ISCAS-85 Netlist Translator into VHDL Code

Neša P. Tomić[1] and Mile K. Stojčev[2]

***Abstract* – Benchmark circuits are mainly intended to support the research in the area of high-level automatic test pattern generation during production of VLSI ICs. For ease of development of new test tool, the benchmarks have to be standardized both in term of description styles and languages. The VHDL language has been adopted as one of the standard for description of electronic circuits. ISCAS-85 are the most popular benchmark circuits used for performance evaluation and testing of VLSI ICs, but they are described in a netlist format which is inconvenient for CAD tools. In this paper we describe an algorithm for ISCAS-85 benchmark circuit netlist translation into VHDL code.**

***Keywords* – Benchmark, ISCAS-85, VHDL, Translator.**

I. INTRODUCTION

Benchmarks are important vehicles that let industry and academia to develop new tools, compare and contrast different methodologies, and research new algorithms and techniques [1]. During the last thirty years, there have been many attempts to create and use neutral benchmarks for tool evaluation and comparison.

We will define here, benchmark as a standardized problem (circuit or circuit segment) used to compare performance of different tools and algorithms in term of speed, effectiveness, and quality of the result [2]. Typically, a benchmark set consists of a collection of circuits in a common format, which attempt to represent a range of problems for evaluating algorithms and tools within an important problem domain [3].

Nowadays, several benchmark sets are widely used. Among them, ISCAS-85, ISCAS-89, ITC'99, and Politecnico di Torino, are the most popular. In this paper we will focus our attention to ISCAS-85, especially to conversion its netlist into VHDL code, because these benchmarks represent a wide variety of problem domains and are by far the most frequently cited of all benchmarks [3].

Namely, as designers we will use reverse engineering approach in order to determine system specifications, output functions, or other design characteristics from an existing implementation, in a form of VHDL code, for a given set of benchmark circuits.

ISCAS-85 benchmark set of circuits are industrial designs whose functions and high-level designs have not been published, both for confidentiality reasons and to allow them to be viewed as random logic circuits with no significant high-level structure [4]. In this way, instead of using ISCAS-85 netlists, we will use corresponding VHDL codes, which allow us better understanding of benchmark circuit's hardware structure.

The rest of the paper is organized as follows: Section 2 gives a short overview of benchmark circuit types. Section 3 deals with ISCAS-85's netlist format example. In Section 4 the proposed algorithm concerning ISCAS-85 to VHDL code translation is given. Finally, Section 5 is Conclusion.

II. OVERVIEW OF POPULAR BENCHMARK SETS

In the sequel, we will describe briefly some of the most popular benchmark sets.

ISCAS-85: consists of collection of 10 benchmarks contributed by the number of individuals and organizations over a period of 20 years. These benchmark circuits are well-defined, high-level structures and functions based on common building blocks such as MUXs, ALUs, decoders, counters, etc [4].

ISCAS-89: These are a set of 31 digital sequential circuits. Each circuit is described in two files: a generic gate-level netlist and a list of equivalence-collapsed faults. A simple translator is included to read/write the netlist. There are no schematic diagrams [5].

ITC'99: This set of benchmarks provides a realistic example circuits to stress current automatic test-pattern generation (ATPG) algorithms, to provide impetus for the development of new automatic test-pattern generation and design for testability (DFT) algorithms, and to encourage research into fundamental DFT problems for large, complex designs [3]. They are written in either of two hardware description languages: Verilog or VHDL.

Politecnico di Torino: These high-level benchmarks are represented in synthesizable register transfer level (RTL). For their description VHDL is used.

Analog and mixed signal: This set currently include amplifier, filters, an analog/digital converters, PLL, switches and additional circuits that can be added to the set [7].

Other useful benchmark sets are available at various universities and institutes worldwide: Texas Formal Verification Benchmark, IFIP WG10.5, STEED, etc [3].

III. ISCAS-85 NETLIST FORMAT EXAMPLE

The ISCAS-85 netlist format was never formally documented, but it becomes viable despite its shortcomings, since it contains information not present in most other netlist formats.

The netlist format of a small ISCAS-85, six-NAND-gate benchmark circuit, known as "c17" [6] is listed below (see

[1] Neša P. Tomić is with the Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Serbia and Montenegro, E–mail: nesto@eunet.yu

[2] Mile K. Stojčev is with the Faculty of Electronic Engineering, Beogradska 14, 18000 Niš, Serbia and Montenegro, E–mail: stojcev@elfak.ni.ac.yu

Fig. 1), for which in Section 4, a corresponding VHDL code will be presented.

As can be seen from Fig. 1 each line consists of several columns, which are mutually separated by delimiters (space, tab, new line). Naming of each column with its description is given in Table I.

```
* These first five lines are comments.
* The comment character is the "*" (asterisk).
* The comment character may appear anywhere on a
* line and remains in effect until the end of
                    * the line is reached.
  1    1gat inpt   1   0           >sa1
  2    2gat inpt   1   0           >sa1
  3    3gat inpt   2   0    >sa0 >sa1
  8    8fan from    3gat          >sa1
  9    9fan from    3gat          >sa1
  6    6gat inpt   1   0           >sa1
  7    7gat inpt   1   0           >sa1
 10   10gat nand   1   2           >sa1
  1    8
 11   11gat nand   2   2  >sa0 >sa1
  9    6
 14   14fan from   11gat          >sa1
 15   15fan from   11gat          >sa1
 16   16gat nand   2   2  >sa0 >sa1
  2   14
 20   20fan from   16gat          >sa1
 21   21fan from   16gat          >sa1
 19   19gat nand   1   2           >sa1
 15    7
 22   22gat nand   0   2  >sa0 >sa1
 10   20
 23   23gat nand   0   2  >sa0 >sa1
 21   19
```

Fig. 1. ISCAS-85 netlist format for c17 benchmark circuit

Table I. Description of column notation

Address	Unique number for given node
Name	Descriptive string of characters related to the corresponding node
Type	Function performed by a given node (inpt, and, nand, or, nor, xnor, xor, buff, not, from)
Fanout	Number of gates connected to the output of given node
Fanin	Number of nodes which represent inputs to the given node
Fault(s)	Stuck-at-fault(s) on given node that are included in the fault set. Possible values are >sa0 for stuck-at-zero and >sa1 for stuck-at-one

Three types of lines in the netlist (Fig. 1) can be identified. The first, so called **node line** (typical for line 1-Fig. 1), gives basic information about the node. The second, referred as **fanin line** (the first line after node address 10), corresponds to the list of node addresses that drive inputs for a given node. This line appears immediately after node line with *fanin* value greater than 0. The number of addresses that appear in fanin line is identical to the *fanin* number in corresponding node line. The third type of line is called **fanout branch line** (the line with node address 8). This line use similar notation like node line, except the field *type* always takes value *from*, and information from which address this branch starts. This line appears immediately after corresponding node line (and its fanin line).

Each node has *fanout* value greater than 0, excluding primary circuit outputs that have *fanout* value 0.

More details about netlist syntax can be found in [6].

IV. Description of Translation Algorithm

Usually, when engineers create new design, they use standard building blocks for which they know functionality and hardware structure. On the other hand, benchmark circuits are “anonymous neutral circuits” of unknown functionality, but they are mainly used for efficient objective testing of the implemented algorithms. The main disadvantage of ISCAS-85 benchmark circuits is that their descriptions are given in netlist format. Such format is inconvenient for designers, who more prefer high-level or schematic circuit description.

Let us note that ISCAS-85 benchmark circuits are complex circuits with 160 to 3500 gates, 36 to 207 inputs and 7 to 140 outputs. For explanation of the proposed algorithm, without violating its generality, we have chosen one simple circuit, primarily due to limited space of this paper.

In order to obtain a descriptive readable format, acceptable for designer, a code translator has been developed. The translator accepts ISCAS-85 netlist as input, and generates VHDL code as output. The program was developed using Microsoft Visual Basic 6.0.

At the beginning of the program, a start form appears on a display (see Fig. 2), with command button for input ISCAS-85 netlist file selection (upper command button). This file must be textual (with .txt extension).

After file has been selected, we can see the message on the form, which file was chosen.

Fig. 2. Start Form

The translation process starts since the second command button is pressed (lower command button). After that, the name for the VHDL code file must be entered, the entity name for the whole circuit, and the architecture name (software offers default name as the entity name).

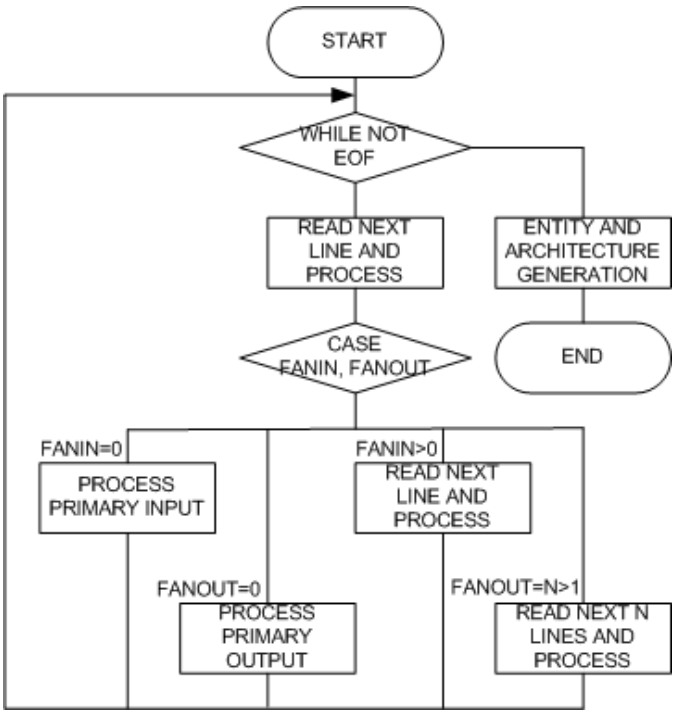

Fig. 3. Translator Algorithm

Program reads line by line, recognizes the type of line and parses data according to text delimiters (tab or space). Empty lines and lines which start with "*" (comment lines) are not taken into consideration, and software continues with reading of the next line. Parsed data are stored in corresponding dynamic data arrays, where each array represents one column from the netlist.

When one line is read, *fanin* and *fanout* values are analyzed (see Fig. 3). These values determine the type of line. If the *fanin* value is 0, this line represents node which corresponds to primary circuit input. If the *fanout* value is 0, than that line represents primary circuit output. If the *fanin* value is greater than 0, program reads the next line, where addresses of nodes, as inputs of a given node, are specified. These addresses, together with node address and its fanin number, are stored in auxiliary dynamic matrix, which is used later for port mapping.

If the *fanout* value is $n>1$, the program reads the n following lines, that represent nodes as branches from node, for which *fanout* value was n. These node addresses are also used for internal signal generation in VHDL code.

Since all lines from netlist are read, according to data structure from dynamic arrays, a VHDL code is generated. In the header of this VHDL code, the list of all gates that appear in the netlist is cited. After generation of entity definition (all primary inputs and outputs are identified), created program begins with architecture definition generation. First of all, the list with all internal signals is generated (all nodes with *fanout* greater or equal to 1). After that algorithm creates port mapping section (gate inputs connected to other gates outputs) for each node (gate), according to data structure stored in auxiliary dynamic matrix. At the end of VHDL code generation, a final message about successful translation finalization and location of the generated file (see Fig. 4) is reported to the designer.

Fig. 4. Final Message

The listing of the VHDL code which correspond to the c17 netlist is presented in Fig. 5.

```
------------------------------------------------
-- In this circuit the following gates are present
-- and must be added into CAD tool in order to complete
analysis:
--
--          nand2   -->        nand     with number of inputs 2

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


library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity c17 is
        port (
                1gat_in_port : in STD_LOGIC;
                2gat_in_port : in STD_LOGIC;
                3gat_in_port : in STD_LOGIC;
                6gat_in_port : in STD_LOGIC;
                7gat_in_port : in STD_LOGIC;
                22gat_out_port : out STD_LOGIC;
                23gat_out_port : out STD_LOGIC
            );
end entity c17;

architecture c17 of c17 is
signal int_10gat, int_11gat, int_16gat, int_19gat :
std_logic;
begin
  10gat : entity work.nand2(nand2)
    port map (1gat_in_port, 3gat_in_port, int_10gat);
  11gat : entity work.nand2(nand2)
    port map (3gat_in_port, 6gat_in_port, int_11gat);
  16gat : entity work.nand2(nand2)
    port map (2gat_in_port, int_11gat, int_16gat);
  19gat : entity work.nand2(nand2)
    port map (int_11gat, 7gat_in_port, int_19gat);
  22gat : entity work.nand2(nand2)
    port map (int_10gat, int_16gat, 22gat_out_port);
  23gat : entity work.nand2(nand2)
    port map (int_16gat, int_19gat, 23gat_out_port);
end architecture c17
```

Fig. 5. VHDL code for c17 ISCAS-85 benchmark circuit

According to the VHDL code listed in Fig. 5, the designer now can draw a schematic of the circuit, analyze its function. In our case, the schematic of c17 is given in Fig. 6.

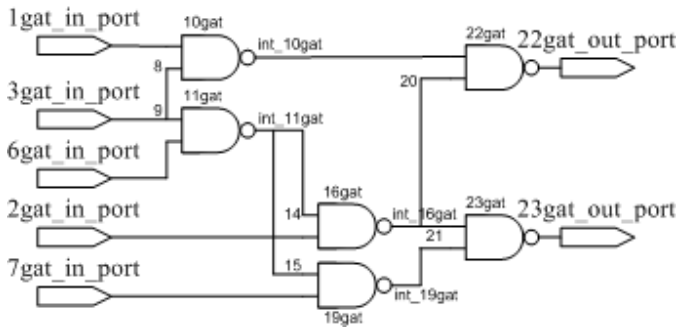

Fig. 6. c17 Schematic

The generated VHDL code obtained from ISCAS-85 benchmark circuit netlist can be used for testing some other application specific circuits. In order to realize that analysis, it is necessary to add all gates cited in the header of created VHDL code, into the VHDL project, i.e. to give their entity and architecture description.

V. Conclusion

Effective high-level automatic test pattern generation tools are increasingly needed as an essential element in the quest for reducing as much as possible the designer work on gate-level descriptions. One of the crucial parameters for speeding up and making more effective the design evolution process in any technical research is the availability of suitable and meaningful benchmark.

ISCAS-85 are one of the most popular benchmark circuits used for performance evaluation and testing of VLSI ICs. Unfortunately, they are described in a netlist format which is inconvenient for CAD tools and designers themselves. Having this in mind, in this paper we describe a suitable algorithm for ISCAS-85 benchmark circuit netlist translation into VHDL code, which, in our opinion, will be of great benefit for VLSI IC designer, during the phase of testing and performance evaluation of their solutions.

References

[1] Basto L., “First Result of ITC’99 Benchmark Circuits”, IEEE Design & Test Computers, Vol. 17, No. 4, pp. 54-59, 2000.
[2] Davidson S., Harlow J., “Introduction: Benchmarking for Design and Test”, IEEE Design & Test Computers, Vol. 17, No. 4, pp. 12-14, 2000.
[3] Harlow J., “Overview of Popular Benchmark Sets”, IEEE Design & Test Computers, Vol. 17, No. 4, pp. 15-17, 2000.
[4] Hansen M., Yalcin H., Hayes J., “Unveiling the ISCAS-85 Benchmarks: A Case Study in Reverse Engineering”, IEEE Design & Test Computers, Vol. 16, No. 4, pp. 72-80, 1999.
[5] http://www.cbl.ncsu.edu/pub/Benchmark_dirs/ISCAS89/DOCUMENTATION/iscas89.ps.
[6] http://www.cbl.ncsu.edu/pub/Benchmark_dirs/ISCAS85/DOCUMENTATION/iscas85.ps.
[7] http://www.ee.washington.edu/research/mad/benchmarks/benchmarks.html.