The KARL/KARATE System -
Integrating Functional Test Development
into a CAD Environment for VLSI
Gerold Alfs, Reiner W. Hartenstein, Andrea Wodtko
Informatics Dept.
Kaiserslautern University
P.O. Box 3049
D-6750 Kaiserslautern

Abstract
KARATE is a system for testability analysis and automatic test pattern generation. It is embedded in the KARL design environment. Like the KARL environment, KARATE works on hardware descriptions at RT, Gate and Switch level. A comfortable, interactive user interface makes the process of test pattern generation transparent to the user. Badly controllable or badly observable hardware components or data paths are exactly localized and the user is supported to eliminate these shortcomings. Attacking the test problem at RT level has the advantage, that redesigns in order to increase testability are cheaper than during later design phases.

1. Problem Representation
KARATE (KARl Automatic Test Extractor) [1], [2] generates and describes tests and other informations about hardware units symbolically [1], which is a new concept and much more efficient than bit vector oriented approaches. Hardware descriptions are entered either as KARL-3 (KAiserslautern Register Transfer Language) [3] text files or through the MLFD graphic editor[4]. Figure 1 shows an example of such a graphic hardware description.

The structure of a hardware unit is represented by an RTGRAPH, which contains a node for every subunit. Figure 2 shows the structure of the RTGRAPH, derived from the example hardware description of figure 1.

The graph consists of the 'example' node, representing the whole circuit and four subnodes, one for each component:
- add: a 32-bit cyclic adder
- sub: a 32-bit cyclic subtractor
- mux: a 32-bit multiplexer with a 2-bit selector input
- accu: a 32-bit positive edge triggered register

All test-related information about each particular unit is represented by so-called PATTERNs, which describe value assignments to the unit's ports. Value assignments within a PATTERN may belong to different time frames. This is necessary, if the hardware description contains sequential components such as registers. KARATE distinguishes two kinds of Patterns: TEST PATTERNs and TRACING PATTERNs.

TEST PATTERNs describe stimuli for the unit's inputs and the correct responses of its outputs. The TEST PATTERN example of figure 3 describes bivectors, that have to be applied to the 'add' submodule in order to test it. A set of eight such TEST PATTERNs tests the adder completely according to a functional fault model [5].

The adder's input ports have to be stimulated by the bivectors 1010...10 and 0101...01, respectively, and the circuits response, which is expected to be 1111...11 must be observed, i.e. propagated to a primary output of the 'example circuit'. Since the adder is a combinatorial circuit, all value assignments relate to time frame t=0.
TRACING PATTERNS describe value assignments to the unit’s ports that are used for the stimulation of the unit’s output ports by certain values or the propagation of fault effects from one of the unit’s inputs to its outputs. Figure 4 shows examples of TRACING PATTERNS for the 'add' and for the 'accu' submodules. Besides bitvectors, there are two other kinds of values, that may be assigned to ports: Variables and symbolic functions. The TRACING PATTERN of figure 4 assigns a variable X to one of the adder's input ports and a variable Y to its output. The remaining input has to be stimulated by a value, described by the symbolic function expression 'minus(Y, X)'. Here 'minus' is a symbolic function, representing cyclic subtraction. Possible uses for this PATTERN are the stimulation of port 'out' by an arbitrary value Y or the propagation of a value X at port 'inl' to the adder's output port.

Figure 4: TRACING PATTERN examples

The second TRACING PATTERN of figure 4 may be used to stimulate the register's output by an arbitrary value or to propagate a value from the input to the output port. To achieve these goals, the clock input has to be stimulated by the values 0, 1 and 0 for 3 consecutive time frames. Figure 5 shows TRACING PATTERNS for the 'mux' subunit.

Figure 5: TRACING PATTERNS for submodule 'mux'

2. Pattern Generation
A heuristic search algorithm [1] generates PATTERNS for a hardware unit from the PATTERNS of the unit’s components. Hierarchy, structure and regularities of the hardware description are exploited. To generate TEST PATTERNS for a user defined unit, all TEST PATTERNS of its subunits are applied. TRACING PATTERNS enable the propagation of the test values of one subunit through the other subunits. The application of the TEST PATTERN of figure 3 to the adder will illustrate this process.

A PATTERN application is first decomposed into elementary subtasks, which are the input for the search algorithm. In our example there are three such subtasks:
- stimulate port add.in1 with value 1010...10
- stimulate port add.in2 with value 0101...01
- propagate the value 1111...11 at port add.out

All tasks refer to an initial time frame, say time=0. Figure 6 shows the initial situation. The second task is immediately fulfilled, since port add.in2 is directly connected to a primary input.

Figure 6: Initial situation

The circuit provides only one possibility to fulfil the first task. Port accu.out has to be stimulated by value 10...10, what is

result[31..0]

Figure 7: TRACING PATTERN applied to 'ACCU'
achieved by the application of the TRACING PATTERN for the register (figure 4), where the variable X is replaced by the bitvector. Figure 7 shows the situation after application of the TRACING PATTERN. The task to stimulate port add.in1 has been fulfilled, but 4 new tasks have been created:

- stimulate port accu.clk with value 0 for time = -2
- stimulate port accu.clk with value 1 for time = -1
- stimulate port accu.clk with value 0 for time = 0
- stimulate port accu.in with value 1010...10 for time = -2

Port 'accu.clk' is connected to a primary input, so that three of these tasks are immediately fulfilled. For the remaining task, the first TRACING PATTERN for the subunit 'mux' of figure 5 is applied. The resulting situation is presented in figure 8.

The search algorithm continues to apply TRACING PATTERNS until all tasks are fulfilled. Figure 9 shows the situation after termination of search.

The resulting TEST PATTERN for the 'example' module is read from the module's ports and later transformed into a testprogram format. Different output formats are provided such as EDIF, SENTRY and SCIL, the KARL System's Simulator Command Input Language [6]. These SCIL programs may be applied to real hardware via a tester, using KARL's physical model extension [6]. Figure 10 shows a SCIL-3 fragment generated for the example TEST PATTERN.
3. Libraries

So-called 'Primitive Libraries' provide PATTERNS for each KARL basic hardware component. Other libraries, the so-called 'User Libraries' may contain PATTERNS for user defined units.

In many ATPG approaches the fault model is implicit in the used algorithms. In the KARATE system, the underlying fault model is explicitly specified by the TEST PATTERNS in the libraries.

The faults modeled are those, that would cause one of the specified outputs of a hardware unit to differ from the correct value. This concept of 'explicit fault modelling' allows the system to work with different fault models.

Not only KARATE generates PATTERNS for these libraries but also the user may enter PATTERNS. A comfortable specialized editor for PATTERNS enables the designer to make his test related knowledge available to the system.

Other specialized ATPG tools may use these libraries as an interface and enter their generated PATTERNS. The library interface also enables the integration of DFT (Design for Testability) and BIST (Built In Self Test) approaches.

4. Conclusions

In contrast to other RT level ATPG tools, which test the data paths only, KARATE includes the test of the a circuits components and allows the modification and redefinition of fault models.

KARATE has been implemented in PASCAL on a VAX 11/750 and is currently being used in an in-house pilot application test. A Primitive Library containing TEST PATTERNS for KARL primitives according to a functional fault model and a 'stuck-at' fault model based library for gate level circuit descriptions have been built up.

Comparison of performance to other ATPG tools is difficult, because KARATE belongs to a new generation of RT level tools. The combinational gate level benchmarks suggested at the IEEE Symposium on Circuits and Systems 1985 [7] are not really applicable. RT level benchmarks are not available as far as we know.

5. REFERENCES