novel framework of a Web-based automated electronic design environment. The design files of a user working in a remote machine are transparently shipped to our local Compute Center, the relevant computations are performed in a parallel environment and then the results are returned back to the user. The whole process is done efficiently and transparently. Results from a preliminary implementation of a parallel placement tool are impressive and show the feasibility of the approach.

Circuit	Number of cells	Number of nets	Initial wirelength
fract	125	163	12839
struct	1888	1920	388707
primary1	760	1172	1202241
primary2	3014	3817	7292946
industry1	2271	2583	5301571
avq.large	25114	25384	129120230

Table 1: Circuit Information

Circuit	Total Data Transmitted (Kbytes)	Local Exec Time of TWSC (sec.)	Local Exec Time of mpiPLACE 1 proc. (sec)	Local Exec Time of mpiPLACE 8 proc. (sec.)	Remote Exec Time of mpiPLACE 1 proc. (sec.)	Remote Exec Time of mpiPLACE (8 proc.) (sec.)
fract	75	205	242	51	255	65
struct	783	1221	1327	271	1352	297
primary1	1020	408	482	94	500	115
primary2	1464	3299	3947	721	3980	793
industry1	1814	1717	2145	412	2160	515
avq.large	20579	78742	86522	17012	87400	17981

Table 2: Comparison of runtime

References

[1] S. Kim, J. A. Chandy, S. Parkes, B. Ramkumar and P. Banerjee, *ProperPLACE : A portable parallel algorithm for cell placement*, Proceedings of the International Parallel Processing Symposium, Cancun, Mexico, Apr. 1994, pp. 932-941.

[2] S. Parkes, J. A. Chandy and P. Banerjee, *A library-based approach to portable, parallel, object-oriented programming : Interface, implementation and application*, Proceedings of Supercomputing '94, Washington, DC, Nov. 1994, pp. 69-78.

[3] P. Banerjee, *Parallel Algorithms for VLSI Computer Aided Design Applications*, Englewood Cliffs, NJ, Prentice Hall, 1994.

[4] S. M. Parkes, *A class library approach to concurrent object-oriented programming with applications to VLSI CAD*, Ph.D. dissertation, University of Illinois at Urbana-Champaign, Sept. 1994, Tech. Rep. CRHC-94/20/UILU-ENGG-94-2235.

[5] B. Ramkumar and P. Banerjee, *ProperCAD : A portable object-oriented parallel environment for VLSI CAD*, IEEE Transactions on Computer-Aided Design, vol. 13, no. 7, pp. 829-842, July 1994.

[6] V. Krishnaswamy and P. Banerjee, *Actor Based Parallel VHDL Simulation Using Time Warp*, Proceedings of the 1996 Workshop on Parallel and Distributed Simulation, Philadelphia, PA, May 1996.

[7] K. De, J. A. Chandy, S. Roy, S. Parkes and P. Banerjee, *Portable parallel algorithms for logic synthesis using the MIS approach*, Proceedings of the International Parallel Processing Symposium, Santa Barbara, CA, Apr. 1995, pp. 579-585.

[8] S. Kirkpatrick, C. D. Gelatt and M. P. Vechhi, *Optimization by simulated annealing*, Science, vol. 220, no. 4598, pp. 975-986, 1984.

[9] R. A. Rutenbar, *Simulated annealing algorithms : An overview*, IEEE Circuits and Devices, vol. 5, no. 1, pp. 19-26, Jan. 1989.

[10] S. Kirkpatrick, *Optimization by simulated annealing : Quantitative study*, Journal of Statistical Physics, vol. 34, pp. 975-986, 1984.

[11] E. H. L. Aarts and P. J. M. van Laarhoven, *Statistical cooling : A general approach to combinatorial optimization problems*, Philips Journal of Research, vol. 40, no. 4, pp. 193-226, 1985.

[12] C. Sechen and A. Sangiovanni-Vincentelli, *The TimberWolf placement and routing package*, IEEE Journal for Solid State Circuits, vol. SC-20, no. 2, pp. 510-522, Apr. 1985.

[13] J. Lam and J. M. Delosme, *Performance of a new Annealing Schedule*, Proceedings of the Design Automation Conference, 1988, pp. 306-311.

[14] C. Sechen and K. W. Lee, *An improved simulated annealing algorithm for row-based placement*, Digest of Papers, International Conference on Computer-Aided Design, Santa Clara, CA, Nov. 1987, pp. 478-481.

[15] D. R. Greening, *Parallel Simulated Annealing Techniques*, Physica, vol. D42, pp. 293-306, 1990.

[16] J. A. Chandy, *Parallel Algorithms for Standard Cell Placement Using Simulated Annealing*, Ph.D. dissertation, University of Illinois at Urbana-Champaign, July. 1996, CRHC-96-10/UILU-ENGG-96-2216.

[17] D.Lidsky and J.M.Rabaey, *Early Power Exploration-A World Wide Web Application*, 33rd Design Automation Conference, Las Vegas, 1996, NY, USA. (URL http://infopad.eecs.berkeley.edu/PowerPlay)

[18] Shankar Subramaniam, *Biology Workbench, URL http://bioweb.ncsa.uiuc.edu.*

[19] *TimberWolf : Mixed Macro / Standard Cell Floorplanning, Placement and Routing Package*, Yale University, Sept. 1991.

[20] Richard Newton, *URL http://www-cad.eecs.berkeley.edu/Respep/Research/weld.*

[21] *URL http://snf.stanford.edu/staff/losleben*

[22] *URL http://agamemnon.stanford.edu/ levitt/vc.html*

[23] *URL http://speedie.stanford.edu*

[24] *URL http://www.ics.uci.edu/pub/ieff/html/rfc1867.txt*