NPS-EC-95-006

NAVAL POSTGRADUATE SCHOOL
Monterey, California

Analyzing VLSI Component Test Results of a GenRad GR125 Tester

by

D. Zulaica and C.-H. Lee

June 1995

Approved for public release; distribution is unlimited.

Prepared for: Office of Naval Intelligence
Washington, DC

Naval Postgraduate School
Monterey, California 93943-5000

Rear Admiral T. A. Mercer
Superintendent

R. Elster
Provost

This report was funded by the Office of Naval Intelligence.

Approved for public release; distribution unlimited.

This report was prepared by:

D. ZULAICA
Computer Specialist
Department of Electrical and
Computer Engineering

C.-H. LEE
Professor,
Department of Electrical and
Computer Engineering

Reviewed by:

MICHAEL A. MORGAN
Chairman,
Department of Electrical and
Computer Engineering

Released by:

PAUL J. MARTO
Dean of Research

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 30, 1995	Interim Report June 1, 1994–June 1, 1995

4. TITLE AND SUBTITLE
Analyzing VLSI Component Test Results of a GenRad GR 125 Tester

5. FUNDING NUMBERS

6. AUTHOR(S)
D. Zulaica and C.-H. Lee

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NPSEC-95-006

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Foreign Materials Branch, ONI-FMB (2333)
Office of Naval Intelligence
4251 Sutland Road
Washington, D.C. 20395-5720

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this report are those of the authors and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The GenRad GR125 VLSI chip tester provides tools for testing the functionality of entire chips. Test operation results, such as timing sensitivity or propagation delay, can be compared to published values of other manufacturers' chips. The tool options allow for many input vector situations to be tested, leaving the possibility that a certain test result has no meaning. Thus, the test operations are also analyzed for intent. Automating the analysis of test results can speed up the testing process and prepare results for processing by other tools. The procedure used GR125 test results of a 7404 Hex Inverter in a sample VHDL performance modeler on a Unix workstation. The VHDL code is simulated using the Mentor Graphics Corporation's Idea Station software, but should be portable to any VHDL simulator.

14. SUBJECT TERMS	15. NUMBER OF PAGES
VLSI tester, VHDL specification, automatic test equipment, GenRad GR125 tester	37
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

Analyzing VLSI Component Test Results Of a GenRad GR125 Tester

Table of Contents

I. Background

The GenRad GR125 VLSI chip tester provides tools for testing the functionality of entire chips. Test operation results, such as timing sensitivity or propagation delay, can be compared to published values of other manufacturers' chips. The tool options allow for many input vector situations to be tested, leaving the possibility that a certain test result has no meaning. Thus the

test operations are also analyzed for intent. Automating the analysis of test results can speed up the testing process and prepare results for processing by other tools.

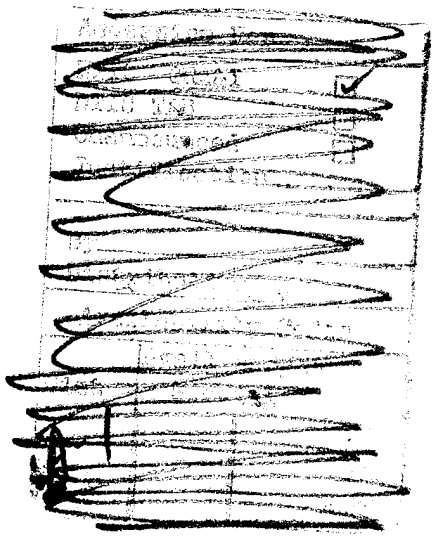
The test results from the GR125 can also be used in hardware modelling using a hardware description language such as VHDL. The automation process should also provide a means to allow VHDL to input the test results.

II. Objectives

1. Generate shmoo plots.
2. Create VHDL program to read the shmoo plot results.
3. Do Testing verification in VHDL.

III. Summary

Following is a procedure to use some GR125 test results of a 7404 Hex Inverter in a sample VHDL performance modeler on a Unix workstation, Figure 1. The VHDL code was simulated using the Mentor Graphics Corporation's Idea Station software, but should be portable to any VHDL simulator.



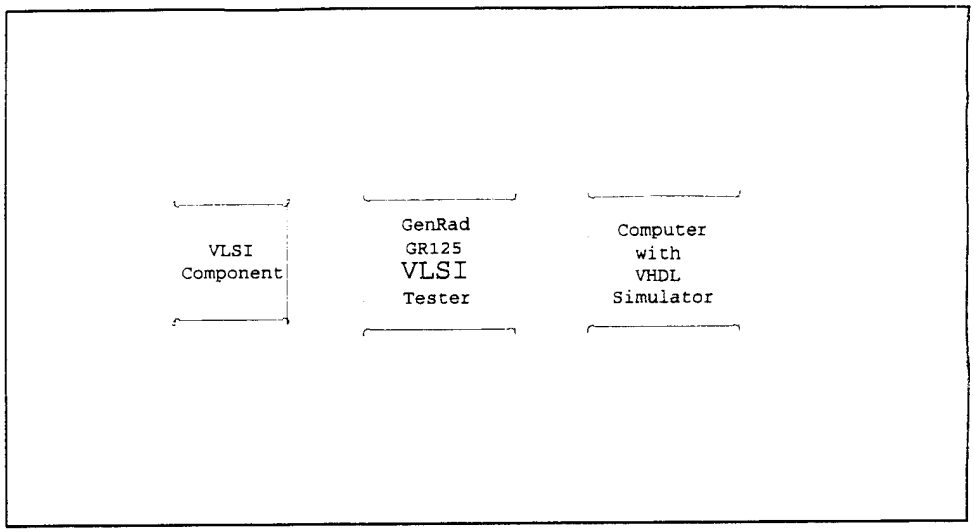


Figure 1. Extending GenRad GR125 capabilities.

IV. Shmoo Plots

The GR125 test results can be viewed as one or two dimensional characterization plots, or shmoo plots, [1] and [2]. Three variables are available for plotting, time, voltage, and current. Time provides means of analyzing switching characteristics such as propagation delay. Voltage and current will show electrical throughput limitations.

To get meaningful test results in the shmoo plot, an important parameter is in the Characterization Control-Setup screen. The pin level axis has a level option, which can have Voltage out low (Vol) or Voltage out high (Voh) among other values. The other

values such as pin level set 2 and pin condition set 2 should correspond with the appropriate pins being tested.

An example shmoo plot is shown in Figure 2. This example shows the minimum delay, of a low to high propagation delay test, occurring for a f74f04 hex inverter. The header portion of the plot explains that the minimum results are displayed, since the results from more than one test run can be stored. The GenRad 125 can run several different types of tests during one test run, for example, a low to high propagation delay test and a high to low propagation delay test can be run in sequence. The shmoo plot header states that the test operation number used is number 11, which was the low to high propagation delay test of the Test Operation setup screen.

Data points are Minimum Results Page 1

Test Operation Number: 11 Plot Name: f74f04 propagation delay

| 10.0_v -+ 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n
| 9.00_v -+ 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n
| 8.00_v -+ 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n
| 7.00_v -+ 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n
| 6.00_v -+ 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n
| 5.00_v -+ 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n
| 4.00_v -+ 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n
| 3.00_v -+ 13.00n 13.50n 13.00n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n 13.50n
| 2.00_v -+ 13.50n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.50n 13.50n 13.50n
| 1.00_v -+ 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n
| 0.00_v -+ 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n 13.00n
| |------- |-------- |------- |-------- |------- |-------- |------- |-------- |-------- |-------- |-------
| 0.00_n 2.00_n 4.00_n 6.00_n 8.00_n 10.0_n 12.0_n 14.0_n 16.0_n 18.0_n 20.0_n

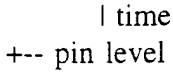


Figure 2. Low to high propagation delay shmoo plot for a f74f04 Hex Inverter.

A listing of test operation number 11, Figure 3, shows that this test is indeed a propagation timing test. The Timing Array Set number 2 is used to test the low to high propagation delay.

The data portion of the propagation delay shmoo plot in Figure 2 shows the minimum delay values obtained.

V. Storing Shmoo plots

Shmoo plots are displayed on a text display screen and stored in binary form in a GR125 output data logged file. When a test is finished a ^C (control-c) is typed to save the results to the data logged file and return to the screens menu control. The shmoo plots can be stored separately from the data logged files in ASCII format to allow transferring and printing of its contents. Following are the current procedures to store shmoo plots and to transfer all files from the GR125 to other machines.

```
TEST OPERATION 11
AC Functional  T-propagation  Comment:  tplh
```

```
Pass Range:     s < DUT <       s
     Time Value Set =     1     Search from/at = 12.000  ns
     Reference Edge =     1     Search to = 12.000  ns
     Search Edge    =     2     Resolution     = 200.000      ps
     Tracking Edge  =     8     At Search Time = 30.000  ns
     Tracking Edge  =           At Search Time =      s
     Tracking Edge  =           At Search Time =      s
     Tracking Edge  =           At Search Time =      s

Pin Test Set:  2     Outputs    Only Pin Test Set
Match on PT Set:                Compare All at Once

Pin Levels Set:      2     AC levels DO NOT SEARCH
Power Supply Set:    3     5.0 V     Linear Search
Timing Array Set:    2     tplh timing
Edge & Format Set:   1     NRZ Format      ETO Option = 0x0
Load Relay Set:      1     N/A

Module Name:         Timeout After =      3
Start uSeq at 21           Stop uSeq at          Enab Err V Set: 5
```

Figure 3. High to low propagation delay test operation setup for the f74f04 device.

A. Saving Characterization (Shmoo) Plots

The characterization plot displayed on the test screen can be saved for reference, later printing, or transferred to another machine for other possible uses. GenRad includes a program, pplot, [3], which translates a binary data logged file's data into an ASCII file. The ASCII file also includes control characters for printing to an Okidata 192 printer.

Pplot will create results based on the following information in the data logged file:

1. Minimum results obtained, or,

2. Maximum results obtained, or,

3. The last vector executed.

An option is also available to display results from a specific test operation number (-b#).

Pplot options are shown below:

-b# Plot a specific test operation. # can be values from 1 to 100.

-p# Plot one of three possible types:

1 The minimum points of all of the tests.

2 The maximum points of all of the tests.

3 The last vector executed

-d show debugging information

-h show help pages and legend.

V Verbose mode for debugging.

v> show version information.

Examples:

Plot the minimum results obtained from the f74f04 test log file. (There will be a prompt to specify the output filename.)

```
pplot -p1 f74f04.log
```

Plot the minimum results for test operation number 11.

```
pplot -p1 -b11 f74f04.log
```

Printing:

The new files can be printed on an Okidata printer with the following command,

```
lpr -w output-file
```

B. Transferring GenRad GR125 Files to Another Machine Via a PC

To transfer files from the GenRad GR125 to another machine perform the following.

1. Turn on the IBM PS/2 machine.

2. Invoke Zstem to log into the Genrad GR125,

   zstem /s

Zstem will now emulate a vt100 terminal through software.

3. Hit the Alt key to get the ZSTEM? prompt so that the communications protocols will be compatible between the GR125 and PS/2, ie. the baud rate.

4. Set the baud rate with the following command,

   baud r 19200

This sets the PS/2's baud rate at 19200 to match the remote, [r], GR125's baud rate.

5. Hit the enter key to return back to the vt100 terminal.

6. Hit enter again to get the login prompt.

7. Login to the GR125.

8. Note: To get characterization plots in ASCII format read the Saving Characterization Plots procedure.

9. To copy a file to a PS/2 disk drive run the kermit program,

   kermit

10. At the C-Kermit> prompt type in the download command,

    send <filename>

11. Hit the Alt key to get the Zstem prompt.

12. Type kermit to run kermit locally to receive the file.

13. At the options prompt type R to receive.

14. At the Local file [rmt]? prompt type in the new filename to reduce the filename to the eleven character DOS maximum.

15. When the download has finished, type F to end the local Kermit session.

16. Hit enter at the Zstem? prompt to return to the GR125 terminal and the C-Kermit> prompt.

17. When done transferring files type exit to exit the kermit program.

To transfer files from the PS/2 to the GenRad follow the above procedure with the following modifications:

1. Replace send in step 10 with receive.

2. Replace R in step 13 with S to send a file.

Symbol	parameter	Min	Nom	Max	Units
Vcc	Supply Voltage	4.75	5.0	5.25	V
Vih	High Level Input Voltage	2			V
Vil	Low Level Input Voltage			0.8	V
Ioh	High Level Output Current			-0.4	mA

Iol	Low Level Output Current			16	mA
Ta	Free Air Operating Temperature	0		70	C

Table 1: Recommended operating conditions for National Semiconductor's DM7404.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
Vi	Input Clamp Voltage	Vcc=Min Ii=-12mA			-1.5	V
Voh	High Level Output Voltage	Vcc=Min Ioh=Max Vil=Max	2.4	3.4		V
Vol	Low Level Output Voltage	Vcc=Min Iol=Max Vih=Min		0.2	0.4	V
Ii	Input Current @Max Input Voltage	Vcc=Max Vi=5.5V			1	mA
Iih	High Level Input Current	Vcc=Max Vi=2.4V			40	uA
Iil	Low Level Input Current	Vcc=Max Vi=0.4V			-1.6	mA
Ios	Short Circuit Output Current	Vcc=Max (Note 2)	-18		-55	mA

Icch	Supply Current With Outputs High	Vcc=Max		6	12	mA
Iccl	Supply Current With Outputs Low	Vcc=Max		18	33	mA

Note 1: All typicals are at Vcc=5V, Ta=25C.
Note 2: Not more than one output should be shorted at a time.

Table 2: Electrical characteristics for the DM7404, over recommended operating free air temperature (unless noted).

VI. Comparing Propagation Delay

A f74f04 Hex Inverter is used as one example to compare its low to high propagation delay with the published values of National Semiconductor's DM7404 Hex Inverter. The specifications for National Semiconductor's DM7404 device is shown in Tables 1-3, [4]. A two dimensional shmoo plot comparing high to low propagation delay with pin voltage level for a f74f04 is shown in Figure 2. Here the propagation delay at the 3 Volt level is observed in attempting to match the Typical (Typ) High Level Output Voltage of the DM7404, Table 2. Assuming the correct interpretation of the results, (see Figure 3 for the operation test values used), the plot shows

the minimum low to high propagation delay to be 13.0 nanoseconds (ns). The Typical value shown in Table 3 for the DM7404 is 12 ns.

These results imply that the DM7404 has quicker switching characteristics than the f74f04.

Parameter	Conditions	Cl=15pF Rl=400 Ohms Min	Cl=15pF Rl=400 Ohms Typ	Cl=15pF Rl=400 Ohms Max	Units
tplh Prop. Delay Time Low to High Level Output			12	22	ns
tphl Propagation Delay Time High to Low level Output			8	15	ns

Note 2: Not more than one output should be shorted at a time.

Table 3: Switching characteristics for the DM7404, at Vcc=5V and Ta=25C.

VII. Automation

The above observations can be automated and allow the results to be examined using a hardware modelling language, VHDL.

To read the ASCII shmoo plot from a Unix file into VHDL a package, stdlib, with comparable Unix functions was created, Figure 4. It currently has two functions; atoint converts the numeric digits from character to integer values; and atof converts a character string to a real number. Unix has a similar

atof function which is accessed through the stdlib.h header file in the C language.

Atoint is different from the Unix atoi function in that atoint only converts one integer character at a time into an integer. Two constant arrays were created, the string delimsint and the integer array delimsintval, to hold every integer digit, 0-9, as a character and integer. The string type is defined to index with all positive integers. To accommodate functionality with Unix, which was written using C, a new string type could be defined to include 0 or the NATURAL range defined in the VHDL std library.

Atoint is mainly a loop comparing the input character to all of the values in the delimsint array. If a match is found the integer type representation of that character integer is returned. The boolean variable ierror is used to stop the simulation if no integer digit is found. Changing ierror to false if a numeric character is found may not be needed, but was used for possible modification of the function and readability. Atoint can also be written to return a -1 if a non-numeric character is found, which is the standard error reporting procedure in the C language and Unix.

Atof takes an ASCII string with a maximum of 20 characters and converts it into a real value. This VHDL atof is similar to the Unix function strtod in that the string length is also required.

The Unix atof function calls strtod. Atof consists of five parts run in sequential order, as outlined below,

I. Get the integral part of the string (LOOP_I)

   A. Check if the character is a numeric digit

   B. Stop checking if the dot, '.', is found

II. If an integral part is found convert to real

   A. Get multiplication factor for first digit (LOOP_II)

   B. Multiply each digit by its 10's factor (LOOP_III),

      ie. for '432', (4 x 100) + (3 x 10) + (2 x 1) = 432 should result.

III. Increment the string astrng to pass over the '.' character

IV. Process the fraction part of the string if it exists (LOOP_IV)

   C. If a numeric digit is found multiply by its fractional 10's factor (LOOP_IV_A)

   B. Error check to reveal if the string is corrupted

V. Return the final real value

In the VHDL code loops are labelled in outline fashion for readability.

```
package stdlib is

--************************************************************

    constant MASL: integer := 20;                -- maximum ASCII string length
```

```
    constant NDI : integer := 10;                    -- delimiter string length
    type delim_int_vect is array (1 to 10) of integer;      -- an integer vector
    constant delimsint: string (1 to NDI) := "1234567890";
    constant delimsintval: delim_int_vect := (1, 2, 3, 4, 5, 6, 7, 8, 9, 0);

-------------------------------------------------------------------------------
-- Function atoint converts a single ASCII character digit to its integer value.

-- Input: asc is a character variable which should be from 0 to 9. Any other
-- ASCII character results in an error.

    FUNCTION atoint (constant asc : in character) return integer;

-------------------------------------------------------------------------------
-- Function atof is similar to the standard unix atof function, returning a
-- floating point value from an ASCII input string.

-- Input: astrng is an ASCII string representing a floating point number, ie.
--              "xxxx.xxxx "      The spaces are ' ' characters.
-- ASL is the astrng string length.

-- Not implemented:    1. Negative sign is not processed; an error will result.
--          2. A fraction must always start with a '0.' instead of just '.'.
--              3. Scientific notation is not supported.

    FUNCTION atof (constant astrng : in string (1 to MASL);
                   constant ASL : in integer)   RETURN real;

end stdlib;
```

```
-- ***************************************************************************
-- ***************************************************************************

PACKAGE BODY stdlib IS

-- ***************************************************************************

    function atoint (constant asc : in character) return integer is

    variable ierror: boolean := true;          -- Should change to false if integer found

    begin
        LOOP_I: for i in 1 to NDI loop
            if asc = delimsint(i) then              -- match the ASCII value and
                ierror := false;
                return delimsintval(i);             -- return the integer representation
            end if;
        end loop LOOP_I;

    assert ierror = false
        report "Error: The ASCII value was not an integer 0-9, (in stdlib.atoint)"
        severity error;

    end atoint;

-------------------------------------------------------------------------------

    function atof (constant astrng : in string(1 to MASL);
                   constant ASL : in integer)   return real is
```

```
    variable intgrl : integer := 0;          -- length of integer part of number
    variable dot : boolean := false;         -- TRUE if a '.' is found
    variable counter : integer := 0;         -- keep track of decimal point
    variable mult : real := 0.0;             -- 10's multiplier for each digit, including
fractional
    variable ansr : real := 0.0;             -- return the real number, ansr
    variable intg: integer;                  -- returned atoint() integer value
    variable tdigit: boolean;                -- flagged true if a fractional digit was found

    begin
        assert ASL <= MASL
            report "Error: ASCII string length greater than maximum allowed"
            severity error;

        LOOP_I: for n in 1 to ASL loop
            LOOP_I_A: for i in 1 to NDI loop
                if astrng(n) = delimsint(i) then   -- Is astrng(n) a numeric ,0-9, character?
                    intgrl := intgrl + 1;
                    exit LOOP_I_A;
                end if;
            end loop LOOP_I_A;

            if astrng(n) = '.' then              -- process the integral part
                dot := true;
                exit LOOP_I;
            end if;
        end loop LOOP_I;

        if intgrl > 0 then
            LOOP_II: for n in 1 to intgrl loop   -- multiplication factor for first digit
```

```
        if n = 1 then
            mult := 1.0;
        else
            mult := mult * 10.0;
        end if;
    end loop LOOP_II;

    LOOP_III: for n in 1 to intgrl loop
        intg := atoint(astrng(n));
        ansr := ansr + (real (intg) * mult);
        mult := mult / 10.0;
    end loop LOOP_III;
end if;

if dot = true then
    intgrl := intgrl + 1;        -- increment to process the fraction part of astrng to
end if; -- the right of the '.'

if intgrl < ASL then             -- no fraction part found
    LOOP_IV: for n in intgrl + 1 to ASL loop
        tdigit := false;         -- TRUE if astrng(n) is an integer digit
        LOOP_IV_A: for i in 1 to NDI loop
            if astrng(n) = delimsint(i) then
                tdigit := true;
                intg := atoint(astrng(n));
                ansr := ansr + (real (intg) * mult);
                mult := mult / 10.0;      -- keep track of the fractional placement
                exit LOOP_IV_A;
            end if;
        end loop LOOP_IV_A;
```

```
            if tdigit = false then
                if astrng(n) = ' ' then
                    exit LOOP_IV;
                else
                    assert astrng(n) = ' '
                        report "Error: invalid ASCII string; must not be a floating point
number (in stdlib.atof)"
                        severity error;
                end if;
            end if;
        end loop LOOP_IV;
    end if;
    return ansr;
  end atof;
END stdlib;
```

Figure 4. Stdlib package duplicating standard Unix functions documented in Stdlib.h.

The VHDL entity getdelay reads the ASCII shmoo plot file by parsing it and extracting the pertinent information for comparison. For this example the minimum propagation delay value is extracted at the 3 volt level. A generic, vlevel, is used to specify the voltage level. A generic, cdelay, is also used to specify the typical delay for the DM7404. These values could also be written to a simple input file which than can be read by getdelay. The result is the difference between the DM7404 and the

f74f04 written to an output file, result. Here the answer is -1.0 showing that the f74f04 is slower than the DM7404 reference value.

An outline of the getdelay process follows,

I. Read and process each line in the header section (LOOP_I)

A. Parse the current line read (LOOP_IA)

1. Find a delimiter for word separation (LOOP_IA1)

2. Check for end of file (a manually inserted character here)

3. If a non-delimiter is found get the word (LOOP_IA2)

II. Leave a NOTE that the data section will be processed, (for debugging purposes)

III. Read and process each line in the data section (LOOP_II)

A. Parse the current line read (LOOP_IIA)

1. Find an integer digit (LOOP_IIA1)

2. If an integer digit is found get the real number (LOOP_IIA2)

a. Check for end of file

b. Find an integer digit (LOOP_IIA2a)

c. Real number read into string 'strng'

1. Add spaces to the rest of 'strng'

2. check for unit of measure

3. Convert 'strng' to a 'real' type, (call atof)

4. If the line is at the correct voltage level get the minimum delay value

IV. Get the propagation delay differences and write the answer to a file

LOOP_I processes the header information of the shmoo plot. The Plot Name, see Figure 2, can contain only the device under test name to be extracted and used in the output file. To check for an end of file a '@' was entered at the end of the shmoo plot file. Since the shmoo plot file was created automatically by a software program its format should not change and here an end of file should not be expected. '@' was used for debugging purposes.

The shmoo plot format allowed for an easy check to find the data section. In Figure 2 the first '|' is seen at the beginning of the top line of data. This also meant that a different parsing algorithm could be used to find the data values.

The VHDL TEXTIO package has a function for reading real values. However, the shmoo plot file contains non-numeric characters in the data lines. This could be fixed by first filtering each line, but the unit of measure (UOM) for each value is also used. Rather than keep track of the UOM for each data value LOOP_II finds the value and checks for its UOM right away. The parsing was accomplished by using the ten numeric digits and '.' as delimiters rather than loop through all possible characters,

including control characters.

```
--          getdelay.vhd
--

entity getdelay is
    GENERIC (constant vlevel : in real := 3.0;      -- the voltage level to compare
             constant rvalue : in real := 12.0;     -- reference delay value
             constant ND : in integer := 12;  -- number of delimiter characters
             constant NDI : in integer := 11; -- number of delimiter characters during
                -- integer read
             constant WS : in integer := 128;       -- length of word
             constant WIS : in integer := 10; -- length of word integer
             constant SL : in integer := 256);      -- length of READ string, strng
end getdelay;

library unix;
use unix.stdlib.all;
use std.textio.all;

architecture proc of getdelay is

file infile : text is in "cpfile.plt";
file outfile : text is out "result";
signal zout: character := '0';              -- test signal
-- signal delimflag: boolean := false;      -- testing TRUE if a delimiter was found
signal words: string (1 to WS);             -- test word as a signal
```

```
signal wordsint: string (1 to WIS);      -- wordint as a signal, maybe out port
signal fltval: real;                        -- wordint translated to real, check signal
signal volts: real := 0.0;                -- test signal, shows voltage used
signal mindels: real := 10000.0;          -- test signal, should show min delay value for
voltage

begin
    proc_I : process
    variable line_ptr : line;
    variable resline : line;
    variable strng: string (1 TO SL);     -- characterization plots have up to 132 columns
                                                -- plus control characters
    variable j: integer;                     -- index of strng
    variable noeol: boolean;                -- end of line flag
    variable noeof: boolean := true;        -- end of file flag, should only change once
    variable delimflag: boolean;            -- TRUE if a delimiter was found
    variable delimflagint: boolean;         -- FALSE if an integer delimiter was found
    variable word: string (1 to WS);           -- The current word to process
    variable wordint: string (1 to WIS); -- The current integer to process as int
    variable wi: integer := 0;                -- index of word, initially at 1
    variable wl: integer;                   -- length of word
    variable bmtrue: boolean := false;         -- true if "|-" found
    variable mptrue: boolean := false;         -- true if "-+" found
    constant delims: string (1 to ND) := " ;=|-^[" & lf & cr & nul & ht & esc;
    constant delimsint: string (1 to NDI) := "0123456789" & "."; -- couldn't compile with
only
        -- one string
    variable volt: boolean;                    -- TRUE if voltage axis is found
    variable delay: boolean;                -- TRUE if a delay value is found
    variable word2flt: real;                -- wordint translate to floating point
```

```
    variable getdel: boolean := false;      -- get the delay value if TRUE, for 5V values only
    variable mindel: real := 10000.0;       -- initial minimum delay in nanoseconds
    variable answer: real;      -- answer to the difference between reference value and read
delay

    begin
        -- Read and process each line in the header section of the plot file
        LOOP_I: WHILE noeof loop                -- noeof will be FALSE if time axis has been
read
            READLINE (infile, line_ptr);    -- no provisions for EOF?
            wait for 1 ns;
            noeol := true;
            j := 1;

            -- LOOP_IA - process file header information
            -- Process each character in the current line
            LOOP_IA: WHILE noeol loop -- noeol is FALSE if end of line reached
                wi := 0;
                j := j + 1;
                READ (line_ptr, strng(j), noeol);
                zout <= strng(j);
                wait for 1 ns;
                delimflag := false;

                LOOP_IA1: FOR i IN 1 to ND loop     -- find a delimiter
                    wait for 1 ns;
                    if strng(j) = delims(i) then

                        if strng(j) = 'l' then
                            exit LOOP_I;
```

```
        end if;

        delimflag := true;
        exit LOOP_IA1;
    end if;
end loop LOOP_IA1;

if zout = '@' then              -- end of file indicator
    noeof := false;
end if;

LOOP_IA2: WHILE delimflag = false loop
    wi := wi + 1;                   -- write to next character of word
    word(wi) := strng(j);           -- character added to word
    j := j + 1;                     -- read the next character from the line
    READ (line_ptr, strng(j), noeol);

    if zout = '@' then              -- end of file indicator
        noeof := false;
    end if;

    zout <= strng(j);
    wait for 1 ns;

    LOOP_IA2a: FOR i IN 1 to ND loop   -- find a delimiter
        wait for 1 ns;
        if strng(j) = delims(i) then
            delimflag := true;
            wl := wi;  -- last increment of word index is word length
```

```
                    LOOP_IA2a1: for k in wi+1 to WS loop
                        word(k) := ' ';
                    end loop LOOP_IA2a1;

                    words <= word;
                    exit LOOP_IA2a;
                end if;
            end loop LOOP_IA2a;
        end loop LOOP_IA2;
    end loop LOOP_IA;
end loop LOOP_I;

assert NOT (strng(j) = 'l')
    report "At the beginning of the delay data"
    severity note;

-------------------------------------------------------------------------

    -- Read and process each line in the data section of the plot file
    LOOP_II: WHILE noeof loop          -- noeof will be FALSE if time axis has been
read
        READLINE (infile, line_ptr);    -- no provisions for EOF?
        wait for 1 ns;
        noeol := true;
        j := 1;

        -- LOOP_IIA - process the delay information
        -- Process each character in the current line
        LOOP_IIA: WHILE noeol loop -- noeol is FALSE if end of line reached
            wi := 0;
```

```
j := j + 1;
READ (line_ptr, strng(j), noeol);
zout <= strng(j);
wait for 1 ns;

delimflagint := false;
LOOP_IIA1: FOR i IN 1 to NDI loop  -- find an integer delimiter
    wait for 1 ns;
    if strng(j) = delimsint(i) then
        delimflagint := true;
        exit LOOP_IIA1;
    else
        if strng(j) = '@' then              -- end of file indicator
            noeof := false;
        end if;
    end if;
end loop LOOP_IIA1;

LOOP_IIA2: WHILE delimflagint = true loop  -- get the integer from the file
    wi := wi + 1;                          -- write to next character of wordint
    wordint(wi) := strng(j);               -- character added to wordint
    j := j + 1;                            -- read the next character from the line
    READ (line_ptr, strng(j), noeol);

    if zout = '@' then                          -- end of file indicator
        noeof := false;
    end if;

    zout <= strng(j);
    wait for 1 ns;
```

```
delimflagint := false;
LOOP_IIA2a: FOR i IN 1 to NDI loop -- find an integer delimiter
    wait for 1 ns;
    if strng(j) = delimsint(i) then
        delimflagint := true;
    end if;
end loop LOOP_IIA2a;

if delimflagint = false then   -- finished getting a number
    wl := wi;                  -- last increment of wordint index is word length

    if wi /= WIS then          -- should never be greater than WIS
        LOOP_IIA2b: for k in wi+1 to WIS loop
            wordint(k) := ' ';
        end loop LOOP_IIA2b;
    end if;

    volt := false;
    delay := false;
    if strng(j) = '_' then     -- find out if number is axis or delay value
        j := j + 1;            -- the input file format is already established
        READ (line_ptr, strng(j), noeol);
        if strng(j) = 'v' then
            volt := true;
        end if;
    elsif strng(j) = 'n' then
        delay := true;
    end if;

    word2flt := atof(wordint, WIS);
```

```
                fltval <= word2flt;

                if volt = true then
                    if word2flt = vlevel then
                        getdel := true;
                        volts <= word2flt;          -- test signal, should show vlevel
                    else
                        getdel := false;
                    end if;
                end if;

                if getdel = true then               -- find the min value
                    if delay = true then
                        if word2flt < mindel then
                            mindel := word2flt;
                            mindels <= word2flt;
                        end if;
                    end if;
                end if;

            end if;
        end loop LOOP_IIA2;
    end loop LOOP_IIA;
end loop LOOP_II;

answer := rvalue - mindel;
write (resline, answer);
writeline (outfile, resline);

wait;
```

```
    end process proc_I;
end proc;
```

Figure 5. VHDL source code to read a shmoo plot and extract the minimum voltage.

VIII. Conclusions

The example of checking the propagation delay of a device to that of a reference device through a VHDL program is a first step in the automation process. New programs can be written to test different characteristics of a device. This provides the opportunity to standardize programming techniques as in-house templates similar to standard programming language syntax. Some code could be easily recognizable to new and fellow programmers. Programming productivity should also increase.

An example is provided in the proc architecture of getdelay, (Appendix C). There are two main parsing loops, LOOP_I reads the header information and LOOP_II gets the voltage level and delay values. A template may look like the following,

```
        -- Read and process each line in a file
        LOOP_I: WHILE noeof loop          -- noeof will be FALSE if time axis has been
read
            READLINE (infile, line_ptr);
            wait for 1 ns;
```

```
        noeol := true;
        j := 1;
        -- LOOP_IA
        -- Process each character in the current line
        LOOP_IA: WHILE noeol loop --  noeol is FALSE if end of line reached
            wi := 0;
            j := j + 1;
            READ (line_ptr, strng(j), noeol);
            . . .
            . . .
            . . .
        end loop LOOP_IA;
    end loop LOOP_I;
```

Another example is a simple template which has all of the steps to read data from a file,

```
    file <file-pointer-name> : text is in "<filename>";
    variable <line-pointer-name> : line;

    READLINE (<file-pointer-name>, <line-pointer-name>);
    READ (<line-pointer-name>, <variable-name>, <boolean-type>);
```

Ultimately a database can be developed which can be accessed from VHDL as well as C/C++, Figure 6. In the internet a parts database protocol may be developed to allow potential buyers to review parts specifications at either no cost or preview cost.

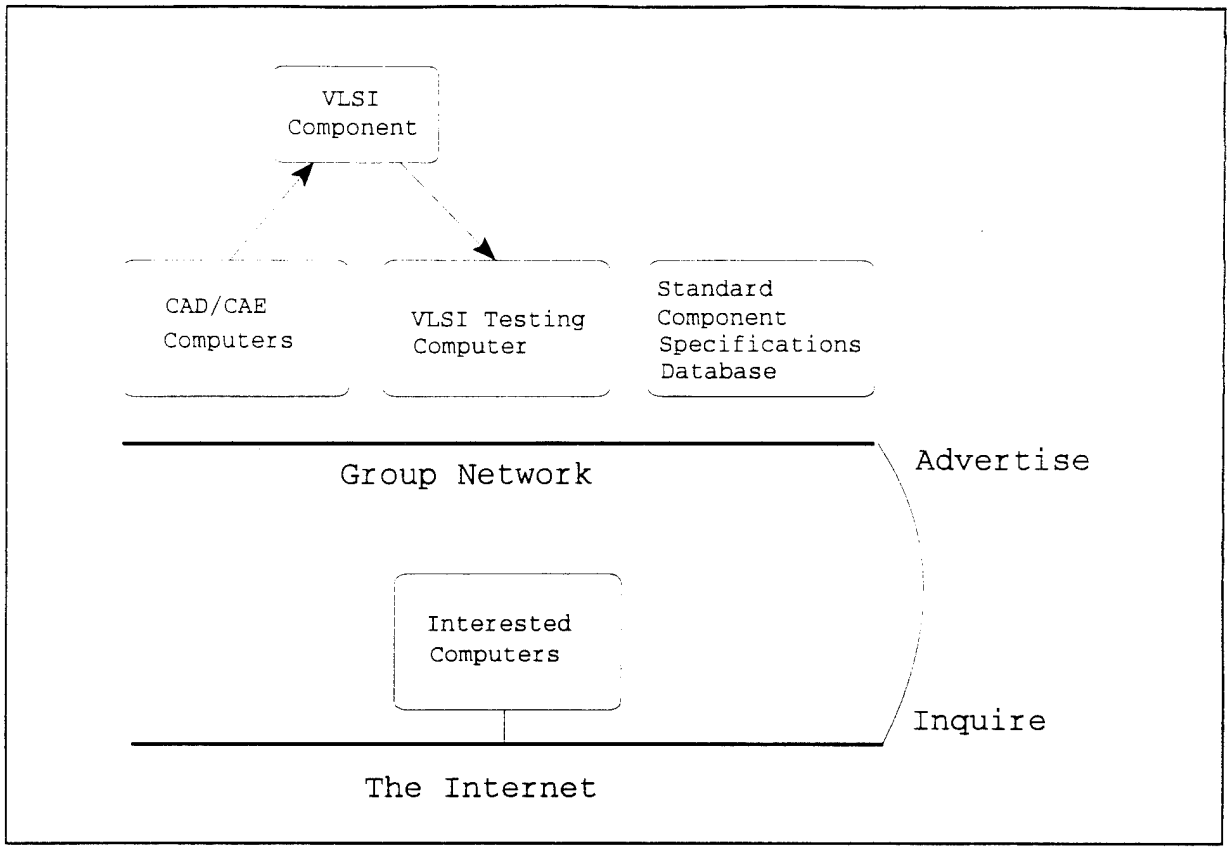


Figure 6. Internet-wide database.

IX. References

1. GenRad Component Test Systems, "GR125 User's Guide," GenRad, Inc., Concord, MA, 1990.
2. Loeblein, James T., "A Digital Hardware Test System Analysis With Test Vector Translation," Naval Postgraduate School, Monterey, CA, 1992.
3. GenRad, "GR115/125/130 Version 2.1 Release Notes," GenRad, Inc., Concord, MA, 1989.

4. National Semiconductor Corporation, "Logic Databook, Volume II," National Semiconductor Corporation, Santa Clara, CA, 1984.

5. Mentor Graphics Corp., "Design Architect User's Manual," Mentor Graphics Corp., Wilsonville, OR, 1994.

6. Mentor Graphics Corp., "Mentor Graphics VHDL Reference Manual," Mentor Graphics Corp., Wilsonville, OR, 1994.

7. Mentor Graphics Corp., "Quicksim II User's Manual," Mentor Graphics Corp., Wilsonville, OR, 1994.

8. Plauger, P.J., "The Standard C Library," Prentice Hall, Inc., Englewood Cliffs, NJ, 1992.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Dudley Knox Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5101	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
4.	Professor C.-H. Lee, Code EC/Le Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	4
5.	Dan Zulaica, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1