Copyright © 1980 American Telephone and Telegraph Company
THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 59, No. 9, November 1980
Printed in U.S.A.

Designer's Workbench:

The User Environment

By J. R. BREILAND and R. A. FRIEDENSON

(Manuscript received May 1, 1980)

Designer's Workbench, an interactive approach to integrating design aids, overcomes most impediments that normally restrict the use of these aids. Written under the PWB/UNIX* *operating system, these programs smooth the flow of data between the application programs that reside on various computer systems and are used to aid in the design of transmission equipment. This paper describes the user environment. It includes a description of the design technologies, processes, and aids and presents the Designer's Workbench approach to improving the acceptance and efficiency of those aids.*

I. INTRODUCTION

Designer's Workbench (DWB) integrates the design aids used at two transmission area locations of Bell Laboratories into a uniform system.[1] The set of design aids at the Holmdel, New Jersey location is primarily used for computer architecture and printed circuit board design. The set of aids at the North Andover, Massachusetts location is used to design custom electronics from the silicon chip through the printed circuit board interconnection. Before Designer's Workbench, these aids consisted of more than 40 programs running on 12 different computer configurations at 9 physically separate locations.

A major reason for the lack of acceptance of many design-aids systems is that little attention has been given to the needs and limited level of computer expertise of the user community. The users' requirements have been the controlling force in the implementation of Designer's Workbench. The capabilities and functions they require are extensive. Foremost, they need a fully integrated set of application programs for electrical design, test development, and physical design.

* UNIX is a trademark of Bell Laboratories.

Besides the necessary application programs, they require:

(*i*) Ease of data entry.

(*ii*) Ease of data modification.

(*iii*) Automatic naming and maintenance of internal files.

(*iv*) The ability to move data from one computer to another.

(*v*) The ability to have data translated from one application program to another.

(*vi*) Information about missing and inconsistent data before an application program is initiated.

(*vii*) Automatic formating of the job control language (JCL) for the various application programs and machines.

(*viii*) Interpretation and cross-referencing of their output data.

(*ix*) Interactive dialogue at multiple skill levels.

(*x*) A self-learning environment with on-line manuals.

The needs of the departments that support design-aids systems also affect the user environment. One goal is to reduce the dependence on consultants, or "gurus," for each application program that is supported. Other goals include the support of users who possess a wide diversity of skill levels, speeding the access by users to new applications programs that were not developed at the local site, and introducing bug-free enhancements to existing software.

This paper first explores the state of the design-aids used before this project was undertaken. The viewpoint is the users, that is, the electrical designers, physical designers, and test developers. Then the present user environment under DWB is discussed, including the DWB tutorials, data entry and modification, job preparation, job submission, and postprocessing of output data. The paper concludes with user feedback, user statistics, and future directions.

II. TECHNOLOGY FOR TRANSMISSION EQUIPMENT

Telecommunications equipment for digital transmission contains a wide range of both electrical and interconnection technology.

2.1 High volume custom electronics

Equipment that interfaces voice frequency lines contains hybrid transformers, relays, plugs, jacks, analog filters and amplifiers, signaling control circuitry, encoders, decoders, multiplexers, demultiplexers, compressors, expanders, etc. Although most of these functions are implemented with analog circuitry, significant portions are realized with digital circuitry. The logic structure of this circuitry is generally random, asynchronous, and sequential. Because of the large manufacturing volumes, custom, rather than off-the-shelf, electronics is used. The custom digital technology includes TTL, IIL, ECL, and MOS.

Because of the few codes and large manufacturing volumes involved, the line interface product includes a level of interconnection of electronics not generally found in other types of equipment. Most other equipment interconnects the electronics at the silicon level and the printed circuit board level. However, there is an intermediate level of interconnection in this product. This intermediate level of interconnection is the interconnection of digital devices, linear devices, thin-film resistors, and/or thin film capacitors on ceramic modules. These modules are called hybrid integrated circuits (HICs). Thus the design process to realize a circuit pack involves three levels of electrical and physical design: the silicon integrated circuit (SIC) level, the hybrid integrated circuit (HIC) level, and the circuit pack level.

The advantages in choosing a customized electrical and interconnection approach include:

(*i*) Lower interconnection cost.
(*ii*) Lower power consumption.
(*iii*) Higher speed electronics.
(*iv*) Higher packing density.
(*v*) Higher reliability.
(*vi*) Lower manufacturing cost because of automation of processes.

On the negative side, development times are generally longer and capital investment is generally higher.

2.2 Electronics for low volume control functions

Transmission equipment that interfaces digital lines or switching equipment is generally realized in a much more structured form than the voice frequency interfaces. The circuit packs contain DIPs, microcomputers, and memories, and are characterized by common hardware, custom firmware, and many different codes. At times, custom MSI and LSI will appear on the packs. The advantages of this approach include:

(*i*) Fast turnaround of designs.
(*ii*) Quick response to engineering changes.
(*iii*) Manufacturing economies by use of high volume components.
(*iv*) Greater commonality of design aids.

III. DESIGN PROCESS

Although the design processes for the above two equipment architectures and interconnection technologies are similar, there are basic differences in the sequence and way these steps are carried out. These differences are mainly due to technological constraints and the availability of design tools (with their consultants). The major steps in the design process are:

(*i*) Design
  (*a*) Architectural level
  (*b*) Functional level

(*c*) Algorithmic level
(*d*) Logic level
(*ii*) Testability analysis and incorporation
(*iii*) Design verification
(*a*) Hardware simulation (breadboard)
(*b*) Software simulation
(*iv*) Timing analysis
(*v*) Test development
(*vi*) Partitioning, placement, and layout
(*vii*) Processing and assembly
(*viii*) Prove-in testing on test machine
(*ix*) System test, evaluation, and modification
(*x*) Documentation for manufacture
(*a*) Schematic diagram
(*b*) Physical design information
(*c*) Test information
(*d*) Consistency checking.

The design process can be split into three major phases, the simulation and breadboard phase, the test development for manufacturing phase, and the physical design and documentation phase. Engineers who take the full custom route must repeat each phase in the process for each level in the hierarchy (SIC, HIC, circuit pack).

IV. DESIGN AIDS

The design aids for digital electronics have been developed by many different computer-aided design groups over the past decade. Some of these groups are internal to Bell Laboratories, others are outside suppliers. This section describes the state of design aids in the transmission area of Bell Laboratories in early 1978.

4.1 Circuit pack design aids

The circuit pack design process is illustrated in Fig. 1. Coding of the engineer's circuit sketch in the LSL circuit description language[2, 3] is the entry point for the design aids system. The inclusion of physical design information along with the interconnectivity information enables the creation of the HIWIRE[4] data base for that board. This data base is used as input to the programs that create information to drive either an automatic wirewrap machine or a semi-automatic quick connect machine for quick turnaround breadboard models. The HIWIRE coding can be used as input to the LAMP[2, 3] logic simulator for design verification and test development. When the test data are combined with the physical design information provided in the HIWIRE description, diagnostics can be created for guided fault isolation.

When the electrical designer is reasonably confident that his bread-

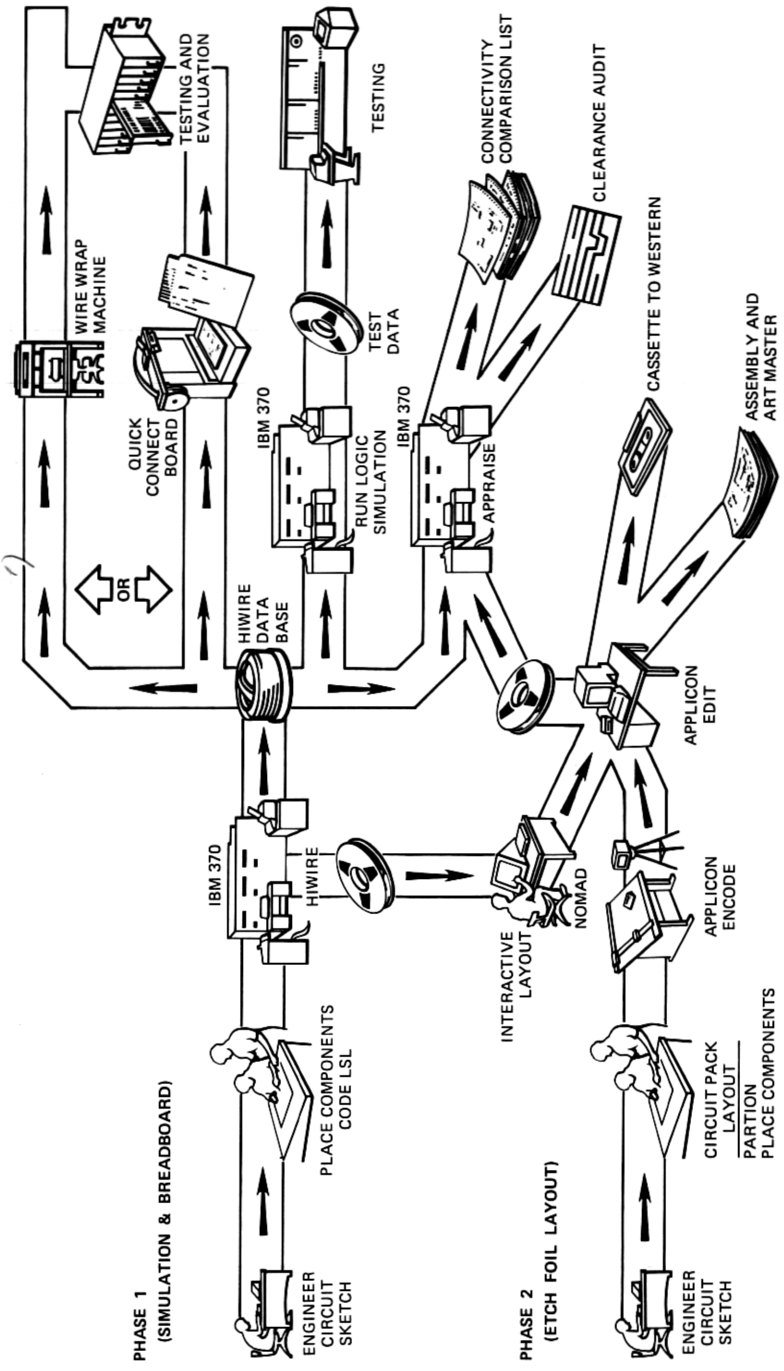


Fig. 1—Circuit pack design process.

board meets the design intent, layout for manufacture begins on a computer-aided layout and documentation system. To ensure that the layout agrees with the schematic, a connectivity comparison is performed between the initial LSL coding and the metal-to-metal interconnectivity derived from the layout data base.[5] Only when the two are in agreement are assembly and art master drawings created.

The file structure, libraries, and flow of information between major programs for the electrical portion of this design process is shown in Fig. 2. Note that there are five libraries (package, component, I/O pins, simulation, and test) that must be created and maintained, and numerous internal files whose data must be compatible with more than one program. A list of the functions performed in this process,

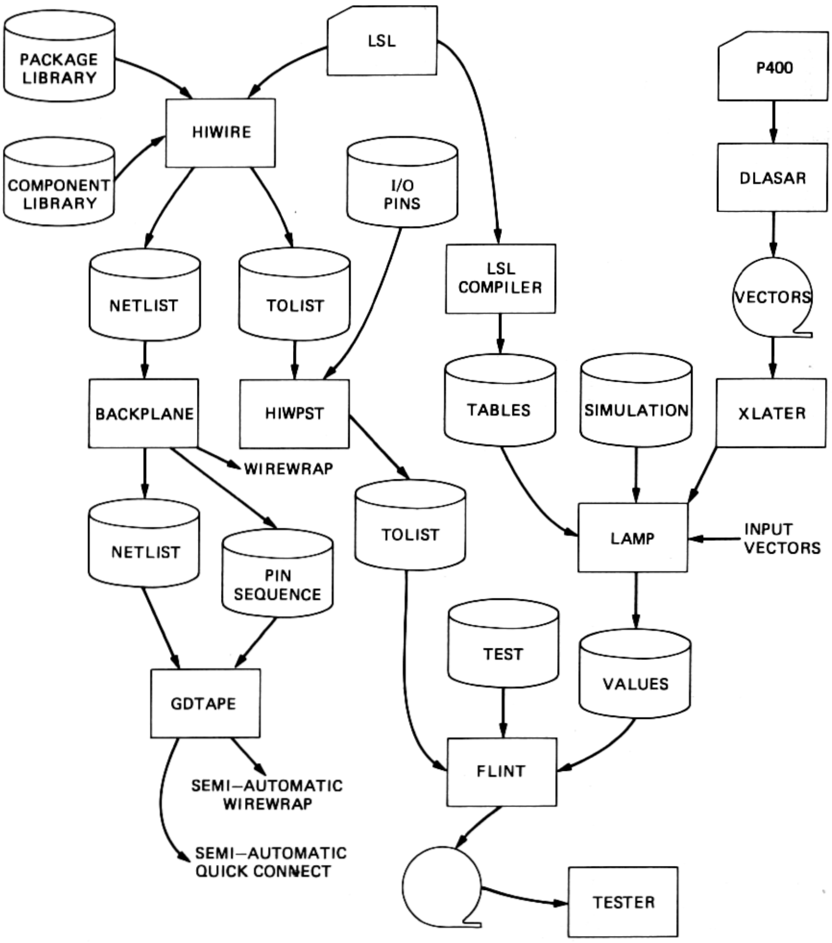


Fig. 2—Circuit pack design aids.

along with the number of programs and computers used is given in Table I. This process required 17 programs, which ran on seven different machines.

4.2 Custom electronics design aids

As another example of the complexity of the use of design aids, consider the test development process for the hierarchy of custom electronics (Fig. 3). Test development for custom silicon and hybrid integrated circuits is generally done using the LAMP simulator. The test sequences first exercise the chip's functions and then structurally exercise nonactivated gates. They are used to verify the performance of the silicon. Loading the device tester at our Allentown, Pennsylvania location first required transmission from LAMP (run under IBM TSS at Naperville, Illinois) to our CDC CYBER machine at North Andover, Massachusetts, then to the computer-aided silicon design system at Allentown, which generated a test tape for the device tester. These data transmissions were accomplished using a multipurpose batch station (MBS) over the interlocation computing network or over dial-up lines.

For circuit pack test development, one available tool is an outside supplier's automatic test generator. Translating the circuit description from LAMP to that test generator required the use of programs on both the CYBER and a HP 2100 computer. As a check for consistency, the vectors generated by the test generator were run against the initial coding of the circuit description on the LAMP simulator. The test development process required interaction with 13 different programs, residing on 11 computers, from 9 manufacturers, at 5 geographic locations. The process involved 11 hand carry links and 8 data transmissions.

Table I—Circuit pack design aids

Function	Number of Programs	Computers
Breadboard	5	IBM OS/370
Testability analysis	1	IBM TSS/370
Design verification, test generation, and diagnostics	3	IBM TSS/370 IBM OS/370
Schematic capture	3	DEC PDP-15 DEC PDP-11 IBM OS/370
Layout	1	DEC PDP-11
Metal extraction	1	DEC PDP-11
Connectivity audit	1	CDC CYBER 72 IBM OS/370
Test set	2	Data General Nova Computer Automation

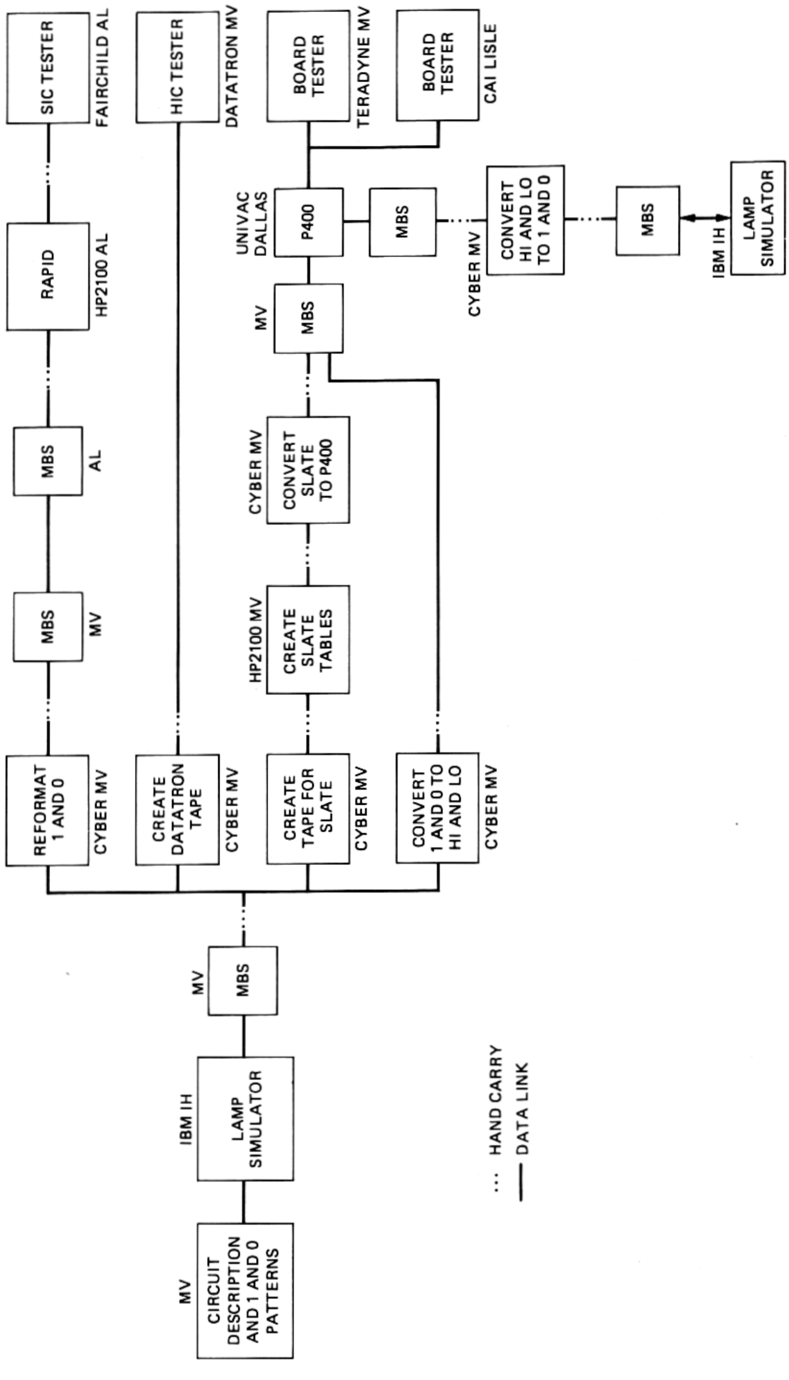


Fig. 3—Custom electronics test development process.

Table II—Custom electronics design aids

Function	Number of Programs	Computers
Design	4	CDC CYBER 72
Design verification, test generation, and diagnostics	7	IBM TSS/370 HP 2100 CDC CYBER 72 Univac 1108 IBM OS/370
Timing analysis	1	HP 2100 Harris
Layout		
Silicon	2	HP 21MX
HICs & PCBs	1	DEC PDP-11
Prove-in testing		
Column editor	1	CDC CYBER 72
Derace vectors	1	CDC CYBER 72
Test sets	3	Fairchild FST2 Data General Nova Teradyne M365
Translation		
Simulator to simulator	2	CDC CYBER 72
Simulator to test set	2	CDC CYBER 72

A list of the functions performed in the custom electronics design process is given in Table II. It includes the number of programs used and the host machines. For the entire process, 24 different programs are used.

V. DWB INTERACTIVE APPROACH

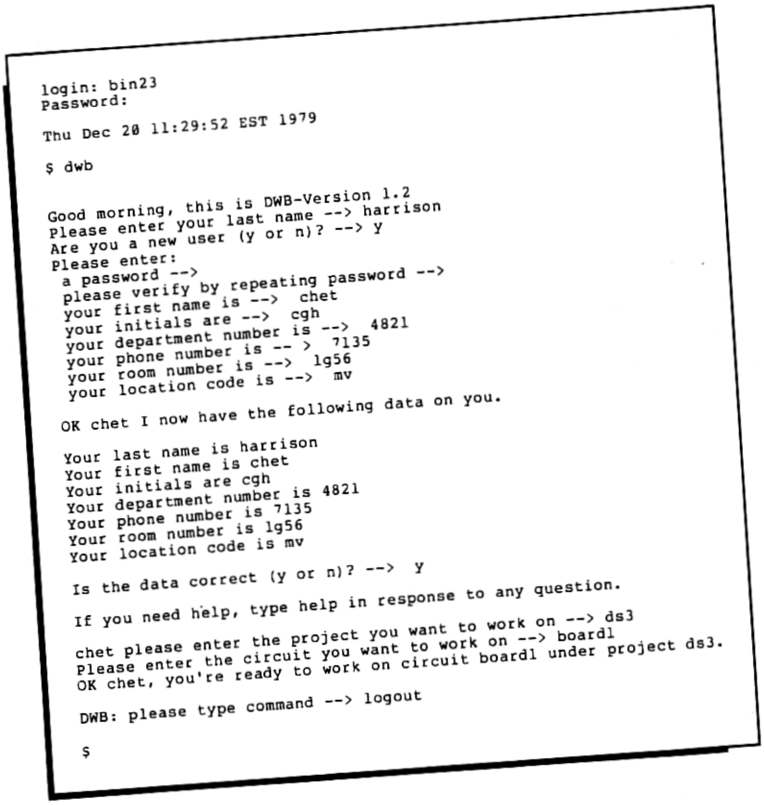
Like many design aids systems, the above systems did not gain ready acceptance by new or inexperienced computer users. A major reason is that the users generally faced a significant learning curve while trying to meet tight design schedules. Because of this learning curve, they relied on tried and true procedures rather than contending with numerous languages, passwords, phone numbers, file systems, libraries, cryptic unintelligible instructions, and error messages. These obstacles have been overcome in DWB by creating a "friendly," circuit-designer-oriented environment. This environment uses a minicomputer with the PWB/UNIX operating system[6,7] and provides the communications between the user and the application programs, which reside on many different main frame computers. This system effectively acts as a highly intelligent terminal to buffer the user from the intricacies and inconsistencies of the various main frame computers and application programs.

5.1 The environment

To control Designer's Workbench, a user must have access to a computer with the PWB/UNIX operating system that offers the package of DWB programs.

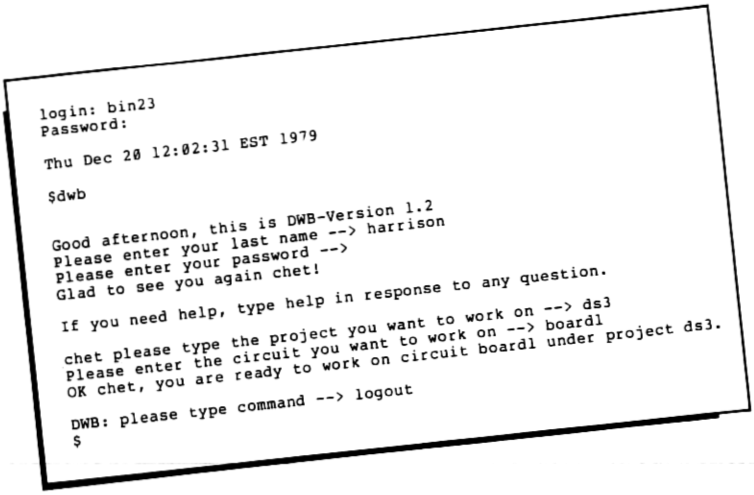
A new user must first have the staff of the computer department set up a valid user identification for the computer on which Designer's Workbench resides. To provide control and security, the DWB librarian must then add a user's name to the valid user file and appropriate project file. The user is now ready to start an interactive session on DWB.

When the user logs into the computer, he or she will see the PWB/UNIX operating system. The user next executes the Designer's Workbench program from PWB/UNIX and is prompted for his or her last name. If the user is new to Designer's Workbench, a user profile is set up by interactively prompting for required information. This profile is saved to be used in later sessions. The profile is built efficiently by requiring only the necessary information to do the task at hand (see Fig. 4). If additional information is needed for a specific application program, it will be prompted for at the time that program is accessed and added to the profile. Parameters that have logical and commonly



```
login: bin23
Password:

Thu Dec 20 11:29:52 EST 1979

$ dwb


Good morning, this is DWB-Version 1.2
Please enter your last name --> harrison
Are you a new user (y or n)? --> y
Please enter:
 a password -->
 please verify by repeating password -->
 your first name is -->  chet
 your initials are -->  cgh
 your department number is -->  4821
 your phone number is -- >  7135
 your room number is -->  1g56
 your location code is -->  mv

OK chet I now have the following data on you.

Your last name is harrison
Your first name is chet
Your initials are cgh
Your department number is 4821
Your phone number is 7135
Your room number is 1g56
Your location code is mv

Is the data correct (y or n)? -->  y

If you need help, type help in response to any question.

chet please enter the project you want to work on --> ds3
Please enter the circuit you want to work on --> board1
OK chet, you're ready to work on circuit board1 under project ds3.

DWB: please type command --> logout

$
```

Fig. 4—Initial session of new user.



```
login: bin23
Password:

Thu Dec 20 12:02:31 EST 1979

$dwb

Good afternoon, this is DWB-Version 1.2
Please enter your last name --> harrison
Please enter your password -->
Glad to see you again chet!

If you need help, type help in response to any question.

chet please type the project you want to work on --> ds3
Please enter the circuit you want to work on --> board1
OK chet, you are ready to work on circuit board1 under project ds3.

DWB: please type command --> logout
$
```

Fig. 5—Session of user known to DWB.

accepted default values are automatically set. However, the user may reset these or any other parameters at any time.

If, however, the user is recognized as one who has previously used DWB, the logon procedure continues with a friendly greeting and prompts for a password (if needed for security), project name, and finally circuit name (see Fig. 5).

From then on, all commands, edits, and operations will be automatically performed on files associated with this particular circuit. Each circuit has a profile that grows and changes according to the actions taken by the user and is used to store items such as library names, data types, and process keywords.[9]

The user has no knowledge of actual file names, file types, or naming conventions that other computer-aided design systems require. File maintenance is greatly simplified by using generic file names. For example, the file containing the electrical and physical circuit description is accessed by typing "lsl", since it is a variation of the LSL-LOCAL language. The file of input stimuli for logic simulation is accessed by typing "vectors." These and other keywords are used to describe the files that the system maintains for a particular circuit.

The interactive session focuses on one project and circuit. However, to change from one circuit to another, the "circuit" command is issued. Projects can be changed in a similar manner, but only if the user has authorization to access that project.

5.2 Tutorials

Users cannot be expected to operate a multioptioned software program without guidance and sources of reference. Printed manuals and user guides are available to study when there is time, but typically an unfamiliar technique is needed or a question arises while the user is on-line and in the middle of the job. The quick and easy-to-use series of on-line tutorials in Designer's Workbench eliminates the problem of hunting through manuals or, worse yet, stopping the session to go look for the answer somewhere else.

DWB is designed to let the user enter the tutorials at any time by typing the keyword help. A new user who enters the tutorials from the DWB command level has a menu of topics that explains the commands of interest. More details on special commands, utilities, application programs, or options are available if the user asks for information by entering the keywords that are provided. The tutorials are organized in a hierarchical manner, allowing the user to pursue as much or as little of a topic as needed. In addition to the hierarchical approach implemented to learn a topic for the first time, DWB has the feature that, if the user asks for help in the middle of a task, the section of the tutorial that pertains to the work at hand is immediately available. This saves the knowledgeable user from having to search for the information in an index mode. The user can end the tutorial session at any time and is ready to resume the job at hand, from the exact place in the program before the "help" command was requested (see Appendix A).

5.3 Data entry and modification

Data entry is always required to use computer-aided design programs. It is time-consuming and prone to human errors and misunderstandings. A goal of DWB is to make this step as easy as possible, yet flexible enough to serve the needs of the user community. Data entry takes two forms, directly appending information from a terminal or transferring data prepared somewhere else. Details on the transferring of files is discussed in its own section and is not discussed here.

Direct data entry through a terminal is provided using either the standard PWB/UNIX editor or the DWB "easy" editor. The "easy" editor has been designed to meet the needs of users who do not have computer editing experience. The PWB/UNIX editor is available to users who are experienced with computer systems and may profitably use the powerful and flexible features of the current generation of context editors.

For data entry and modification of test data, two additional features are available. One is a vector input language that allows a shorthand method of entering "one, zero" information, and the other is a columnar editor. These programs are used to build up the data files quickly and with a minimum of errors. The output files of these programs can be edited using the text editors.

Either editor can be used by the user to enter or modify any or all the user files. These files are accessed using the easy-to-remember generic names instead of actual file names. DWB again maintains the files and keeps track of names, types of data, and access dates for consistency checking.

Another form of data modification that does not use the editors should be mentioned. DWB provides default parameters at the circuit, project, or global levels. To modify parameters of the user's profile or of the circuit or project profile, a special "change" routine is provided. The user can at any time change the default conditions and the other parameters of the profiles.

5.3.1 Easy editor

The "easy" editor is designed to be used with the minimum amount of learning. It is similar to creating a punched card deck using a keypunch machine. The commands are immediately recognizable and the functions are basic. The user can append, change, delete, find, and list lines of the file to be edited. Alternatively, a, c, d, f, and l could be used. Operations are on a line-by-line basis, and the new file is saved only if the user says "yes" to the question concerning saving the new file. This "easy" editor is used to introduce DWB to new users who do not have a background in text editing.

5.3.2 Standard editor

The second editor is the standard PWB/UNIX editor, but it is accessed through the DWB program. This editor is extremely powerful, with string location, string modification, metacharacters, and global commands that can process complete files using a single complex statement. This power comes with a price, however, and that price is a longer learning period and less-than-clear command constructions. This editor is included in DWB to meet the needs of the growing user population who have learned this editor and feel at home using it.

5.3.3 Vector input language

The vector input language is not an editor but a powerful program to generate the stimuli for logic simulation or test set programs. It is described here since it is an important feature of the DWB interactive environment and does not exist in other computer-aided design systems.

The vector input language has features to allow the "one, zero" information used by logic simulators or test sets to be entered either in a timing relationship (functional form) or in a spatial relationship (test generation form). Control statements and macros allow the data set to be built quickly, understandably, and almost error-free.

5.3.4 Column editor

The column editor is another feature of DWB not found in many design-aids systems. Stimuli data for simulation programs and test sets have the characteristic that information for a test lead is stored in a column form rather than in a row form. While context editors easily handle lines or rows, most are clumsy if column manipulations are needed.

The column editor can add, change, and delete columns as easily as context editors edit lines. Moving columns and complimenting data in columns are commonly needed actions, but would be difficult to do without the column editor.

5.4 Job preparation

The environment that the user sees is friendly and easy to use when application programs are run. Although the application programs are run on computers located remotely from the user, the job is initiated, monitored, and retrieved under the control of DWB. For the application program selected to be run by the user, DWB provides the following preparation services:

(*i*) The circuit files are checked to see if they have been modified, thus invalidating output results that may have been obtained previously.

(*ii*) Missing data to run this option of this program are requested and added to the circuit profile.

(*iii*) The proper translators are invoked to build data files in the proper languages of the application program selected.

The following sections expand on these features.

5.5 File and job checking

Many user problems are caused by omitted information unknown to the user or by misunderstandings about the operations to be performed. Designer's Workbench has many checks and audits incorporated to protect the user from wasting time and resources running application programs with wrong data.

5.5.1 Syntax checking

Checking starts with a syntax check and verification of the input data supplied by the user. If information is missing, it is interactively requested and can be added by the user just before the application job is run. Note that only information needed to run the application of interest is required. Once this information is known, it is added to the profile of the circuit and used automatically in future runs.

5.5.2 Library checking

To insure that library elements referenced in a circuit description are available on a remote machine, a directory listing of remote libraries is kept on DWB. This listing is updated periodically. If DWB has the library element in its data base, this element is added to the job deck before submission to the remote machine.

5.5.3 Consistency checking

Checking is also done to inform the user about the proper order of doing things. DWB checks to be sure that the user runs application program A before application program B if it is required. Audits are made of the times files were last changed and application programs last run to detect any inconsistent conditions. If the user modifies the circuit description after an application run has been made, the output files will not agree with the current input files. The user is informed that the input and output are now out of phase with one another. This feature is particularly helpful to users who develop new circuit designs that change during the design cycle. Physical design, logic simulation, and test generation are kept in step with the "current" circuit description.

For example, to run the FLINT application program,[4] an automatic test set program generator that includes guided probe diagnostics, first the LAMP logic simulation program and then the HIWIRE physical connection program must be run (see Fig. 6). DWB keeps track of the steps executed, the order and if changes were made after some steps were completed. The checks to properly run FLINT are listed below:

(*i*) Check syntax of LSL input.
(*ii*) Run LAMP compile before LAMP run is allowed.
(*iii*) Check syntax of vectors before LAMP run is made.

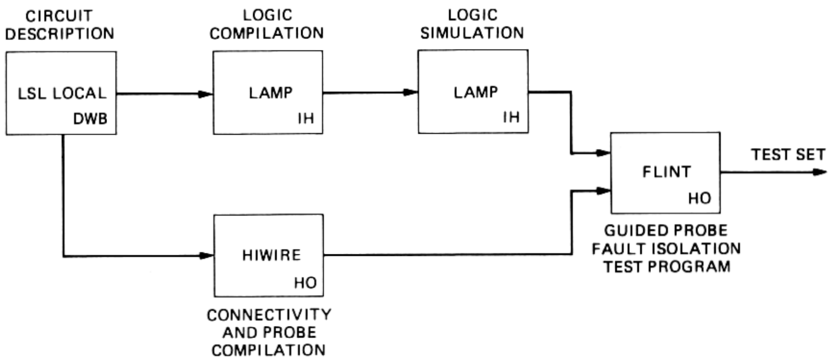


Fig. 6—Running FLINT application program.

(*iv*) Run HIWIRE before FLINT.
(*v*) Run LAMP before FLINT.
(*vi*) Run FLINT only if HIWIRE and LAMP use same LSL input.

5.5.4 Test vector checking

Before a job is submitted to a remote computer, the test vectors are checked for consistency with the circuit description and for legal vector values.

5.6 Data translation

An important ability of Designer's Workbench is to translate data files from one form into another. The input language used in DWB to describe the physical and logical interconnections of a digital circuit is based on LSL-LOCAL. The LAMP logic simulation program uses one dialect of LSL-LOCAL, while the HIWIRE model building program uses a different dialect of LSL-LOCAL. Designer's Workbench provides the user with an automatic method to translate from one dialect to another and even to completely different languages. For example, DWB can translate into the logic description language for LASAR[8] and for the Computer Automation (CAI) simulator.[10] Appendix B illustrates the conversion from the LSI-LOCAL language to the LASAR language. Other user files such as input vectors are also translated from one form into another. The "one, zero" form for LAMP becomes the "hi, low" for LASAR or a different "hi, low" for CAI.

Using the single master description stored on DWB, the user is able to translate into logic simulation languages, test generation languages, and physical model building languages, and into test program languages for automatic test equipment. The translators of DWB can easily interface to new applications as needed, yet provide the user with a single circuit description and test vector set to update and maintain.

To accomplish this, both the circuit description and test vectors are first translated into a neutral form. For the circuit description, this is a set of tables that contains component and connectivity information. Then the information is translated from the neutral tables into the target language.

VI. JOB SUBMISSION

A major feature of DWB is the semiautomatic or automatic submission of runs to remote computers and obtaining results from those computers. This section describes the elements of that process.

6.1 Communications network

Designer's Workbench uses the extensive communication capabilities of the PWB/UNIX operating system for this function. PWB/UNIX has a high-speed synchronous data link into the Bell Laboratories Interlocation Computing Network. These links operate at 4800, 9600, or 56000 baud, depending on the computers involved. Each computer system is assigned a unique identification and is a node on the network. The job control language directs the flow of information allowing a job to be sent to a remote site to be processed and have the results sent to a different location. The interlocation computing network provides the path to most application programs used by Designer's Workbench.

For those application programs that cannot be accessed through the Bell Laboratories Interlocation Computing Network, PWB/UNIX has alternate means used by DWB. PWB/UNIX has a call-originating feature that allows users to connect to the asynchronous time-share ports of remote computer systems. The dialing of the telephone number and the setup of the data lines is automatically provided as a PWB/UNIX feature. DWB uses this feature to communicate with systems not on the network.

6.2 Distributed database and libraries

The creation and maintenance of large data bases has always been a significant technological challenge. This, combined with the desire to minimize the amount of data transferred between the host computer on which DWB resides and the remote computer on which the application program resides, lead to the implementation of a distributed data base. Only enough data are transmitted to insure that all the required information can be reconstructed on the remote computer. The existence of separate data bases requires that DWB provide the consistency checking described above to insure that only consistent information is retained.

For example, instead of maintaining a complete library for all application programs on DWB, only cross-reference listings of libraries that exist remotely are maintained. If a user uses a subnetwork description of a function or part that exists on a library, this is quickly checked to see if it is in the current cross-reference library. Warnings are given to the user if DWB is not aware of the existence of the subnetwork.

6.3 Transferring files

The transfer of information from one person to another or from one physical location to another is often required, especially if the system and logic design is done at a different location from the fabrication or from the location of the computer-aided design programs that are used. The transfer process has been a major problem to users. Each location has different equipment, techniques, and formats that are unfamiliar to all but a few experts. Designer's Workbench can communicate with many different programs and different computers at locations remote to the user.

Most transfers are done automatically by DWB without the direct involvement of the user. For example, the running of application programs generates a data file transfer from the local system to the remote main frame computer. Completion of the job generates a data file transfer from the remote site back to DWB. This transfer is also done automatically.

A request for a file transfer from a remote computer to DWB can be done under user control and is often used to create the user files for a new circuit or to return results from application programs that have been run. The method of transfer will either be to submit a copy request using the proper job control language of the remote computer or to have DWB simulate an interactive time-share session. The user in either case must only supply the file name, location, and security requirements. Designer's Workbench will generate the proper job or protocol to successfully acquire the data. Transfer of a DWB file to a remote location is handled in a similar manner.

6.4 Submission options

Currently, applications programs can be run on remote systems in three ways. They are remote batch mode, interactive mode, and remote auditing mode. This section describes the implementation of those modes.

6.4.1 Remote batch

This mode involves combining the constituents necessary to run a batch run on a remote computer, submitting the card images over the computer network, and receiving the results via the computer network. Features include:

(*i*) A job is created to run on the remote computer inserting the customizing information from the user profile, the circuit profile, and the global profile.

(*ii*) The maze of accounting and job control statements is hidden from the user.

(*iii*) The data files are merged with the job control statements and program commands at the correct places.

(*iv*) The newly created job is automatically sent to the remote main frame, and the job is initiated.

(*v*) The outputs from the remote application program are sent via the network to the home location's computer printer.

6.4.2 Interactive TSS

Programs, such as LAMP, that run on the Time-Share System (TSS) can be accessed in two interactive modes. For both modes, DWB dials the remote machine and handles the logon (including encrypted passwords). In the direct mode, the user works on TSS as if the connection were made without the interjection of DWB, that is, as if the user had dialed directly. In the other mode, each command issued by the user is translated into the proper TSS command, and the responses from TSS are translated into user-understandable English. An example of this later approach is given in Fig. 7, which illustrates the check for the existence of a file on the TSS machine.

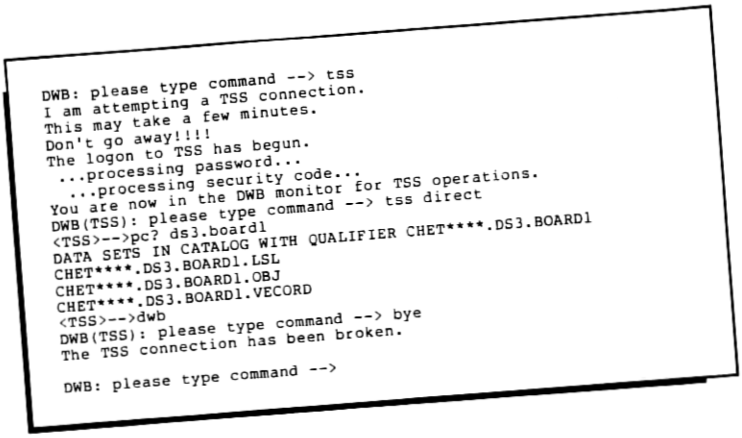
6.4.3 Remote auditing mode

LASAR is a commercial logic simulation package that runs on a time-share system outside Bell Laboratories. Remote batch access is not possible from DWB, yet an interactive session is not needed either. The remote audit mode of job submission is used to run the LASAR simulation program via DWB.

The remote audit mode of DWB uses the steps and features of remote batch submission except that the data file is sent in real time with the user observing the transfer of data and watching for the proper completion of the task. If the remote system is down or if transmission errors are observed (no error checking on this connection), the process can be restarted at a later time.

6.5 Job and network status

DWB provides the user with two status functions. The first enables the user to determine the status of any jobs that were submitted to remote machines by typing status lamp for LAMP or status abc for the ABC application program. The status of the last job submitted can be determined by just typing status. In addition, DWB informs the user of expected turnaround times for jobs submitted to remote machines.



```
DWB: please type command --> tss
I am attempting a TSS connection.
This may take a few minutes.
Don't go away!!!!
The logon to TSS has begun.
 ...processing password...
  ...processing security code...
You are now in the DWB monitor for TSS operations.
DWB(TSS): please type command --> tss direct
<TSS>-->pc? ds3.board1
DATA SETS IN CATALOG WITH QUALIFIER CHET****.DS3.BOARD1
CHET****.DS3.BOARD1.LSL
CHET****.DS3.BOARD1.OBJ
CHET****.DS3.BOARD1.VECORD
<TSS>-->dwb
DWB(TSS): please type command --> bye
The TSS connection has been broken.

DWB: please type command -->
```

Fig. 7—Interactive check for existence of a file on TSS.

This is accomplished by using a PWB/UNIX utility that periodically schedules jobs to be submitted to remote computers and measuring their turnaround times.

6.6 Postprocessing output data

Output data from application programs are sent to the user as printed results or as data sets of cards, magnetic tape, or disk files. All printout is automatically directed to the printout bin of the user, but data sets are sometimes saved on the remote computer until the user is ready to receive the data. DWB's ability to transfer data from one computer system to another is used to retrieve data for postprocessing. DWB is aware of the actual file names, the date modified, and contents of files generated by application programs for each circuit.

Data files from remotely run application programs often are not in the correct format to be used by another application program on a different remote computer. DWB postprocesses data from one form into an intermediate form that can be translated easily. Then, when this information is required, the proper output translator is called. Changes in the data structure of one application program can be accommodated without affecting the translators to the other applications. This translation takes place automatically, and the circuit profile is updated to show the status and options available to the user.

VII. USER FEEDBACK AND STATISTICS

The user response to the capabilities of DWB has been favorable. Inexperienced computer users who formerly would not have used design aids because of the significant learning curve have been converted to DWB. Experienced computer users, who had previously used applications programs on remote computers, have used DWB to create, check, and transfer files to the remote machine and retrieve output for postprocessing and transfer to other applications programs. Programs that were only used if there was a local consultant available are now seeing heavy use with no consultant available. The amount of time spent helping users solve JCL problems and use applications programs has been reduced by 80 percent. Significant effort has been required to match the tutorials and procedures to the thought processes of the user community. Most of the usage of the two design-aids systems that were integrated into DWB have shifted to the workbench.

The most significant requests for new capabilities have been to replace the predominantly batch-type operation with a fully interactive mode for executing programs on other computers and to provide completely automatic data transfer capability between all major computer systems in Bell Laboratories. The tutorial help has been found so popular that the user community has requested that the same type of help and guidance be provided for all interactive programs.

DWB was introduced to a limited number of "friendly" users in September 1978 and, after thorough testing, to the general-user community in January 1979. Since the beginning of 1979, we have seen steady growth of its use. The greatest portion of the use is in data preparation and syntax checking before submission of a job to a remote machine. In November 1979, usage totaled 519 logon hours and 111 application program submissions, with about 65 percent from the New Jersey location and 35 percent from the Massachusetts location. As more features and applications programs are added, we expect these figures to increase significantly.

VIII. NEW FEATURES

This section describes features that were not in the original specifications of DWB but have been already added or are in the process of being added. The modular development of DWB allows the requests of the user community to be addressed quickly and implemented without much of a change to sections already in use.

8.1 Testability analysis

DWB offers the user two capabilities for evaluating the "testability" of a circuit. The first is an interactive question-and-answer session between the user and DWB that qualitatively evaluates the testability of the circuit. Questions are asked about the design of the circuit to ascertain its inherent testability based on the strategy that previously had been applied manually by test engineers. Then DWB provides suggestions on how to improve the design of the circuit to make it easier to test. A report is automatically generated that summarizes the results of this testability review.

The second is the TMEAS[11] program that uses the LSL-LOCAL input circuit description language to evaluate the observability and controllability of the various parts of the circuit. The user is provided with a quantitative figure of merit on the testability of the design and a list of problem areas, if any exist. The program can be used interactively as an aid in resolving testing problems by providing insight into the value of additional test access.

8.2 Analog circuit design aids

DWB uses the LSL-LOCAL language to describe digital components, both electrically and physically (HIWIRE form). Analog components have been added by a new analog circuit description language. Tables are built for all components, and these tables are available for the various output translators to use. We are planning to establish links to both analog circuit analysis tools and physical design systems.

8.3 Interface to silicon design process

The silicon design process includes using LAMP for logic simulation and test generation. This DWB does as a regular feature. The timing analysis that is important for MOS LSI silicon chip designs uses the MOTIS[12] program. Input to the MOTIS simulator is another dialect of LSL-LOCAL and input stimuli in a native MOTIS form. DWB with its ability to transfer data and its translation programs is now integrating the MOTIS timing simulator into the remote application program selection. Further connections can be made in the future to lay out and test generation aids for silicon chips.

8.4 Downloading physical design process

The physical design process in the transmission area uses minicomputer based graphical data entry and editing stations from Applicon, Inc. Using these tools, the schematic diagram of the digital or analog circuit description can be captured and sent to DWB in the LSL-LOCAL (digital components) and UCL (analog components) circuit description languages. DWB will audit and compare the physical schematic capture with the data stored on Designer's Workbench, if it exists, or will use the data to build the tables used by other programs.

To help check the physical design process, DWB will initiate on the remote computer an audit of the connectivity of the physical printed wiring board, hybrid, or chip with the electrical connectivity used for model building, simulation, and test generation.

IX. CONCLUSIONS

The Designer's Workbench approach has greatly increased the user acceptance of design aids programs. Most user complaints about the aids have been removed by shielding the user from the idiosyncrasies of the computers, both the job control language and the input languages of the programs. In addition, most tedious, error-prone steps of data encoding, file management, consistency verification, and job customization have been removed or greatly simplified for the novice user without restricting the flexibility provided for the experienced user. Furthermore, it has created an environment in which new applications can be readily written and integrated with the existing aids.

The program modules and utilities that were written as part of DWB and the utility programs available from UNIX provide a solid foundation on which to build. Finally, Designer's Workbench is inexpensive to use. The operational costs are low because a minicomputer is used for the interactive operations while the large programs are run on main-frame computers. Thus, the costs are reduced by using each class of computer where it is most efficient.

APPENDIX A

Example of DWB Tutorial Session

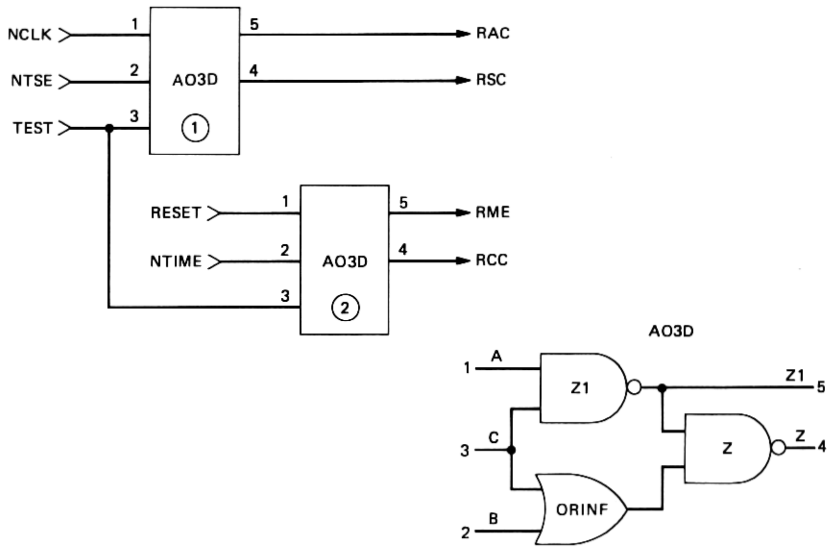
```
DWB: please type command → run lamp
Which option for ''lamp''? → help
lamp options are: build compile cval logic race timing

  DWB DOES NOT SUPPORT THE TIMING OPTION AT THIS TIME!

For additional information type ''more'' else type the option
Which option for ''lamp''? → more
There are four independent simulators available under lamp:
   1. cval   —a true value simulator which simulates circuit behavior
              without faults
   2. logic  —the standard logic simulator which drops star faults
   3. race   —a race simulator which flags and propagates star faults
   4. timing—a timing simulator which performs detailed timing
              analysis.

        DWB DOES NOT SUPPORT THE TIMING SIMULATOR
                     AT THIS TIME!

The true value (cval), logic, or race simulators may be accessed by
the options: cval, logic, or race, respectively.
Other options allowed are:
   1. compile —for compilation of your lsl local circuit description
   2. build    —to build a flint file on the HO IBM 370 for Datatron
               testing
For more detail type ''tutorial'' else type the option.
Which option for ''lamp''? → tutorial
***DWB TUTORIAL***
Lamp is a set of logic simulation programs used to generate and
evaluate test sequences. It is a Bell Laboratories product that is
interconnected to other aids systems. Lsl and vector files are required
as input. Other optional input files include vecord and lamp command
files. All output listings are returned to your computer bin.
Lamp simulates your circuit, using the test vectors you supplied DWB
and produces outputs according to procedures defined by the option
selected.
For even more detail consult the manual ''LAMP System-TSS Com-
mand Language (Version 4)'' TM-5936-10. If you have a specific
question on running lamp on DWB contact the DWB librarian at
```

```
extension 1537 at HO or 7893 at MV. For assistance in running lamp
itself on TSS contact Al Gibson on extension 9312 at IH.
You can also get help for:
        build    compile   cval     faults    logic     lsllocal
        race     starflt   timing   vecord    vectors
Type "end" or a keyword → compile
A compilation must be your first run. Your lsllocal circuit description
is sent from DWB to TSS at IH where it is stored under the user login
id. The lsllocal compiler is run and your circuit is checked for errors.
A good circuit is compiled into tables for use by the LAMP true value
and fault simulators (cval, logic, and race).
See also: cval logic race lsllocal
Type "end" or a keyword → end
***END DWB TUTORIAL***
Which option for "lamp"? → compile
```

APPENDIX B

Example of Translation from Augmented LSL to LASAR

1. Circuit to be coded:



2. Augmented LSL coding:

```
subnetwk:
   cktname: ao3d;
      inputs: a,b,c;
      outputs: z,zl;
**pin:in=(1,2,3),
out=(4,5);
      nodelay: orlnf;
      *unfault: orlnf(0,1); * unicad only;
      nand: zl,(a,c); z,(zl,orlnf);
      or: orlnf,(b,c);
*control circuit;
*options: propt = (nolist), cktsize = 8;
*option: autorout = .2;
network:
cktname: cl;
inputs: nclk,reset,ntse,ntme,test;
outputs: rsc, rac, rmc, rcc;
**pin: in = (1,2,3,4,5),
out = (6, 7, 8, 9);
ao3d:
a1, (nclk, ntse, test), (rsc, rac);
a2, (reset, ntme, test), (rcc, rmc);
**pd:
ao3d:
a1,, loc = (a-22),
a2,, loc = (a-23);
```

3. LASAR coding:

```
c1 CIRCUIT
INSERT ao3d
USE o2
NAME = o2
ADD ao3d
NAME = AA
MODEL/
110na/1, 3/5/120na/5, 6/4/130o2/2, 3/6/
INPUT/1, 2, 3/OUTPUT/4, 5/
END
END
CIRCUIT
PI nclk, 1P1
PI reset, 2P1
PI ntse, 1P2
PI ntme, 2P2
PI test, 1P3, 2P3
PO rsc, 1P4
PO rac, 1P5
PO rmc, 2P5
PO rcc, 2P4
DEVICE
1, 2 = ao3d
END
```

REFERENCES

1. L. A. O'Neill et al., "Designer's Workbench—Efficient and Economical Design Aids," Proc. 16th Design Automation Conference (June 25–27, 1979), pp. 185–199.
2. H. Y. Chang, G. W. Smith, Jr., and R. B. Walford, "LAMP: Logic Analyzer for Maintenance Planning, System Description," B.S.T.J., *53*, No. 8 (October 1974), pp. 1431–1450.
3. S. G. Chappell, C. H. Elmendorf, and L. D. Schmidt, "LAMP: Logic Analyzer for Maintenance Planning, Logic-Circuit Simulators," B.S.T.J., *53*, No. 8 (October 1974), pp. 1451–1476.
4. D. S. Evans and L. A. O'Neill, "An Integrated System for the Design of Printed Wiring Boards," ELECTRO '76 Professional Program, Session 26 (1976), Boston, Mass.
5. R. M. Allgair and D. S. Evans, "A Comprehensive Approach to a Connectivity Audit, or a Fruitful Comparison of Apples and Oranges," Proc. 14th Design Automation Conference (June 20–22, 1977), pp. 312–321.
6. K. Thompson and D. M. Ritchie, "The UNIX Time-Sharing System," B.S.T.J., *57*, No. 6, Part 2 (July–August 1978), pp. 1905–1930.
7. T. A. Dolotta, R. C. Haight, and J. R. Mashey, "The Programmer's Workbench," B.S.T.J., *57*, No. 6, Part 2 (July–August 1978), pp. 2177–2200.
8. *The Users Manual for the Teradyne P400 Automatic Test Plan Generation System*, Boston: Teradyne Inc., 183 Essex St.
9. P. H. McDonald and T. J. Thompson, "Designer's Workbench—The Programmer Environment," B.S.T.J., this issue, pp. 1791–1807.
10. *Capable Logic Simulation System User's Manual*, Irvine, California: Computer Automation Inc., 181 Dupont Drive.
11. J. Grason, "TMEAS, a Testability Measurement Program," Proc. 16th Design Automation Conference (June 25–27, 1979), pp. 156–161.
12. B. R. Chawla, H. K. Gummel, and P. Kozak, "MOTIS—NMOS Timing Simulator," IEEE Trans. on Circuits and Systems, *CAS-22*, No. 12 (December 1975), pp. 901–910.