4 JACOBS, G.M., and BRODERSEN, R.W.: 'A fully asynchronous digital signal processor using self-timed circuits', *IEEE J. Solid-State Circuits*, 1990, **25**, (6), pp. 1526–1537

5 BACKHAUS, C.: 'Performance evaluation of primitives in event-driven-logic'. Tech. Report, Centre for Applied Microelectronics, University of Gran Canaria, Spain, 1994

6 LÓPEZ, J.F., ESHRAGHIAN, K., SARMIENTO, R., and NÚÑEZ, A.: 'Gallium arsenide pseudodynamic latched logic', *Electron. Lett.*, 1996, **32**, (15), pp. 1353–1355

7 ESHRAGHIAN, K., SARMIENTO, R., CARBALLO, P.P., and NÚÑEZ, A.: 'Speed-area-power optimization for DCFL and SDCFL class of logic using ring notation', *Microprocess. Microprogr.*, 1991, **32**, pp. 75–82

OBDD-based network reliability calculation

Fu-Min Yeh and Sy-Yen Kuo

Indexing term: Reliability theory

An efficient method for evaluating the terminal-pair reliability based on an edge expansion tree and using an OBDD (ordered binary decision diagram) is presented. The effectiveness of the algorithm is demonstrated on the larger benchmarks collected in previous work. One notable case of the experimental results for a 2×20 lattice network is that the number of nodes in the OBDD is linearly proportional to the number of stages. This is significantly superior to previous algorithms which are based on the sum of disjoint products and has exponential complexity.

Introduction: The terminal-pair reliability is the probability that in a network system at least one path exists between the source and the sink. Theoretically, this reliability is the summation of the probabilities of disjoint paths, but the complexity of identifying all paths or cut sets is exponential. The terminal-pair calculation is a well known NP-hard problem [1], so the determination of reliability is very time-consuming. Therefore, research in terminal-pair reliability [2, 3] has been focused on speeding up the calculation by reducing the computation efforts as much as possible. Previous references have emphasised the improvement of two classical techniques: (i) efficient decomposition or factoring of a network with minimal sum of disjoint products (SDP) [1, 2], or (ii) given path/cut sets to reduce computing redundancy in the sum of disjoint products [3]. Although these algorithms were demonstrated with a reasonable efficiency on medium-scale networks, there still exists one inherent drawback: the sum of disjoint production forms is inefficient in dealing with larger Boolean functions.

In this Letter, an efficient method for evaluating the terminal-pair reliability based on the edge expansion tree using an ordered binary decision diagram (OBDD) is presented. First, the success-path function of a given network is constructed based on the OBDD by traversing a network with edge expansion. The reliability of the network is then obtained by directly evaluating this OBDD recursively.

Four assumptions for terminal-pair reliability are described as follows: (i) the network is modelled as a directed graph; (ii) The p_i (success) and q_i (failure) probabilities, $i = 1, ..., n$ are known for links; (iii) nodes are fault free; (iv) all failure events are mutually independent statistically.

Symbolic method with an edge expansion tree: A symbolic method for evaluating network reliability based on an edge expansion tree using an OBDD is presented. This tree diagram is able to represent all of the paths between terminal vertices. During the construction of the diagram, a multiple-level Boolean function of the paths, noted as the path function, is efficiently manipulated by the OBDD. Once the OBDD-based path function is obtained, determination of the reliability becomes straightforward. Our method consists of three main steps:
(i) We propose a heuristic approach for a good variable ordering of the OBDD-based path function. Variable ordering is advantageous in the sense that it yields a compact OBDD. Variable ordering of an OBDD-based path function is a consequence of edge variables in the network.
(ii) The path function is built using the OBDD according to the edge expansion tree.
(iii) The network reliability is obtained by recursively evaluating the probability of every node in the OBDD-based path function.

Variable ordering: For a given network G and initial index, first insert the edges connected with the source into a queue and mark these edges; then remove an edge e from the head of the queue. If the queue becomes empty, then the process is completed, otherwise, for each edge connected to e and not marked, insert it into the tail of the queue; then mark these new inserted edges. Declare a new variable of the OBDD for the edge variable e with the current value of the index. Increase the value of the index by one for the next new variable. Repeat the above steps until the queue becomes empty. This heuristic approach keeps the local property of edge variables as intact as possible and can be referred to as a breadth-first search ordering.

Path function construction with OBDD: In the following, we are to expand a given network into a tree so that the path function can be efficiently constructed by an OBDD. The basic idea of the edge expansion tree is to recursively expand edges of the source for each sub-graph instead of partitioning an s-t path, a cut-set, or an arbitrary edge factoring theorem. Our method is shown as follows:
(i) Expand the graph with edges connected to the source. Let $(e_1, e_2, ..., e_k)$ be edges linked to the source. Expand the given network G into a number of new subnetworks according to $(e_1, e_2, ..., e_k)$. The first term corresponds to e_1, the second term e_2, ..., etc.; there are k terms. For each ith term $(1 \le i \le k)$, perform *contracting operations* to obtain a new network, G^*e_i. In G^*e_i, the vertices originally linked to e_i are merged into a single vertex as a new source and all edges originally connected to it are deleted. The path function of G is determined by

$$P(G) = E_1 P(G^* e_1 + E_2 P(G^* e_2) + ... \\ + E_i P(G^* e_i) + ... + E_k P(G^* e_k)$$

where $P(G)$ is a path function of G, and E_i is a Boolean variable of edge e_i.
(ii) Remove redundant vertices to avoid the inefficient manipulation of the numerous redundant expansions. A vertex other than the source and the sink is called a redundant vertex if only one vertex is connected to it.
(iii) For each network G^*e_i, repeat the above steps until reaching the sink and then return a logical value of True. The final path function of the network is obtained by recursively composing the results of intermediate path functions.

Calculating probability from OBDD: The OBDD is based on Shannon expansion which is well known as a disjoint decomposing function. Thus, the OBDD can be recognised as being a graph-based set of disjoint products. Given the probability of each variable, the reliability of an OBDD-based function f can be recursively evaluated by

$$\text{Prob}(f) = P(x_i)\text{Prob}(f_{xi=1}) + (1 - P(x_i))\text{Prob}(f_{xi=0})$$

where x_i is the top variable, and f is partitioned into two disjoint sets $f(x_{i=1})$ and $f(x_{i=0})$, respectively. Instead of changing the original OBDD node structure, a hash table is used to avoid the redundant computation of the shared nodes in the Prob() procedure. During calculation of the probability, the number of multiplication operations and additions is of the same order as the number of OBDD nodes.

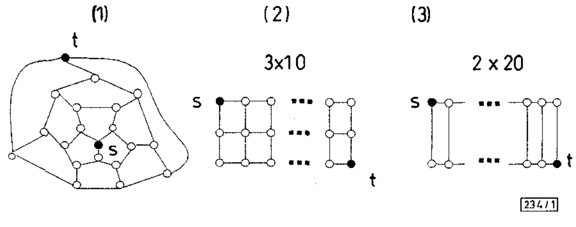

Fig. 1 *Terminal-pair networks*

Experimental results: Our method for determining the terminal-pair reliability has been evaluated on a Sun SPARC 20 workstation with a 128 Mbyte memory. All of the programs are written in

C language. In the evaluation, we have used larger networks (the number of paths > 500) which is the collection of networks [2, 3] as shown in Fig. 1. All success probabilities of links are 0.9. In Table 1, we first compare our results with those in [2, 3]. In [2], they employed the cofactor theorem to partition the network on an arbitrary edge with network reduction rules. An SDP generating method with a random and preprocessed list of path-sets was proposed in [3]. *Dpath* is the number of disjoint paths. To the best of our knowledge, the best result so far for the minimal number of disjoint products was reported in [3]. *Node* is the number of OBDD nodes in our method. A comparison of the effectiveness of the SDP form and the OBDD representation would not be straightforward because the OBDD represents a Boolean function by a graph-based set of disjoint products, which differs from the SDP with two level forms. However, we still compare the number of disjoint paths to the number of OBDD nodes of our results as a reference. For smaller networks, the performance of the SDP and the OBDD is of the same order as our experiment, but it is not shown in this Letter. When the number of network paths > 500, the number of nodes in the OBDD is significantly lower than the number of paths. However, the number of disjoint products must be greater than the number of paths. Obviously the compact OBDD size is superior to the number of SDPs in the representation of all paths for the analysis of the network reliability problems.

Table 1: Comparisons of representations of SDP and OBDD

N	Paths	Reliability	[2]	L&C [3]		EET_BDD	
			Time	DPath	Time	Node	Time
1	780	0.99712	–	53298	2.6	3591	0.4
2	49322	0.96447	240.4	–	–	257	6.7
3	524288	0.78448	–	–	–	115	58.8

N: networks, –: no data
L&C: cardinality and lexicographic ordering
[2]: on VAX 8530
[3]: on FPS 500 system

The OBDD is not only an effective representation of a Boolean function but is also a graph-based set of disjoint products [4, 5]. Once the OBDD-based path function is obtained, the network reliability is efficiently achieved by recursively evaluating the probability of every node in this OBDD. In Table 1, the computation time of [3] is the CPU time in seconds, not including the path-set generating time. All of their preprocessing times are nearest to 0.1s. The EET_BFS is a tree-based partition using OBDD with a breadth-first search ordering. *Time* is the CPU time in seconds, and includes the time of searching variable ordering, constructing path function with OBDD, and evaluating the probability of the OBDD. One noteworthy result obtained from our experiments is that for a 2 × 20 lattice network, the number of OBDD nodes is linearly proportional to the number of stages. The number of OBDD nodes is incremented by six when one stage is added. This is significantly superior to previous algorithms which are based on the sum of disjoint products and have exponential complexity. The CPU time of reliability calculation for a 20-stage lattice network is only ~58.8s with 115 nodes in the generated OBDD.

Conclusions: Our method has avoided many of the limitations of existing algorithms in manipulating larger networks, and the results obtained are superior to those of previous approaches as demonstrated by the networks of Fig. 1. With the proposed approach, we have evaluated the terminal-pair reliability of larger networks, hitherto thought impossible.

© IEE 1997 *10 February 1997*
Electronics Letters Online No: 19970549

Fu-Min Yeh and Sy-Yen Kuo (*Department of Electrical Engineering, Room 415, National Taiwan University, Taipei, Taiwan, Republic of China*)

Sy-Yen Kuo: Corresponding author

E-mail: sykuo@cc.ee.ntu.edu.tw

References

1 BALL, M.O.: 'Computational complexity of network reliability analysis: an overview', *IEEE Trans.*, 1986, **R-35**, pp. 230–239

2 THEOLOGOU, O.R., and CARLIER, J.G.: 'Factoring and reduction for networks with imperfect vertices', *IEEE Trans. Reliability*, 1991, **40**, pp. 210–217

3 SOH, S., and RAI, S.: 'Experimental results on preprocessing of path/cut term in the sum of disjoint products technique', *IEEE Trans. Reliability*, 1993, **42**, pp. 24–33

4 BRYANT, R.E.: 'Graph-based algorithms for Boolean function manipulation', *IEEE Trans.*, 1986, **C-35**, (8), pp. 677–691

5 YEH, F.-M., and LIN, C.-S.: 'OBDD variable ordering by interleaving compacted clusters', *Electron. Lett.*, 1995, **31**, pp. 1724–1725

Wired-OR property and improved structure of recovered energy logic (REL)

Chulwoo Kim and Soowon Kim

Indexing terms: Logic circuits, Logic design

A modified MOS REL structure is proposed, which exhibits the wired-OR property and enhances speed and power characteristics. Proposed MOS REL gates have been fabricated and tested. It is shown that the power × delay product of an MOS REL inverter is enhanced by 26% with a smaller silicon area.

Introduction: The adiabatic logic family 2N-2N2D [1], adiabatic dynamic logic [2], and REL [3], are advantageous in terms of the minimisation of heat dissipation and the recovery of stored energy. Among these logic devices, only logic circuits REL use a TSPC (true single phase clock) which resolves the clock skew problem with ease. In addition, although REL circuits use more transistors and are slower than CMOS logic circuits, they are suitable for low voltage systems. This Letter presents an improved MOS REL structure, which exhibits the wired-OR property that is useful in logic synthesis. Using a 0.8μm *n*-well CMOS process, the proposed circuits have been verified through their use in several test blocks.

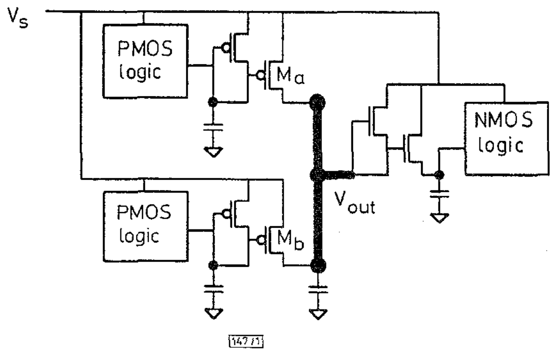

Fig. 1 *Basic concept of wired-OR*

Wired-OR property: The wiring of the outputs exhibits a wired-OR property as the wired-AND property in an open collector. As an example, a PMOS gate is shown in Fig. 1. When any PMOS gate output is high, the wired output goes high (logical 'T'), whereas the output remains low (logical 'F') as long as both PMOS gate outputs are low. It is apparent that V_{out} in Fig. 1 functions as an output of an OR gate, hence the terminology is justified. Use of the wired-OR property in logic synthesis could simplify the sophisticated digital building blocks. For example, Fig. 2*a* shows the third and fourth stages of 4bit CLA realised by REL. It is easily noted that logical operations are identical to

$$X = \overline{\overline{(A+B)} \oplus \overline{C}} = (A+B) \oplus C = Y \qquad (1)$$

However, using the wired-OR reduces the number of stages as in Fig. 2*b*. In particular, it requires only a half clock cycle for its operation.

MOS implementation of REL: An REL gate is comprised of input, precharging and output blocks. Hinman, *et al.*, has proposed a method for realising adiabatic logic circuits with BJTs and MOS-