A&QT- R 2002 (THETA 13)
2002 IEEE-TTTC – International Conference on Automation, Quality, Testing, Robotics
May 23-25, 2002, Cluj-Napoca, Romania

DISTANCE LEARNING OF BDDs USED FOR TEST PATTERN GENERATION

Liviu Miclea*, Carmen Vaida, Enyedi Szilárd***
**Technical University of Cluj-Napoca, Automation Department, Baritiu 26-28 Cluj-Napoca, Romania, Liviu.Miclea@aut.cluj.ro, Szilard.Enyedi@aut.cluj.ro*
*** S.C. IPA S.A. CIFATT Cluj, Republicii 109, 3400 Cluj-Napoca, carmen.vaida@automation.ro*

Abstract: The paper follows distance learning in testing of digital systems. The approach presented here works on boolean functions using binary decision diagrams and also makes use of the PHP scripting language for remote data generation and collection. The method of boolean differences was chosen due to the fact that the handling packages for binary decision diagrams have included functions that process boolean differences. PHP was chosen mainly because its free nature suits academic institutions and because of its resemblance to C, a language taught in many academic institutions.

Key words: distance learning, BDDs, testing, boolean difference, PHP

1. INTRODUCTION

Boolean algebra represents a fundamental base in computer science and in design of digital systems. A lot of the problems that appear in designing and testing for digital logical systems, artificial intelligence can be expressed through operation sequences applied to boolean functions. Such applications would take benefit of efficient algorithms by symbolic representation and boolean function manipulation. It has been developed a great variety of methods for representation and manipulation of boolean functions. Those that are based on classical representations like truth tables, Karnaugh tables or canonic sum of products , are rather unpractical because every function of *n* arguments has a representation of 2^n order or larger. This kind of representations has certain disadvantages. In the first place, some of the functions require anyway representations that will have exponential order size. Secondly, while a certain function has a reasonable size representation, application of a simple operation such as complementing, could produce a function which has exponential representation. Finally, none of these functions has a canonical form; a given function could have more than one possible representation.

Because of these characteristics, the majority of programs that process an operation sequence on boolean functions, usually have an erroneous behavior.

They proceed a time with a reasonable speed, but at one given moment “explode” – which means that either there is no sufficient memory left, or fail in finalizing an operation in reasonable time. The representation that uses binary decision diagrams has a lot of advantages comparing with previous approaches as regards boolean function manipulation.

Most common functions have a reasonable representation. For example, all symmetrical functions are represented through graphs where the number of nodes is rising at the most with the square of argument number. Another advantage is that the performance of a program which is based on this algorithm, during of processing of an operation sequence falls very slowly or at all.

2. ROBDD (REDUCED ORDERED BINARY DECISION DIAGRAMS)

A binary decision diagram (BDD) is the representation of a function as a graph
For any node v that is contained by a function graph, the subgraph given by v is defined as the graph formed of v and all its descendents.This representation uses *binary decision diagrams* introduced by Lee and forward popularized by Ackers.

A binary decision diagram is called *ordered* if each variable is meet at most one time across each way from root to the terminal node and the variables are meet in the same order on all ways.

A binary decision diagram is *reduced* if it doesn't contain nodes which have isomorphic subgraphs , or nodes for which both edges that derive from that specific node point to the same terminal value. A function graph can be reduced without changing the initial function that designate the graph by eliminating the redundant nodes and duplicate subgraphs.

3. FUNCTIONAL TESTING USING BINARY DECISION DIAGRAMS

The objective of functional testing is to *validate the correct operations, which are executed by the system,* according to the functional specification. An approach assumes functioning fault models and will try to generate test that detects the faults defined by these methods. The most simple fault model is to be considered that in any line of circuit can appear a fault that causes its permanent remaining at logical 1 or 0. If the logical value of the line is 1, we will say that the line is stuck at 1 (s-a-1), and if the logical value is 0 then will say that the line is stuck at 0 (s-a-0). For testing a fault of type s-a for a certain line, this must be made sensitive to a logical value that is opposite to the considered blocking value. If an internal line of the circuit is stuck at 1 then the primary inputs must be controlled such as in the normal circuit the value of this line will be 0. After the fault will be made sensitive, the effect will be seen at a primary output. There is necessary to know the expression of the logical function of the circuit that will be tested and then this expression will be transformed in the associated BDD. For an internal node that has the label i, will follow the left or right edge , according to the variable value: 0 or 1.

4. THE BOOLEAN DIFFERENCE METHOD

The concept of boolean difference was introduced by Bob Brighton and is defined as the result of applying "exclusive-or" operation on two cofactors.

Definition: Let f be a function, where f (x_1,x_2, …,x_i,…, x_n).The boolean difference of f in relation with variable x_I is:

$\frac{df(x)}{dx_i} = f(\overline{x_i}\ ) \oplus f(x_i)$; if $\frac{df(x)}{dx_i}$ = 0 then f is independent of variable x_i, or if $\frac{df(x)}{dx_i}$ =1 then f is sensitive at x_i value.

5. BDD PACKAGE

The package used is one for manipulation of logical functions using BDD-s, and it allows graphical mode display of BDD-s. The functions are represented using ROBDDs.

Notes:

It will be used the following mark: s_a_0 for stuck at zero and s_a_1 for stuck at 1.

The following BDD diagrams are the result of using BDD package for the given circuits; the left edge corresponds to the 1 value of variable in the circle and the right one corresponds to the 0 value of variable.

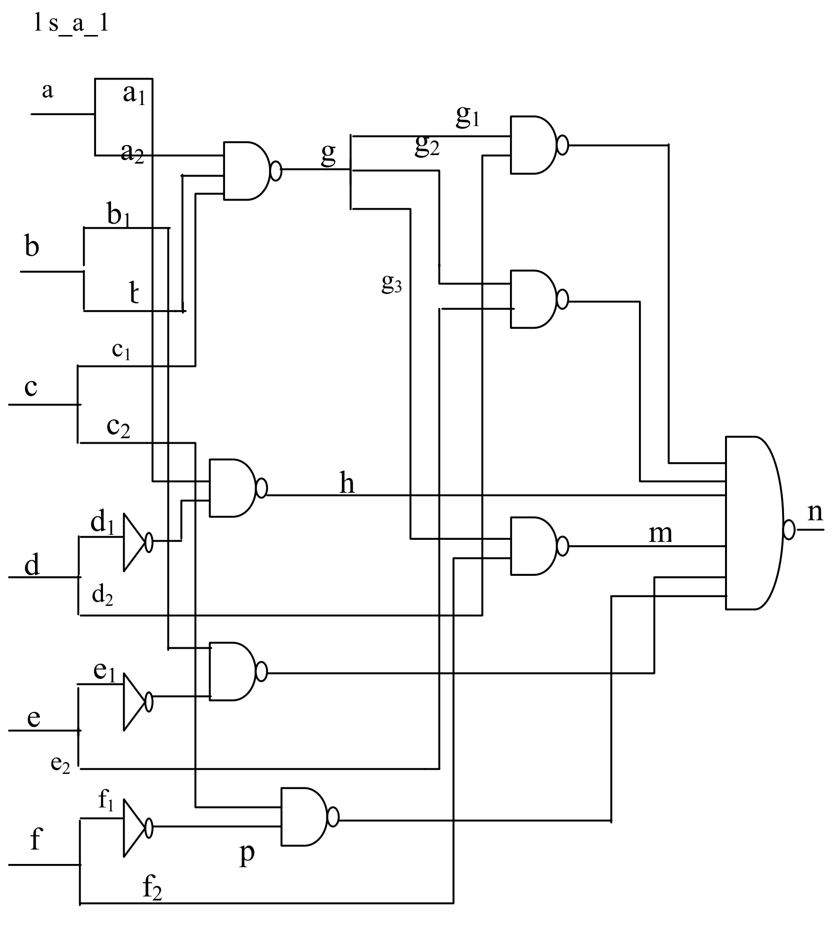
When the variable value is equal with X, means that the respective value is indifferent.

In circuit structures, in the x/y notation, x is the value of line without fault, and y is the value of line with fault. The fault will be detected at the output if $x \neq y$.

Example

Function expression:

n= (a and (not d)) or (b and (not e)) or (c and (not f)) or ((not (a and b and c)) and d) or ((not (a and b and c)) and e) or ((not (a and b and c)) and f)



Satisfying combinations for l s_a_1 fault detection:

a	:= 0
d	:= 0
b	:= 0
e	:= 0
l	:= X
c	:= 1
f	:= 0

a	:= 1
d	:= 1
b	:= 1
e	:= 1
l	:= X
c	:= 1
f	:= 0

Thus the satisfying combinations are: 000010 and 111110 respectively for a, d, b, e, c, f.

Fig. 1 – schematic of example circuit

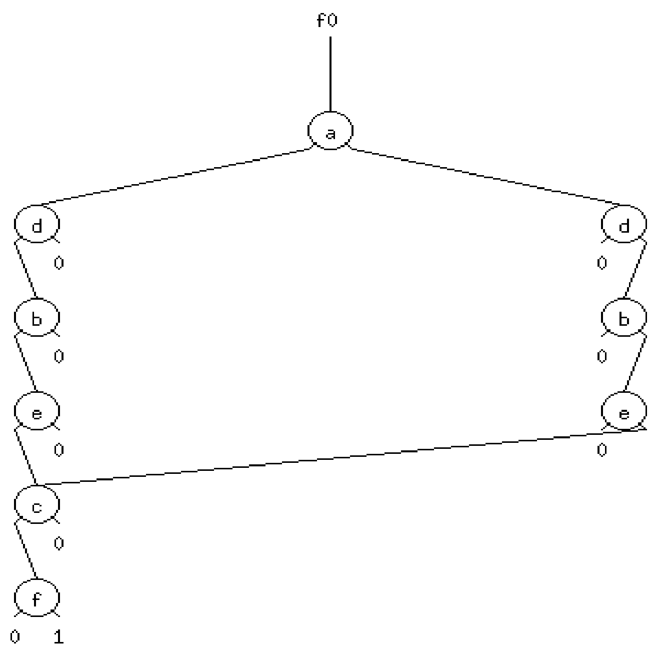


Fig. 2 – reduced BDD of example circuit

6. APPLIANCE OF BDD METHOD FOR SEQUENTIAL CIRCUITS

Any sequential circuit can be modeled by an *iterative array* as it shows Fig.3.

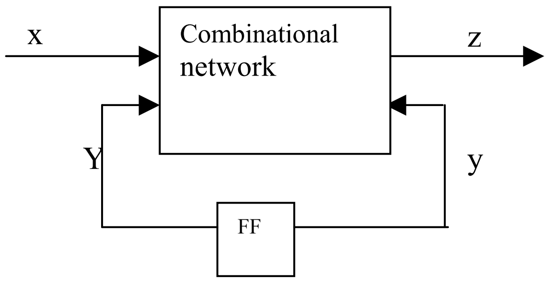


Fig.3 - the structure of a sequential circuit

The steps that will be followed in order to obtain a psedocombinational circuit from a sequential one are:

- extract, from the sequential circuit, his combinational part *C*
- generate k copies of the cell *C*: *C(0)…C(k-1)*
- connect the *k* copies by the *k-1* pseudo flip flops

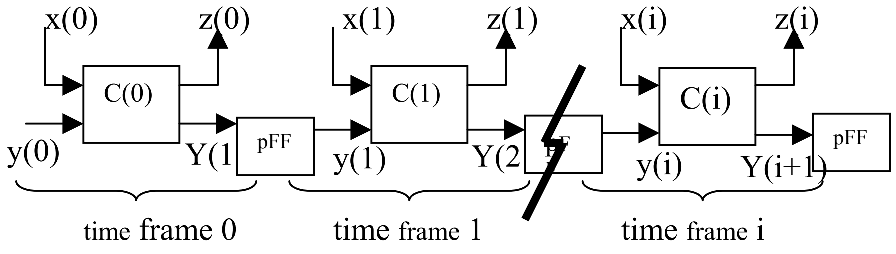
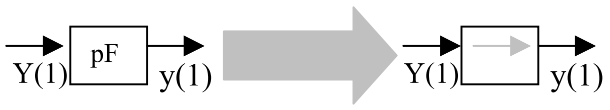


Fig.4 - iterative array corresponding to structure from Fig.3

The flip-flops are modeled as combinational elements and the present state of the flip flop in cell *i* must be equal to the state output of the flip flop in cell *i-1*. The corresponding cell is referred to a “pseudo-flip flop”.



The following figures present an example of how can be obtain a "pseudocombinational" circuit from a sequential circuit:

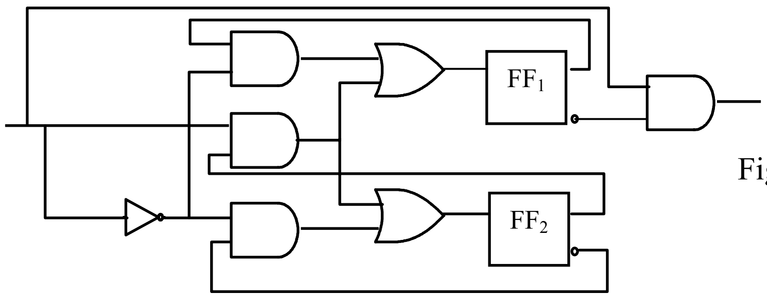


Fig.5 - an example of sequential circuit

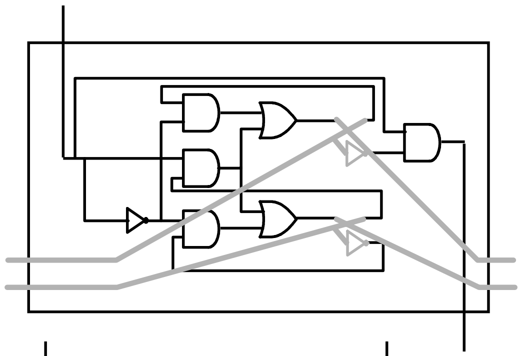


Fig.6 - the i-th cell from the array

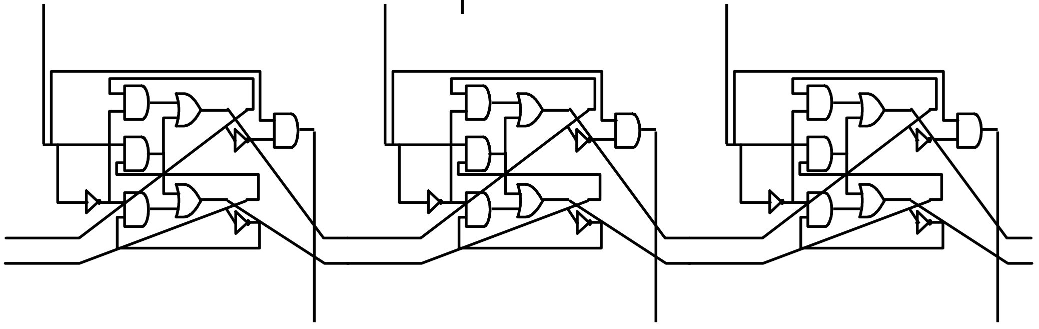


Fig.7 - the cells concatenation

From this point it can be obtained the expression of logical function corresponding to the "pseudocombinational" circuit and further we can apply the BDD method previously presented using the BDD package.

7. PHP

The other major part of the project, beside the BDD manipulation, is the PHP-based distance learning mechanism. Figure 8 shows the basic structure of the system. The user inputs or selects the desired tests in the browser, and sends them to the server. The server runs a php script, which interacts with the test engine. Currently, at simulation level, the script can execute the testing on user-given circuit descriptions or on previously created files on the server.

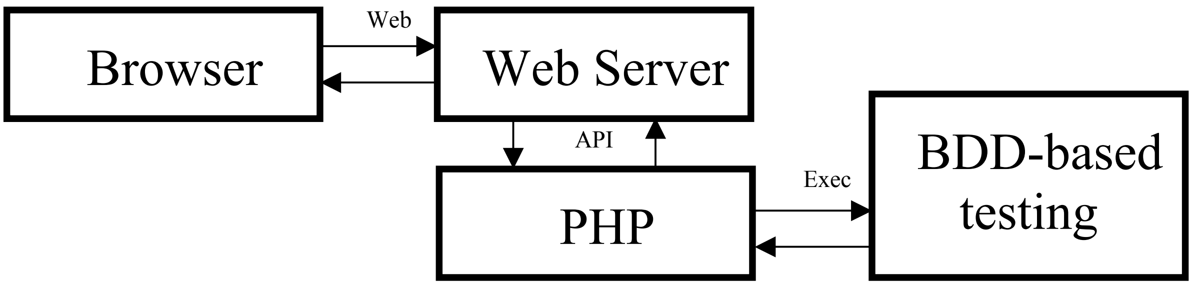


Fig. 8 – Distance data generation and collection using PHP

8. CONCLUSIONS

The testing methods that use binary decision diagrams provide a better speed processing and lower memory consumption then other methods from this field. PHP is free and its syntax is close to C's. Future improvement plans include moving the circuit description files to an SQL-compliant database and connection to a remote examination system for unattended student examination.

9. REFERENCES

[1] Randal E. Bryant, (1986), "Graph – Based Algorithms for Boolean Function Manipulation", IEEE Transactions on Computers,Vol. C-35, NO. 8
[2] S. B. Akers, (1978), "Functional testing with binary decision diagrams" , IEEE Conf. Fault-Tolerant Comput.
[3] Abramovici, M., Breuer, M.A., Friedman, A.D., (1990), "Digital systems testings and testable design, New York, Computer Science Press
[4] Liviu Miclea, (1998), "Fiabilitatea şi Diagnoza Sistemelor Digitale", Editura U.T. PRES Cluj-Napoca
[5] Randal E. Bryant, (1995), "Binary Decision Diagrams and Beyond: Enabling Technologies for Formal Verification", Pittsburg, Carnegie Mellon University