Fast Computation of Data Correlation using BDDs

Zhihong Zeng [1]
Avery Design Systems, Inc.
Andover, MA
zzeng@avery-design.com

Qiushuang Zhang, Ian Harris, Maciej Ciesielski
Dept. of Electrical & Computer Engineering
Univ. of Massachusetts at Amherst, MA 01003
{qzhang, harris, ciesiel}@ecs.umass.edu

Abstract
Data correlation is a well-known problem that causes difficulty in VLSI testing. Based on a correlation metric, an efficient heuristic to select BIST registers has been proposed in the previous work. However, the computation of data correlation itself was a computational intensive process and became a bottleneck in the previous work. This paper presents an efficient technique to compute data correlation using Binary Decision Diagrams (BDDs). Once a BDD is built, our algorithms take linear time to compute the corresponding data correlation. The experimental results show that this technique is much faster than the previous technique based on simulation. It enables testing approaches based on data correlation to handle more practical designs. As one of the successful applications, partial scan is demonstrated by integrating our computation results.

I. Introduction

Reconvergent fanout is a fundamental cause of the difficulty in testing for sequential circuits. That is, signal values are correlated to each other due to the existence of reconvergent fanouts. Both partial scan and partial BIST are well researched topics in VLSI testing arena. Sequential loops are understood to be an important factor in selecting flip-flops in partial scan [1]. Paper [2] even gives an algorithm to compute the optimum solutions for breaking loops. Later on, breaking loops method was also adapted to BIST [3].

Attacking the problem from another angle, Zhang and Harris [4] proposed a metric which is directly based on data correlations of a circuit. Their approach first characterizes the correlations and then breaks correlations by choosing BIST registers in partial BIST. However, an exhaustive simulation based technique was used to calculate data correlations. This limits the previous work [4] to designs with a small number of inputs and flip-flops. In order to extend the approach to larger designs, heuristics were used to constrain the transitive fanin cone from growing too large but the accuracy of the results are compromised through such heuristics.

In this paper, we present a new technique for correlation computation based on Binary Decision Diagrams (BDDs). It computes data correlations much faster than previous approaches and in an exact way. The new computation technique enables the method of breaking correlation presented in [4] to handle larger design and be more useful in practice. Our technique can finish all testcases in the ISCAS89 suite. Furthermore, experiments are conducted by applying our correlation results to guide the selection of scan registers in partial scan. Application to partial BIST can be also done in a similar fashion as in [4] and is not shown in this paper.

The remainder of this paper is organized as follows. Section II reviews some basic knowledge of data correlation and BDDs. Section III introduces how to compute data correlation based on BDDs. Section IV provides the algorithms to calculate correlation and gives experiment results. Section V concludes the paper and discuss some future work.

II. Background

A. Data Correlation

Let us briefly review the definition of data correlation presented in [4]. Given a combinational circuit with n inputs and one single output, let A be one of the inputs and Z be the output. There are total 2^n combinations of inputs, the output Z takes value 0 or 1 for each input combination. The 2^n combinations are divided into 4 groups according to their values of A and Z, shown in table I. The number of combinations for each group is listed in the 3rd column of the table. The value of p is read as "the number of input combinations in which the input A is 0 and output Z is 0". The value of q is

[1] This work was performed when the first author was at Univ. of Massachusetts at Amherst.

read in a similar fashion. Therefore, the data correlation between A and Z can be calculated by Equation 1.

input A	**output** Z	**# of comb.**
0	0	p
0	1	$2^{n-1} - p$
1	0	q
1	1	$2^{n-1} - q$

TABLE I
DISTRIBUTION OF INPUT AND OUTPUT COMBINATIONS.

$$Corr(A, Z) = \begin{cases} \frac{p-q}{p+q}, & p+q \leq 2^{n-1} \\ \frac{p-q}{2^n-(p+q)}, & p+q > 2^{n-1} \end{cases} \quad (1)$$

Data correlation is a binary value relationship between two signals in a circuit. To have an intuitive understanding of data correlation, we can think of it as the "controllability" between a signal and its transitive fanout signal in a combinational circuit. In a sequential circuit, data correlation is computed between flip-flops and primary I/Os which partition the circuit into combinational components. A detailed explanation can be found in [4].

B. Binary Decision Diagram

To compute data correlation, we need to analyze the logic function of a combinational circuit. A *Binary Decision Diagram* (BDD) [5] is a directed acyclic graph with two terminal nodes, 1 and 0, representing logical function 1 and 0, respectively. An internal node associated to an input variable has one or more incoming edges and exactly two outgoing edges, $T\ edge$ and $E\ edge$. T edge points to $T\ child$ and E edge points to $E\ child$. A path from root node to terminal node 1 represents a cube for the logic function that evaluates to 1; that is, BDD is an intelligent way of encoding all the cubes. The graph is levelized and each level is indexed by a support (input) variable. Such graph is *reduced, ordered* and hence *canonical* for a given logic function. In this paper, BDD means *Reduced Ordered Binary Decision Diagram*. Formal treatments on BDD can be found in [5]. Figure 1 shows an example circuit and its BDD structure.

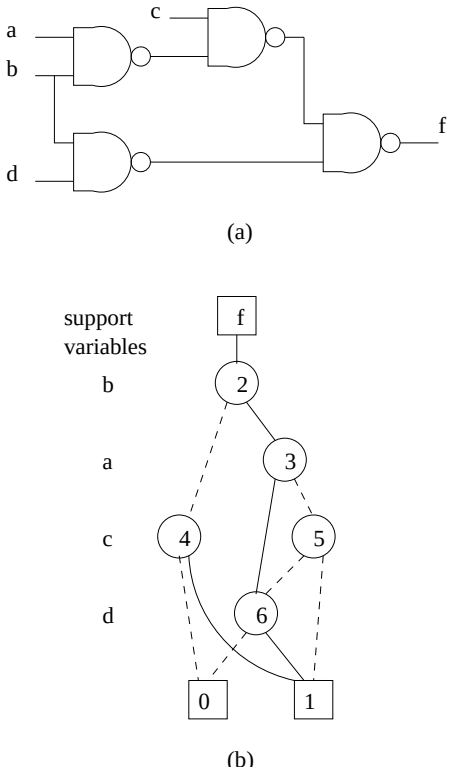

Fig. 1. An example circuit and its BDD structure.

III. COMPUTATION OF DATA CORRELATION USING BDDS

The key step in calculating data correlation is to find the value of p and q which defined in Table I. In the context of a BDD representation, for a given input variable, p is the number of combinations from the root to terminal 0 via all the E edges of this variable, while q is the number of combinations from the root to terminal 0 via all the T edges of this variable. In the sequel, we will show how to calculate p and q on a BDD structure. Consider an example in Figure 1a with its BDD structure shown in Figure 1b, where input variables, $\{a, b, c, d\}$, are shown on the left of the figure. We define each input combination has *weight* of 1, then the *weight* of an edge is the number of input combinations that consists of this edge. The weight of an outgoing edge is exactly half of the total weight of the incoming edges. The incoming edge of a root node is equal to 2 to the power of the number of input variables. Starting from the root node to terminal nodes, we can mark all the weight for each edge. For example in Figure 2, the weight of the incoming edge of node 2 is $2^4 = 16$ and the weight of each its outgoing edge is 8.

Calculation of data correlation consists of two steps.

The first step is to compute $p + q$, i.e. total number of combinations from root to terminal 0. If we normalize $p + q$ by dividing it by 2^n, the result is the *zero probability* (ZP) of a root node. By traversing a BDD from bottom up and iteratively applying Equation 2, the zero probability of a root node can be calculated. Note that the zero probability of an internal node is the zero probability of the sub-BDD rooted at this node. After the first step, a BDD is updated with zero probability associated to each node, for the above example, as shown in Figure 2.

$$\begin{cases} ZP0 = 1 & Terminal\ 0 \\ ZP1 = 0 & Terminal\ 1 \\ ZP = 0.5 * ZP_t + 0.5 * ZP_e & Internal\ node \end{cases} \quad (2)$$

where ZP_t and ZP_e are the zero probability of T child and E child of a node, respectively.

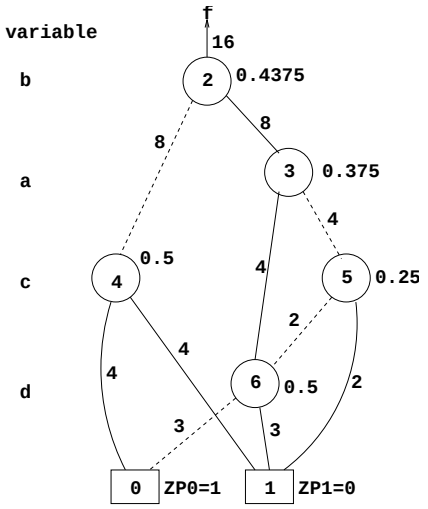

Fig. 2. Updated BDD structure with (1) weight on each edge, (2) zero probability on each node.

The second step is to compute the difference, $p - q$, for each input variable. Note that all BDD nodes on the same level are indexed by the same input variable. A *horizontal cut* on a level consists of nodes and bypass edges. For example in Figure 3a, a cut on level 2 consists of two nodes, 4 and 5, and a bypass edge, from 3 to 6. Figure 3b shows the same BDD with a dummy node replacing the bypass edge. Since both of the outgoing edges of the dummy node connect to the same node (node 6 in the example), the dummy node contributes the same value to p and q. It is easy to show that it is safe to ignore all bypass edges during the computing of $(p - q)$.

Given an input variable v, assume the number of nodes indexed by v is N_v, we can compute the difference $(p - q)$ for variable v by Equation 3, where ZP_t^n is the zero probability of the T child of node n, W_t^n is the weight of the T edge of node n, ZP_e^n and W_e^n have similar interpretations except for the E edge or child.

$$\begin{aligned} (p-q)_v &= \sum_n^{N_v} (p^n - q^n) \\ &= \sum_n^{N_v} (ZP_e^n * W_e^n - ZP_t^n * W_t^n) \end{aligned} \quad (3)$$

In order to be consistent with the sum $(p + q)$ derived in the first step, the difference $(p - q)$ is normalized by dividing 2^n. Finally, the data correlation is computed by plugging the values of the sum and the difference into Equation 1. In the above example, correlation between input c and output f can be calculated as follows.

$$\frac{p-q}{2^n} = \frac{1 \times 4 - 0 \times 4 + 0.5 \times 2 - 0 \times 2}{16} = 0.3125$$

$$Corr(c, f) = 0.3125/0.4375 = 0.714$$

IV. Experiments

A. Implementation of algorithms

We use VIS [6] as the front end to read a benchmark in *blif* format and the CUDD [7] package to build BDDs. In order to minimize the impact of possible BDD blow-up, the *transitive fanin cone* is computed for each primary output or register input so that only a single-root BDD exists in a BDD manager. Furthermore, dynamic reordering with sifting method is turned on if needed.

Algorithm 1 is to calculate the zero probability of a BDD structure corresponding to step 1 in section III. Algorithm 2 is to calculate the data correlation for each input variable with respect to a output, corresponding to step 2. Algorithm 1 is a bottom up process and algorithm 2 is a top down process. The computational complexity of both algorithms are linear in terms of the number of BDD nodes.

B. Experimental Results

Given a sequential circuit, pairs of data correlation between inputs and outputs (or register inputs) are calculated. Table II shows the CPU time of computing such data correlations on the ISCAS89 benchmarks. The overall performance is much faster comparing to the simulation based techniques used in [4], in which only

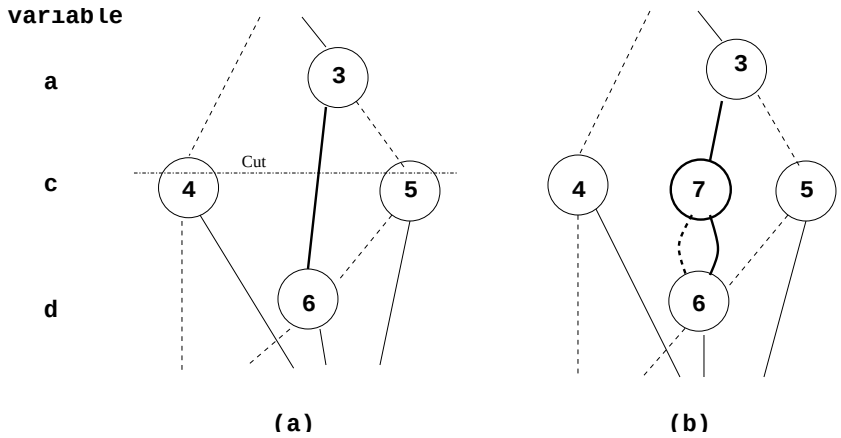

Fig. 3. (a) BDD with a bypass edge from 3 to 6. (b) BDD with a redundant node on the bypass edge.

Algorithm 1 Calculate Zero Probability

```
Read circuit
for each primary output or latch input do
  Compute its support set
  Build BDD for the sub circuit
  Classify BDD nodes into levels
  for each BDD node from level n to level 0 do
    if node is constant 0 then
      ZP(node) = 1
    else if node is constant 1 then
      ZP(node) = 0
    else
      ZP(node) = 0.5 × ZP(T child) + 0.5 × ZP(E child)
    end if
  end for
end for
```

Algorithm 2 Compute Data Correlation

```
ZP(output) = ZP(BDD root)
if ZP(output) > 0.5 then
  ZP(output) = 1 − ZP(output)
end if
for each node in BDD do
  Weight(node) = 0
end for
Weight(root) = 1.0
for each variable i from level 0 to level n do
  DiffZP = 0
  for each node in level i do
    Weight(T child) += 0.5 × Weight(node)
    Weight(E child) += 0.5 × Weight(node)
    DiffZP += 0.5 × Weight(node) × (ZP(E child) − ZP(T child))
  end for
  Corr(i, output) = DiffZP/ZP(output)
end for
```

small number of benchmarks can be finished. Our computation approach can finish all but one testcase, s38417, within 20 seconds on a PIII-850M linux machine with 512MB memory. The computation of s38417 takes much longer but is able to finish in about 5000 seconds. The computation of data correlation for each benchmark is only needed to perform once and the results can be saved for different applications.

Table III shows an application of data correlation to partial scan. After computation of data correlation, we select flip-flops by using the same algorithm in [4] to break all matched reconvergent fanouts in an S-graph. Then test patterns are generated using HITEC [8]. Test vectors length and ATPG times are shown as column 4 and column 6 in the table. The number of selected FFs is less than that of breaking loops method [1] and the test length and ATPG times are compatible with breaking loops method. If we combine both methods in the selection, we may get better results because data correlation and loops are the main causes of difficulty in ATPG. This experiment was conducted on a Sun Ultra-5 workstation (270 MHz).

V. Conclusions and Future Work

This paper presents an efficient technique to compute data correlations using BDDs. Once a BDD is built, it takes linear time (in terms of BDD size) to compute the corresponding correlations. In order to minimize the impact of BDD blow-up problem, we apply (1) dynamic reordering, (2) building only transitive fanin cone for a

test	tot. FFs	sel. FFs	test length	coverage	ATPG time (sec)
s298	14	4	149	1.0	0.03
s344	15	2	140	0.99	4.4
s400	21	5	191	1.0	1.3
s526	21	4	322	0.73	361
s641	19	5	137	1.0	0.7
s713	19	5	145	1.0	3.4
s820	5	4	399	1.0	0.7
s953	29	5	288	1.0	0.47
s1196	18	2	403	1.0	1.4
s1238	18	2	420	1.0	2.8
s5378	179	13	3493	0.96	512
s9234	228	1	6	0.99	3.3
s13207	669	49	927	0.74	6150

TABLE III

PARTIAL SCAN FLIP-FLOPS SELECTION RESULTS BASED ON BREAKING CORRELATION.

test	cpu time (sec)	test	cpu time (sec)
s208	0.03	s832	0.13
s298	0.05	s838	0.14
s344	0.08	s953	0.15
s349	0.06	s1196	0.23
s382	0.07	s1238	0.21
s386	0.05	s1423	0.51
s400	0.09	s1488	0.23
s420	0.07	s1494	0.17
s444	0.08	s5378	1.21
s510	0.06	s9234	3.17
s526	0.05	s13207	4.82
s641	0.18	s15850	10.83
s713	0.16	s35932	15.53
s820	0.10	s38417	5130.54
		s38584	19.21

TABLE II

CPU TIME OF COMPUTING DATA CORRELATIONS ON ISCAS89 BENCHMARKS.

single output, during the construction of a BDD. The computation is much faster than those in previous work as shown by the experimental results. Another set of experiments are conducted by applying our computation results to partial scan. It demonstrates a successful application with our computation technique.

However, the proposed technique might still suffer from the memory blow-up problem, which is an inherent drawback of any BDD-based approach. We suggest to use partitioned-ROBDDs [9] to alleviate the problem, in the future work, if a global BDD can not be built.

Besides applications to partial scan or BIST, data correlation based heuristics can be applied to various algorithms on gate-level circuits, such as ATPG and Satisfiability solving. In an ATPG or Satisfiability algorithm, decision heuristics are needed to choose the next branching variable. This variable should hold some properties, for example satisfying the current objective and reducing the chance of conflicting during the subsequent searching. We believe that data correlation carries some useful information on evaluation of such properties. In the future work we will investigate how our technique of computing data correlation can be combined into an ATPG or Satisfiability algorithm.

REFERENCES

[1] K.-T. Cheng and V. D. Agrawal, "A partial scan method for sequential circuits with feedback", *IEEE Transactions on Computers*, vol. 39, April 1990.

[2] S. T. Chakradhar, A. Balakrishnan, and V. D. Agrawal, "An exact algorithm for selecting partial scan flip-flops", *in Proceedings of the Design Automation Conference*, pp. 81–86, June 1994.

[3] A. P. Stroele and H. J. Wunderlich, "Hardware-optimal test register insertion", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 17, pp. 531–539, June 1998.

[4] Q. Zhang and I. G. Harris, "Partial BIST insertion

to eliminate data correlation", *IEEE Conference on Computer-Aided Design*, vol. , November 1999.

[5] R. E. Bryant, "Graph based algorithms for boolean function manipulation", *IEEE Transactions on Computers*, vol. C-35, pp. 677–691, August 1986.

[6] R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vencentelli, F. Somenzi, A. Aziz, S-T. Cheng, S. Edwards, S. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R. Ranjan, S. Sarwary, G. Shiple, S. Swamy, and T. Villa, "Vis: A system for verification and synthesis", *Proceedings of the Computer Aided Verification Conference*, vol. , pp. 428–432, 1996.

[7] Fabio Somenzi, "Cudd-2.3.1, http://vlsi.colorado.edu/ fabio", 2001.

[8] T. M. Niermann and J. H. Patel, "Hitec: A test generation pachage for sequential circuits", *in Proceedings of European Design Automation Conference*, pp. 214–218, 1991.

[9] A. Narayan, J. Jain, M. Fujita, and A. Sangiovanni-Vincentelli, "Partitioned ROBDDs: A compact canonical and efficient representation for boolean functions", *in Proc. ICCAD*, pp. 547–554, 1996.