DESIGN VERIFICATION OF DIGITAL SYSTEMS BY SOFTWARE TESTING TECHNIQUES

Payal Gulati[1], Suman Kr Jha[2]
[1]Department of Information Technology and Computer Applications,
YMCAUST of Science & Technology, Haryana
[2]Computer Science and Engineering, Mangalmay Institute of Management and Technology

ABSTRACT

With rapid advances, the complexity of digital systems has been increased dramatically; therefore the verification of behavioural models is an important step before transferring a hardware design to the layout. This paper discusses software testing technique for verification of digital systems. The Design verification is the process of proving the correctness of the digital system design. Moreover software verification of system designs before they are fabricated on chips reduces time and money costs related to chip testing. In this paper White Box Testing is applied for verifying and validating digital systems as per requirements of the user.

KEYWORDS: SOFTWARE TESTING, VHDL, DIGITAL SYSTEMS, WHITE BOX TESTING

1. INTRODUCTION

An increasing number of applications in several fields like automotive, telecommunication, consumer electronics, etc. are being implemented using digital systems. Digital System are also coded in VHSIC Hardware Description Language (VHDL). Most of the programmers avoid VHDL language as it is tedious and time consuming [7].

Testing is the most commonly used method for verifying and validating the quality of software systems, and effective testing techniques could be helpful for improving the dependability of digital systems. Software testing can be classified as black-box (functional) testing and white-box (structural) testing [2,3,4]. The goal of black-box testing is to find circumstances in which the program does not behave according to its specifications [2]. On the other hand, white-box testing permits tester to examine the internal structure of the program. Testers can also select different testing criteria to satisfy their own needs. White-box testing can be more powerful than black-box testing in some aspects. However, white box testing can be more expensive than black-box testing, because it requires testers to understand the internal structure of target program.

In this work, architecture has been proposed to test VHDL [6,7] codes used in coding digital system using White Box Testing.

2. ARCHITECTURE

This section discusses the architecture (Figure 1.) containing sequence of steps to be followed in design verification the digital system coded in VHDL.

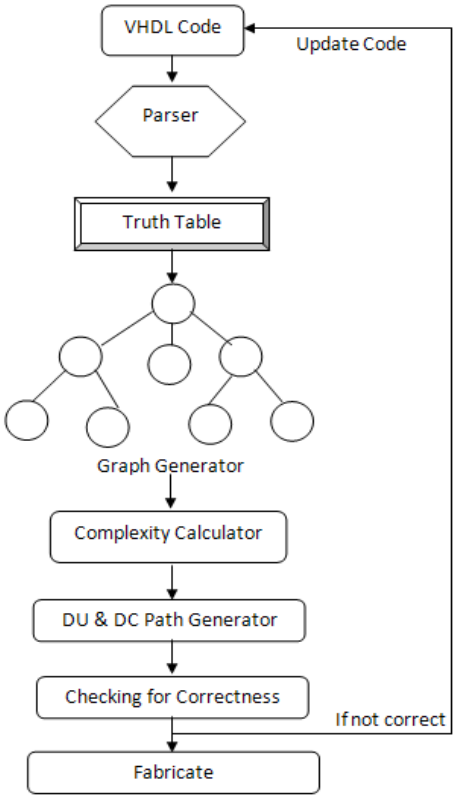

Figure 1. Proposed Architecture for Testing Digital Systems coded in VHDL

2.1. VHDL Source Code

VHDL acronym for very high speed integrated circuits hardware description language, is an IEEE standard for modelling digital systems at many levels of abstraction ranging from the algorithmic level to the gate level. This code is further synthesized and translated into a hardware device such as a field-programmable gate array. VHDL is widely used language for developing digital systems. Some researchers [1] have worked on automatic generation of VHDL source code using UML specification.

2.2. Parser

Parser parses all the statements in VHDL source code. VHDL statements include definition statements, control flow statements, signal assignment statements, variable assignment statements. Definition statements are used to find what input and output variables are present in the code. Control flow statements determine the flow of control in the program. It places assertions, which specify the conditions under which the flow of statements take place. Assignment statements such as signal assignment and variable assignments statements are for assigning value to some variable. In this work, parsing phase takes VHDL source code as input and based on the assignment statements it produces the truth table.

2.3. Truth Table

Truth table is a tabular representation of the logic function. These are used to compute the values of propositional expressions in an effective manner that is sometimes referred to as a decision procedure. Truth table is generated as the output of the parser. Linked list can be used to represent different columns and rows in a truth table. Sample Linked List structure is shown in Figure 2.

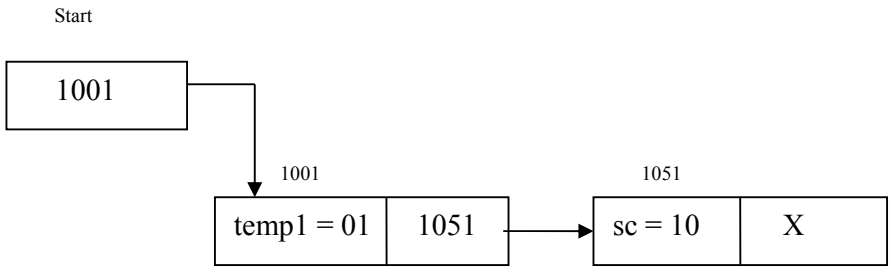

Figure 2. Sample Linked List

2.4. Graph Generator

Graph Generator converts the truth table in system C code using VHDL to C converter and then generated code is converted in a control flow graph determining the control flow of the code. The output control flow graph is further inputted to complexity calculator which is further analyzed.

2.5. Complexity Calculator

This calculates the cyclomatic complexity of the flow graph. This complexity can be calculated by the following methods:

2.5.1. On the basis of number of edges, nodes & connected Regions

e – n + 2p where e is the number of edges, n is the number of nodes and p is the connected region

2.5.2. On the basis of number of decision nodes

Cyclomatic complexity of a flow graph is equal to number of decision nodes plus 1.

2.5.3. On the basis of number of regions

Cyclomatic complexity is equal to number of regions of the flow graph.

2.6. DU and DC Path Generator

The du-paths and dc paths describe the flow of data across source statements from points at which the values are defined to points at which the values are used. The du-paths that are not definition clear are potential trouble spots. In this module all the du and dc paths are identified.

2.7. Checking for Correctness

The du-paths that are not definition clear are checked and if not correct, VHDL source code is again modified. If the code found correct, fabrication is done in the further phase.

2.8. Fabricate

Finally the code tested is fabricated i.e. implementing the design on the target technologies such as FPGA's.

3. SAMPLE WORKING

This section shows the working of steps in proposed architecture (Figure 1). Let us test the VHDL code for half adder. VHDL source code for half adder is shown in Figure 3. below:

```
library ieee;
use ieee.std_logic_1164.all;
entity ha is
port(a , b:in std_logic; sc: out std_logic_vector(0 to
1));
end ha;
architecture ha_beh of ha is
Signal temp1 :std_logic_vector (0 to 1);
begin
process (a,b)
begin
temp1 <= a & b;
case temp1 is
when "00" => sc<="00";
when "01" => sc<="10";
when "10" => sc<="10";
when "11" => sc<="01";
when others => sc<="XX";
end case;
end process;
end ha_beh;
```

Figure 3. VHDL code for Half Adder

Representation of Truth Table in Linked List form of the above mention code is shown in Figure 4. as follows

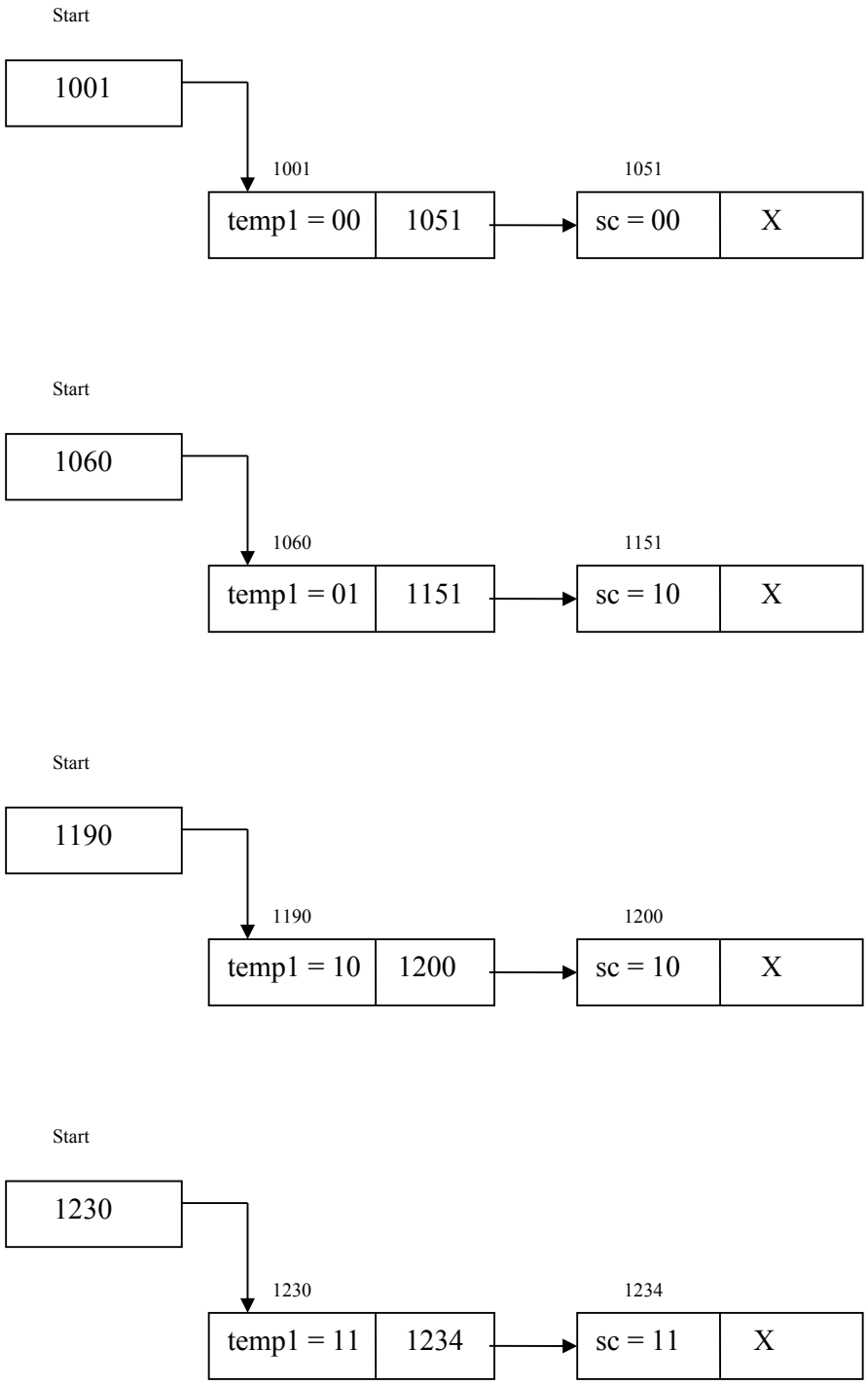

Figure 4. Linked List

C code converted from VHDL source code using VHDLtoC converter is shown in Figure 5.

```
#include     "ha.h"
#ifdef MTI_SYSTEMC
    SC_MODULE_EXPORT(ha);
#endif
void ha::process_line9( )
{
    temp1.write( ( a.read( ), b.read( ) ) );
    switch( temp1.read( ) )
    {
        case 0:
            sc.write( ( sc_uint < 2 > )( "0b00" ) );
            break;
        case 1:
            sc.write( ( sc_uint < 2 > )( "0b10" ) );
            break;
        case 2:
            sc.write( ( sc_uint < 2 > )( "0b10" ) );
            break;
        case 3:
            sc.write( ( sc_uint < 2 > )( "0b01" ) );
            break;
        default:
            sc.write( ( sc_uint < 2 > )( "XX" ) );
            break;
    }
}
```

Figure 5. System C code of Half Adder

Half adder flow graph is shown in Figure 6.

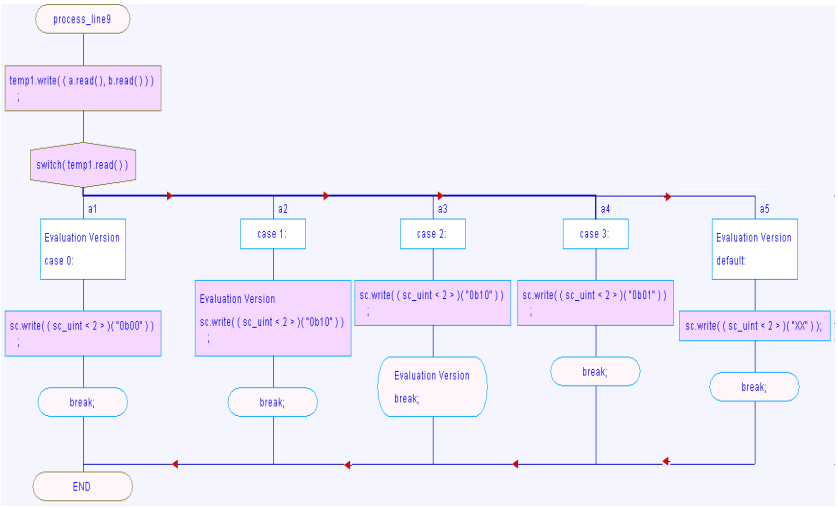

Figure 6. Flow Graph

Cyclomatic complexity of a flow graph (Figure 6.) can be calculated as

(i) Edges in this case are 22, nodes are 19 and connected region (p) is 1. So, cyclomatic complexity according to formula e – n + 2p is 22 –19 + 2= 5.

(ii) Cyclomatic complexity on the basis of number of decision nodes is 4+1 = 5.

(iii) Cyclomatic complexity on the basis of number of regions is 5.

Du-paths that are not dc-paths based on the code in Figure 5. are identified as shown in Table 1. and Table 2. below.

Table 1. DU paths

Variable	Defined at Node	Used at Node
temp1	7	7,8,11,14,17,20,23
Sc	3	11,14,17,20,23

Table 2. DC paths

Variable	Path	DC ?
temp1	7-7	No
	7-8	No
	7-11	No
	7-14	No
	7-17	No
	7-20	No
	7-23	No
sc	3-11	No
	3-14	No
	3-17	No
	3-20	No
	3-23	No

CONCLUSION

Digital systems are becoming increasingly ubiquitous, controlling a wide variety of popular and safety-critical devices. Testing is the most commonly used method for validating software systems, and effective testing techniques could be helpful for improving the dependability of these systems. In this work White box testing approach has been applied on VHDL code and thus digital systems are verified as per user specifications.

REFERENCES

[1] Moreira, T.G. "Automatic Code Generation for Embedded Systems : from UML Specification to VHDL Code", Industrial Informatics (INDIN), 8th IEEE International Conference, August 2010.

[2] J.L. Dalley. The art of software testing. In *Aerospace and Electronics Conference, 1991 NAECON 1991., Proceedings of the IEEE 1991 National*, pages 757 –760 vol.2, May 1991.

[3] Matthew B. Dwyer and Lori A. Clarke. Data flow analysis for verifying properties of concurrent programs. In *Proceedings of the 2nd ACM SIGSOFT symposium on Foundations of software engineering*, pages 62–75, 1994.

[4] All about Code Coverage - A White Box Testing Technique by Yogindernath Gupta available at http://ezinearticles.com/?All-About-Code-Coverage---A-White-Box-Testing

[5] White-Box Testing by Laurie Williams published in 2006.

[6] J Bhaskar, VHDL Premier

[7] http://www.wikipedia.org/