A New User-Friendly ATPG Platform for Digital Circuits

M. Lipovský[1)], J. Švarc[1)], E. Gramatová[1)] and P. Fišer[2)]

[1)] Faculty of Informatics and Information Technologies STU in Bratislava, Slovakia

[2)] Faculty of Information Technology ČVUT in Prague, Czech Republic

***Abstract*—**The paper presents a new graphical platform for automatic test patterns generation and fault simulation for digital circuits. The platform integrates two existing academic tools for test pattern generation and fault simulation: ATALANTA and HOPE. Both tools use a specific format "bench" for circuit description which is not suitable in connection to professional CAD tools. Therefore, the platform has been extended by a new translator for mapping a VHDL digital circuit model to the format "bench". The platform contains also a separate random test patterns generator linked to the fault simulator HOPE and generation of test pairs for delay faults using the transition fault model. The new automatic test pattern generation platform provides a user-friendly environment suitable for education.

Index Terms — digital circuits, ATPG, fault simulation, VHDL, HOPE, ATALANTA.

I. Introduction

The paper is aimed at automatic test patterns generation (ATPG) for digital circuits, both combinational and sequential logic circuits. Current technologies allow various applications of digital and mixed circuits integrated on a chip, and therefore education in design and test topics is still important and useful. Design houses have always needed experts for testing and design for testability techniques application. Therefore, students have to be educated for the mentioned topics and have to receive not only knowledge, but also good skills in using professional and academic software tools in this field. Universities cannot always use professional ATPG tools in education due to high cost of licenses. Thus they use free academic ATPG software tools, e. g. ATALANTA [1], and a fault simulator HOPE [2], [3]. The tools provide a wide range of options, and thus they are very suitable for education. Another advantage is their variability related to test sets, fault dictionary and fault coverage. They are relatively old but currently no other academic tools are available for using in education. Both tools use simple interfaces and a transparent input format, named "bench". The format is used in benchmark digital circuits – ISCAS'85 [4] and ISCAS'89 [5].

It would be useful to have a more efficient, graphical and user-friendly interfaces using up-to-date programming environments and a VHDL circuit description. Therefore a new ATPG platform, named GATE (**G**raphical ATPG platform based on **AT**ALANTA and HOP**E**), has been developed and implemented. A new translator from VHDL digital circuit description to the format "bench" (VHDL2BENCH) has also been integrated there.

GATE is implemented in C# and provides nice graphical windows for both tools and the translator. The ATPG is extended by test pattern pair generation using the transition fault model. The platform has been developed and implemented by two students educated in courses oriented to digital circuits testing and supervised by their teachers. Functionality and effectiveness of the GATE platform have been evaluated using several combinational and sequential benchmark circuits [4], [5] and also using simple in-house circuits modeled in HDL Designer.

The paper is organized as follows. The next section shortly describes the functionality of both academic tools ATALANTA and HOPE. Section III presents the platform proposal. Implementation of the developed platform and experimental results are shown in section IV and V respectively. Section VI concludes the paper.

II. Academic ATPG tools ATALANTA and HOPE

The popular academic tools ATALANTA and HOPE were developed at the Virginia Tech University. They are available as executable files with required libraries and manual pages [6], [7]. They run on Microsoft Windows XP and newer versions of operating system Windows, They use a digital circuit described in format "bench" as input. Both tools use the same circuit description and few similar options for running. These options are shortly described in Table I. The interface is very simple and both tools run as console applications. Thanks to their variability both tools have been selected for integration into the GATE platform.

Table I: Common options and functionalities

Option	**Description**
-f fn	Faults are read from file fn. It is available only for ISCAS'89 netlist format.
-h a	Displays the entire manual.
-l fn	Creates a log file, named fn.
-s n	Random number generator's initial seed is set to number n. If n = 0, the initial seed is set using the day time of computer.
-N	Test compaction is not performed.
-r	Test patterns are generated randomly.
-u	All aborted or undetected faults are printed out in a specified file.
-t fn	Test patterns are saved into file fn (ATALANTA); test patterns are defined in the file (HOPE).

Example: The structural model of the c17 circuit (ISCAS'85) shown in Figure 1 is described in the "bench" format presented in Figure 2. The character # at the beginning of the line specifies a comment. After the comments there is a declaration of inputs and outputs and then each line specifies the function of one logical gate in the circuit.

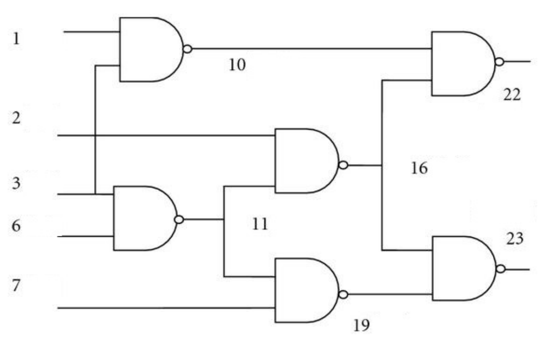
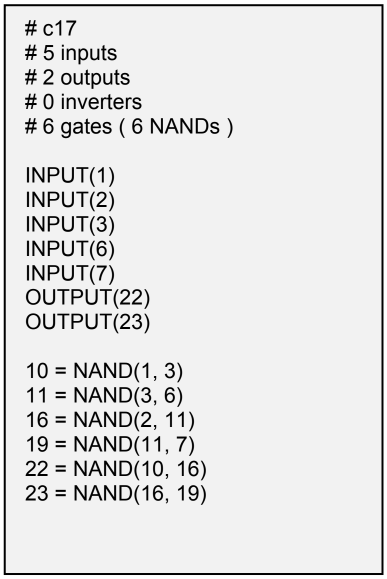


Figure 1: Circuit c17.



```
# c17
# 5 inputs
# 2 outputs
# 0 inverters
# 6 gates ( 6 NANDs )

INPUT(1)
INPUT(2)
INPUT(3)
INPUT(6)
INPUT(7)
OUTPUT(22)
OUTPUT(23)

10 = NAND(1, 3)
11 = NAND(3, 6)
16 = NAND(2, 11)
19 = NAND(11, 7)
22 = NAND(10, 16)
23 = NAND(16, 19)
```

Figure 2: Bench format of circuit c17.

A. ATPG tool ATALANTA

ATALANTA can handle combinational circuits only and it is based on the FAN algorithm [8]. It is used for detection and localization of stuck-at-0 and stuck-at-1 faults. The basic syntax for running ATALANTA is:

"atalanta [options] circuit.bench [> output_file]".

One output file, named "<circuit_name>.test", is generated and it contains test vectors and fault-free output values. For example, a test set for the c432 circuit (ISCAS'85) is saved to the file c432.test. Special options for ATALANTA are described in Table II. ATALANTA uses also a parallel fault simulator FSIM [2] for test patterns compaction without considering test don't cares.

B. Fault simulator HOPE

The fault simulator HOPE can be used both for combinational and sequential logic circuits. The basic syntax for running HOPE is:

"hope [options] circuit.bench [> output_file]".

HOPE uses also several specific options described in Table II.

Table II: ATALANTA and HOPE specific options

Option	Description
ATALANTA	
-A	Diagnostic mode. ATALANTA prints out all test patterns for each fault. All unspecified inputs are left unknown and fault simulation is not performed.
-D n	Diagnostic mode. ATALANTA prints out n test patterns for each fault. All unspecified inputs are left unknown and fault simulation is not performed.
-b n	Specification of the number of maximum backtracks.
-B n	If n>0, ATALANTA generates test in 2 phases. Static unique path sensitization is used in phase 1. If test generation is aborted thus phase 2 is being realized. Dynamic path sensitization is used in phase 2 and n specifies number of backtracks in this phase.
-H	ATALANTA uses fault simulator HOPE.
-c n	Test patterns are compacted using two different methods: reverse order compaction and shuffling compaction. Number n specifies the limit of shuffling compaction.
-s n	Number n is the initial seed for random generator.
-v	All identified redundant and aborted faults are printed out in a file named <ckt>.ufaults.
-Z	One test pattern is generated for each fault.
-0, -1, -x, -R	Value of unspecified inputs is set during test generation to logic 0 or logic 1, unknown, randomly to logic 0 and 1.
HOPE	
-D	The list of newly detected faults is created and written to log file. Option -l should be used to define the log file.
-F fn	Unfaulty and faulty outputs are written for each fault to file fn. No fault dropping method is applied in this option. That means all faults are injected and simulated in parallel.
-x	Potentially detected faults are dropped as soon as they are detected
-0, -1	All flip-flops are set to logic 0 or logic 1.

III. The new ATPG platform GATE

Each integrated tool has a separate graphical window (GUI for ATALANTA, HOPE, delay faults, and toggle test) where all options can be specified and outputs are reported. The user can either specify options by simple clicks or use default commands. The commands are also memorized for the next usage. A test set for stuck-at-0 and stuck-at-1 is generated by ATALANTA or using a separate random or exhaustive test patterns generator that is a part of the platform. Although a random test patterns generation is a part of ATALANTA and HOPE, the separate random generator is dedicated to educational tasks.

Generation of a test set for delay faults is based on a test set for stuck-at faults and the transition fault model [9]. The test patterns are grouped into test pairs and fault coverage is estimated by an integrated fault simulator for transition faults. A specific test set, known as a *toggle test* (based only on changing logical values at signal lines of a tested circuit), is also generated in the platform. This test can be used in I_{DDQ} testing where changes of logical values need not to be visible on primary outputs. I_{DDQ} testing is an alternative testing of CMOS digital circuits where current is measured in steady state when logical values are changed on signal lines of the tested circuit [9]. Such types of test set generation haven't been found as free tools.

When the user selects a VHDL circuit description, it is automatically translated to the "bench" format and saved. Both descriptions can be then displayed and edited, if necessary. The parser VHDL2BENCH accepts the VHDL description only at the structural level of combinational and sequential (using only D flip-flops) circuits.

IV. Implementation

The GATE front-end is divided into several tabs. Under the tab "HOPE" there is situated a GUI for HOPE and the tab "ATALANTA" consists of GUI for ATALANTA. The tabs "DELAY FAULTS" and "TOGGLE TEST" are used for grouping a test set into test patterns pairs for checking logical values transition with or without their propagation to primary outputs. The tabs "INTRO" and "HELP" describe GATE functionalities. The GATE platform is compatible only with Windows Vista and newer.

A. GUI for HOPE

The tab for HOPE is divided into three sections:

1. loading a circuit description,
2. test generation - randomly and exhaustively,
3. fault simulator HOPE.

The user can choose the type of input file in the first section with its visibility in a text box. There is also a possibility to generate an exhaustive or random test set for a loaded digital circuit. The user can choose the number of test patterns to be generated by the random test patterns generator. The generated test set is written in ".test" file in the format that is provided by HOPE. The test set can be applied to fault simulation. There is also an option to show and edit the generated file with the test set. The last section provides all functions that are provided by the HOPE console application but in a more user-friendly form. There are situated all options in this section and the user can choose what options will be used. For several options it is necessary to choose also some arguments. They are selected in a combo box, the specified number is written in the text area or a file is chosen. The user can also select the file for redirecting the standard output of HOPE to. All selected options with their arguments as well as the input circuit appear in the text area below the options as a command that will run HOPE. The command text area is editable and remembers recently used commands. Finally, the user can either run HOPE with selected options or choose one of two default commands (1. "hope circuit.bench –t circuit.test > circuit.out", 2. "hope -s 9999 -r 10000 circuit.bench > circuit.out"). The standard output of HOPE is shown in the text area situated on the right side of this tab after executing the fault simulation.

B. GUI for ATALANTA

The tab for ATALANTA is very similar to the tab for HOPE and consists of two sections:

1. loading the circuit – the same as for the HOPE tab,
2. ATPG ATALANTA – very similar to the third section of the HOPE tab.

There are only two differences between ATALANTA and HOPE tabs. The tab ATALANTA does not contain separate random and exhaustive test generation sections. The second section consists of options specific for ATALANTA.

C. GUI for TEST GENERATION FOR DELAY FAULTS

This tab consists of one button and three text areas. By clicking the button a user can choose a combinational logic circuit, for which the program generates the test set for delay faults. After clicking the button in first text area named "Bench", the circuit description in the "bench" format appears, in second text area named "Test", the test set for stuck-at faults detection in the defined circuit appears. Finally, in the last text area named "Output", the set of generated vector pairs detecting delay faults and their fault coverage appear.

V. Experimental results

Testing of functionalities and the effectiveness of the GATE platform was done using several combinational and sequential ISCAS benchmark circuits [4], [5]. Some experiments were performed using also digital circuits designed by HDL Designer. One example of using the HOPE tab is presented for the circuit s27. HOPE was launched by the following command:

```
hope C:\s27.bench -D -l s27.log -r 16.
```

Fault simulation was done with 16 random test patterns and newly detected faults by each test pattern have been saved into defined log file. Print screen of this fault simulation is shown in Figure 3. The GUI evaluation procedure was as follows:

- A bench file type was selected from the drop-down menu.
- By clicking the "Search" button, then selecting file s27.bench in the open dialog box and clicking the button "Open", a particular file describing the s27 circuit was chosen and the file path was printed to the text field "File".
- The content of s27.bench was printed to the text field "Bench format" by clicking the "Add" button.
- The command was created by selecting the option -D, the option -l with the path to the log file as an argument and the option -r with the number 16 as argument. The selection of the option is marked with a red ellipse in Figure 3. The selected options with their arguments were added to the text field "Command". They are marked with a blue frame in Figure 3.
- Fault simulation was launched by clicking the "Start" button.
- The HOPE output appeared after execution of fault simulation in the text field "HOPE output".

Experiments using the ATALANTA tab were targeted to test functionality of the VHDL2BENCH parser using several ISCAS'85 circuits [4]. The parser was tested as follows:

- VHDL file type was selected from the drop-down menu.
- By clicking the "Search" and "Open" buttons, the file describing the c17 circuit was chosen.

- The circuit c17.vhdl was then translated to the “bench” format, saved to file c17.bench and its content was printed to the text field “Bench format” by clicking the “Add” button.
- By clicking the “Show VHDL” button, the original VHDL file is shown with a possibility to make changes.

VI. Conclusion

A new graphical ATPG platform GATE has been developed and implemented over the academic ATPG tools ATALANTA and HOPE with VHDL2BENCH parser. Generation of test pattern pairs has been added to this platform for delay faults testing and has been also presented. The GATE platform can be used in education or for test set preparation of newly designed combinational and sequential logic circuits. GATE will be free for downloading and using at the academic and educative levels. Future work can be aimed to better visibility of outputs and to other VERILOG2BENCH parser.

Acknowledgment

This work was partially supported by the Slovak Science Grant Agency VEGA 1/0616/14, APVV SK-CZ-2013-0173 in Slovakia and 7AMB14SK177 in Czech Republic, and by the grant GA16-05179S of the Czech Grant Agency, "Fault-Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features" (2016-2018).

References

[1] Lee, H., K. and Ha, D., S.: “Atalanta”: an efficient forward fault simulation ATPG for combinational circuits. Technical Report, 93-12, Dep’t of Electrical Eng., Virginia, Polytechnic Institute and State University, Blacksburg, Virginia, 1991.

[2] Lee, H., K. and Ha, D., S.: An efficient forward fault simulation algorithm based on the parallel pattern single fault propagation. Proc. of International Test Conference, 1991, pp. 946 - 955.

[3] Lee, H., K. and Ha, D., S.:”HOPE”; An Efficient Parallel Fault Simulator for Synchronous Sequential Circuits. IEEE Transaction on Computer=Aided Design of Integrated Circuits and Systems. Vol. 15, 1996, pp. 18 – 1058.

[4] Brglez, F., Fujiwara, H.: A Neutral Netlist of 10 Combinational Benchmark Circuits and Target Translator in Fortran. In Proc. of the International Symposium on Circuits and Systems, 1985, pp. 663 - 698.

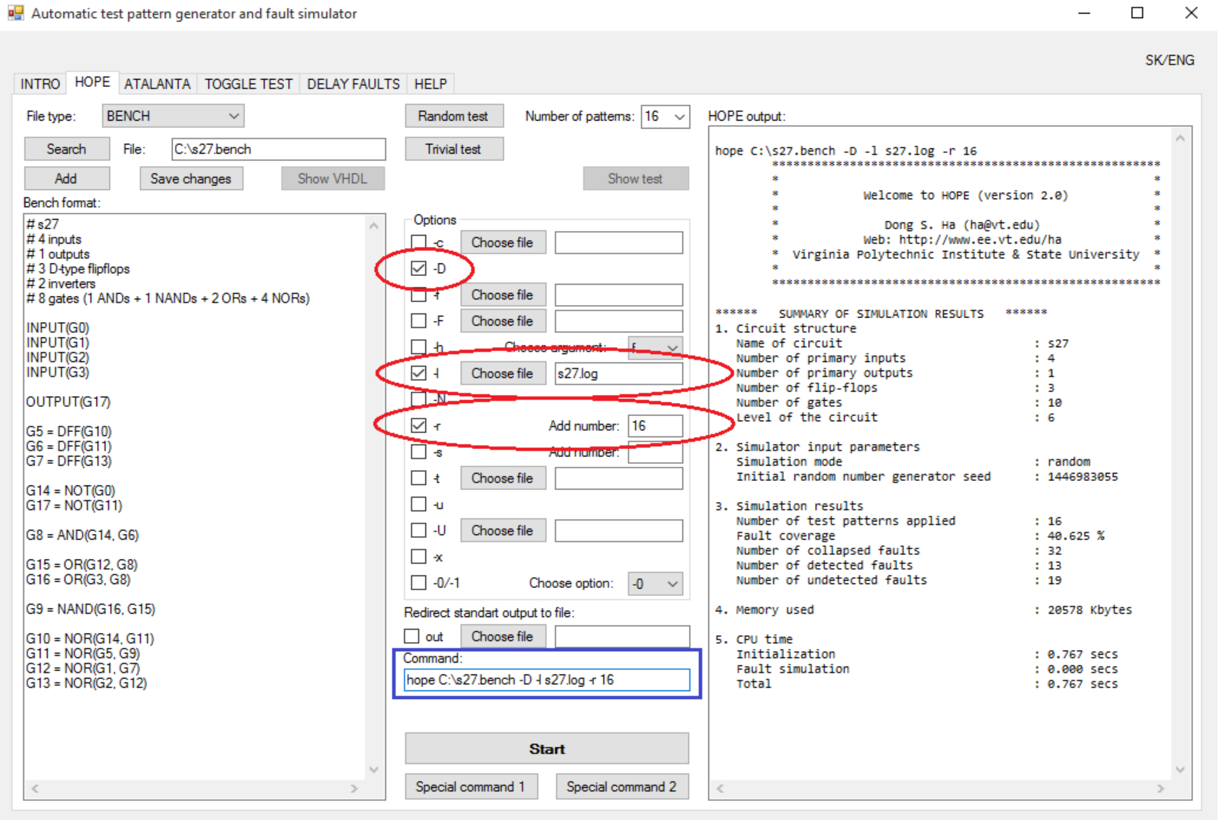
[5] Brglez, F., Bryan, D. and Kozinski, K.: Combinational Profiles of Sequential Bencjmarek Circuits. In Proc. of the International Symposium on Circuits and Systems, 1989.

[6] User's Guide for ATALANTA. Virginia Polytechnic & State University, 1991, https://github.com/hsluoyz/Atalanta [online, 29. 11. 2014].

[7] User's guide for HOPE. s.l. : Virginia Polytechnic & State University, 1991.

[8] Fujiwara, H., Shimono, T.: On the Acceleration of Test Generation Algorithms, Computers. IEEE Transactions on, Volume: C-32, Issue: 12, ISSN :0018-9340, http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1676174 [online, 23. 11. 2014].

[9] Novák, O., Gramatová, E., Ubar, R., Stopjaková, V., Drábek, V.: Handbook of Testing Electronic Systems. Praha: České vysoké učení technické v Praze, 2005. 395 s. ISBN 80-01-03318-X



File type: BENCH

File: C:\s27.bench

Bench format:

```
# s27
# 4 inputs
# 1 outputs
# 3 D-type flipflops
# 2 inverters
# 8 gates (1 ANDs + 1 NANDs + 2 ORs + 4 NORs)

INPUT(G0)
INPUT(G1)
INPUT(G2)
INPUT(G3)

OUTPUT(G17)

G5 = DFF(G10)
G6 = DFF(G11)
G7 = DFF(G13)

G14 = NOT(G0)
G17 = NOT(G11)

G8 = AND(G14, G6)

G15 = OR(G12, G8)
G16 = OR(G3, G8)

G9 = NAND(G16, G15)

G10 = NOR(G14, G11)
G11 = NOR(G5, G9)
G12 = NOR(G1, G7)
G13 = NOR(G2, G12)
```

Number of patterns: 16

Options

- □ -c
- ☑ -D
- □ -f
- □ -F
- □ -h Choose argument: f
- ☑ -l s27.log
- □ -N
- ☑ -r Add number: 16
- □ -s Add number:
- □ -t
- □ -u
- □ -U
- □ -x
- □ -0/-1 Choose option: -0

Redirect standart output to file:

□ out

Command:

```
hope C:\s27.bench -D -l s27.log -r 16
```

HOPE output:

```
hope C:\s27.bench -D -l s27.log -r 16
        ***********************************************************
        *                                                         *
        *             Welcome to HOPE (version 2.0)               *
        *                                                         *
        *                Dong S. Ha (ha@vt.edu)                   *
        *             Web: http://www.ee.vt.edu/ha                *
        *  Virginia Polytechnic Institute & State University      *
        *                                                         *
        ***********************************************************

******   SUMMARY OF SIMULATION RESULTS   ******
1. Circuit structure
   Name of circuit                              : s27
   Number of primary inputs                     : 4
   Number of primary outputs                    : 1
   Number of flip-flops                         : 3
   Number of gates                              : 10
   Level of the circuit                         : 6

2. Simulator input parameters
   Simulation mode                              : random
   Initial random number generator seed         : 1446983055

3. Simulation results
   Number of test patterns applied              : 16
   Fault coverage                               : 40.625 %
   Number of collapsed faults                   : 32
   Number of detected faults                    : 13
   Number of undetected faults                  : 19

4. Memory used                                  : 20578 Kbytes

5. CPU time
   Initialization                               : 0.767 secs
   Fault simulation                             : 0.000 secs
   Total                                        : 0.767 secs
```

Figure 3: Fault simulation for sequential logic s27.bench.