SSBDD Model: Advantageous Properties and Efficient Simulation Algorithms

A. Jutman, J. Raik, R. Ubar
Tallinn Technical University, Estonia, artur@pld.ttu.ee

Abstract – In this paper we describe general properties of Structurally Synthesized Binary Decision Diagrams (SSBDDs) [1], which make SSBDDs very efficient for application in various structure dependent methods and algorithms. In addition, we describe four recently proposed efficient simulation methods of different classes: logic simulation, multi-valued simulation, timing simulation, and fault simulation. We investigate and show the origins of their common advantages and draw conclusions, which hold for all the described algorithms. The experiments conducted on ISCAS'85 benchmarks unveil also some new properties of these circuits, which we present in our paper.

SSBDD Model. This model has several critical features making it very attractive compared to other commonly used mathematical models, such as ROBDD [2] or a gate-level netlist. First of all, the worst case complexity (time) of generating SSBDD model from a circuit's netlist is linear in respect to the number of logic gates, while it is exponential for ROBDDs. Secondly, the size of the SSBDD model is linear in respect to the circuit size (again, ROBDD can be of exponential size). Thirdly, SSBDD model preserves structural information about the circuit while other BDD models do not. And finally, it reduces model complexity compared to the gate-level representation, as algorithms running on SSBDDs need no separate treatment of gates of different types (e.g. AND and OR gates are treated equally). Moreover, instead of considering each gate separately, it deals with *macros* – tree-like subcircuits (i.e. subcircuits with no re-convergent fanouts), which usually consist of several gates. Each single node in an SSBDD, thus, represents a whole *signal path* from a macro input to the output of the macro. This is the most significant feature, which allows development of efficient logic-level simulation algorithms. This feature provides also fault collapsing for fault simulation.

Due to the above mentioned advantages the SSBDD model has been proposed for various CAD problems like fast *deterministic* test pattern generation, efficient *design error* localization, *logic* and *multi-valued* simulation [3], *timing* simulation [4], *fault* simulation [5], *delay fault* analysis, and *fault cover* analysis in dynamic testing.

A BDD that represents a Boolean function $y=f(X)$ over a set of Boolean variables $X=\{x_1,x_2, \dots , x_n\}$ is a directed acyclic graph $G_y=(M,\Gamma,X)$ with a set of nodes M and mapping Γ from M to M. M consists of two types of nodes: non-terminal M^N and terminal M^T. A terminal node m^T is labeled by a constant $e\in\{0,1\}$, while all non-terminal nodes are labeled by variables $x\in X$, and have exactly two successor nodes. Let us denote the associated with node m variable as $x(m)$, then m^0 is the successor of m for the value $x(m)=0$ and m^1 is the successor of m for the value $x(m)=1$.

SSBDD model is not a canonical model. In spite of this, as we will see later, it is a natural BDD representation of a digital circuit. It does not rely on the Shannon decomposition. As the basis, it uses the *equivalent parenthesis form* (EPF), that is, describes a digital circuit structurally. A BDD is called SSBDD, if there is a *one-to-one correspondence* between non-terminal nodes of the BDD and signal paths in the combinational circuit. Non-terminal nodes of an SSBDD are labeled by subscripted input variables, which can be inverted or not. An SSBDD is constructed directly from a gate-level description of a combinational circuit by a graph superposition procedure. In this sense it is equivalent to EPF generation by superposition of Boolean functions. In prior to the superposition the circuit must be partitioned into a set of tree-like fanout-free subcircuits. Each such subcircuit will be represented by its own SSBDD. Therefore the whole circuit is represented not by a single SSBDD but by a set of separate SSBDDs connected by variables from extended set X.

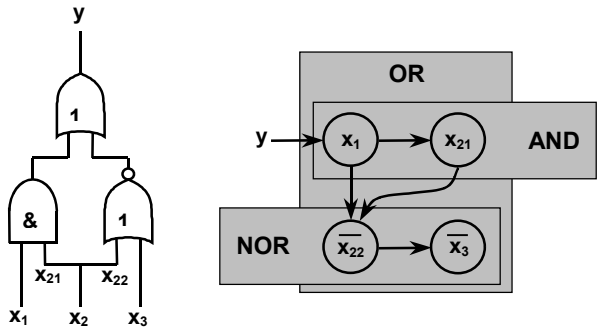

Fig.1. Illustration of the superposition principle

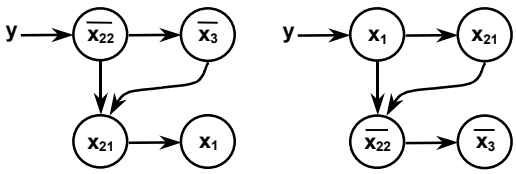

Fig. 2. Two different SSBDD representations for the combinational circuit from Fig. 1

Fig. 1 and Fig. 2 illustrate how an SSBDD is constructed from a combinational circuit and how two different SSBDDs can represent a single circuit. For SSBDDs, it is agreed that the edge corresponding to the value $x(m)=0$ always goes down while the edge corresponding to $x(m)=1$ always goes right. Terminal nodes are not shown on this picture because it is also agreed that if $x(m)=0$ and no edge goes down from the node m, then one gets to the terminal node labeled by 0. Similarly, in the case of missing "right" edge, one reaches the terminal node "1". This *alternative description stile* of SSBDD representation is possible due to introduction of the inversion of variables. Otherwise it would be impossible to represent a NOR gate using this alternative description stile. From Fig. 2 it is also seen that no separate treatment for different logic gates needed in SSBDD model. Another advantage is the modest SSBDD model size illustrated in Table 1 (compared to other BDD types).

Table 1. Comparison of sizes of different BDD models

Circuit	In	Out	Gates	ROBDD [2]	FBDD [6]	SSBDD
c432	36	7	232	30200	1063	308
c499	41	32	618	49786	25866	601
c880	60	26	357	7655	3575	497
c1355	41	32	514	39858	*N/A*	809
c1908	33	25	718	12463	5103	866
c2670	233	140	997	*N/A*	1815	1313
c3540	50	22	1446	208947	21000	1648
c5315	178	123	1994	32193	1594	2712
c6288	32	32	2416	*N/A*	*N/A*	3872
c7552	207	108	2978	*N/A*	2092	3552

Logic Simulation. Two-valued *logic simulation* on SSBDDs is equivalent to path tracing procedure on graphs according to the values of variables at a given input pattern. An assignment to the variables X activates a path $l(m_0, m^T)$ from the *root* node m_0 to a terminal node m^T. The simulation procedure consists in tracing the path $l(m_0,m^T)$ and evaluating the $y=f(x)$ by finding the value e of the terminal node m^T. It is obvious that the worst case complexity of logic simulation is equal to the total number of nodes in SSBDDs. I.e. it is also linear $O(n)$.

Multi-valued and Timing Simulation. For multi-valued simulation, we use a procedure based on calculation of Boolean derivatives on SSBDDs. Denote $l(m_i,m_j) = 1$, if there exists an activated path between the nodes m_i and m_j for a given vector x^t, otherwise, $l(m_i,m_j) = 0$. Given $y = f(x)$ and $x_i \in X$, the condition $dy/dx_i = 1$ for SSBDD $G_y = (M,\Gamma,X)$ where $x(m) \equiv x_i$ is equivalent to the following equation:

$$l(m_0,m) \wedge l(m^1, m^{T,1}) \wedge l(m^0, m^{T,0}) = 1 \qquad (1)$$

Note that equation (1) can be used for calculating Boolean derivatives only in the case where vector x^t is two-valued, because only in this case all the paths are activated uniquely. The general case, when x^t is a multi-valued vector, is considered in [3].

The timing simulation approach is based on the same principle of calculation of Boolean derivatives on SSBDDs. The difference between these methods is that in multi-valued simulation we are tracing paths to search for the nodes with variables having dynamic values while in timing simulation we are searching for nodes that switch in current moment of time (i.e. a notion of time is introduced).

Fault Simulation. In SSBDD representation the combinational circuit is partitioned into tree-like fanout-free regions (FFRs) called macros. The fault analysis procedure is based on combining the parallel backward critical path tracing inside macros with parallel forward critical path tracing between macros for FFR stem fault analysis. By using SSBDD model and shifting to the macro-level, we reduce the model complexity and achieve the natural fault collapsing

Table 2 presents the number of uncollapsed faults, collapsed faults and SSBDD faults in eight ISCAS85 circuits. As we can see from the table, the traditional fault collapsing and SSBDD representations provide almost identical results. The difference in the number of faults is at most 8 % (in the case of c1908). SSBDD achieves in average even 2 % better compaction of the fault list than the traditional approach, reducing the fault lists in average about 1.5 times.

Experimental Results and Conclusions. The experiments were carried out on ISCAS'85 circuits. Figure 3 illustrates the average speed-up obtained by using the SSBDD model for implementing the four algorithms of logic-level simulation. In all cases the speed of simulation for SSBDD model was higher than for gate-level representation [3,4]. The fault simulation algorithm (fat dashed line) shows the most noticeable acceleration. The average speed-up of other three algorithms varies from about 1,5 up to almost 4 times. The rise of simulation performance (and, in fact, reduction of the capacity of required memory) becomes

Table 2. Comparison of collapsed and SSBDD faults

Circuit	Uncollapsed	Collapsed	SSBDD
c880	1550	**942**	994
c1355	2194	**1574**	1618
c1908	2788	1879	**1732**
c2670	4150	2747	**2626**
c3540	5568	3428	**3296**
c5315	8638	**5350**	5424
c6288	9728	**7744**	**7744**
c7552	11590	7550	**7104**

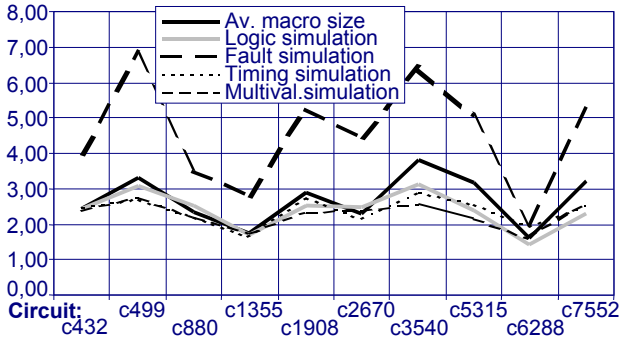

Fig. 3. Logic-level simulation speedup for different algorithms

possible due to the model complexity reduction by shifting from lower gate level to a higher macro level of fanout free subcircuits.

Very interesting property that can be seen in this diagram is that for all the circuits all the four methods give correlated to each other results. The origin of this effect is the variance of average size of macros (measured in the number of gates) for various circuits. This property is shown in Fig. 3 by the bold black line. The behavior of this line is also correlated to the behavior of the simulation curves. This unveils the fundamental property of the

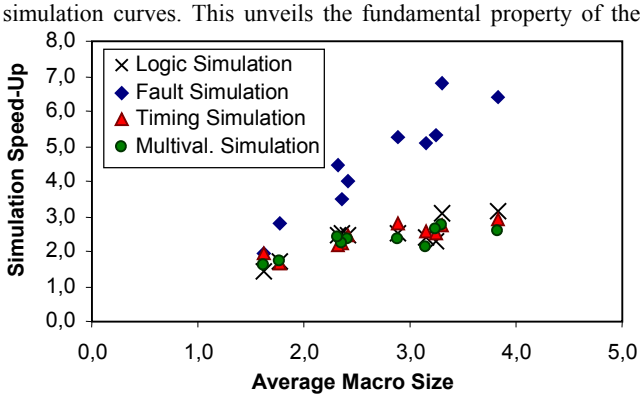

Fig. 4. Logic-level simulation speedup vs. average macro size

investigated methods that the average simulation speed-up is directly proportional to the average size of a fanout-free subcircuit in the circuit (Fig 4). The same property can be used also to describe ISCAS'85 benchmarks themselves. Now we can arrange these benchmarks (Table 3) by the average size of fanout-free subcircuits (macros). This order is significant at least for evaluation of methods, which use SSBDDs as the underlying model.

Table 3. Benchmarks' order according to the average macro size

c6288	c1355	c2670	c880	c432	c1908	c5315	c7552	c499	c3540
1,62	1,77	2,32	2,36	2,42	2,90	3,15	3,24	3,30	3,83

References

[1] R. Ubar, "Test Generation for Digital Circuits Using Alternative Graphs (in Russian)", *Proc. of Tallinn Technical University,* 1976, No.409, Tallinn, Estonia, pp.75-81.

[2] R. Bryant "Graph-based algorithms for Boolean function manipulation", *IEEE Trans. on Comp.*, 1986, C-35, pp. 677-691.

[3] R. Ubar, "Multi-Valued Simulation of Digital Circuits with SSBDDs," *Gordon and Breach Publ., Multiple Valued Logic,* 1998, Vol.4, pp. 141-157.

[4] R. Ubar, A. Jutman, Z. Peng, "Timing Simulation of Digital Circuits with BDDs", in *Proc. of DATE 2001 Conference*, München, Germany, 2001, pp. 460-466.

[5] R. Ubar, "Parallel Critical Path Tracing Fault Simulation" *39 Int. Wiss. Kolloq.*, Ilmenau, Germany, 1994, B1, pp. 399-404.

[6] W. Günther, R. Drechsler, "Minimization of Free BDDs," In *Asia & South Pacif DAC*, Hong Kong, Jan 1999, pp. 323-326