On SSBDD Model Size & Complexity

Artur Jutman

Dept. of Computer Engineering, Tallinn Technical University
Raja 15, Tallinn 12618, Estonia. E-mail: artur@pld.ttu.ee

Abstract. Binary decision diagrams (BDD) have gained a wide acceptance as a mathematical model for representation and manipulation of Boolean functions in VLSI CAD. In this paper we consider a special kind of BDDs called Structurally Synthesized BDDs (SSBDDs), which have an important characteristic property of keeping information about circuit's *structure*. Despite the fact that the SSBDD model itself is not new, we show for the first time in a formal way that this model is of *linear* complexity with respect to the number of logic gates in the circuit. We present new theoretical results in a form of equations that allow exact calculation of SSBDD model size for an arbitrary combinational circuit. Using these equations, we estimate the model complexity and compare it with the complexity of the gate-level netlist. We show in a formal way, that the SSBDD model is always smaller than the netlist still keeping the structure of the circuit.

I. Introduction

The constant increase of integration level of modern digital devices imposes high and further growing requirements on methods and algorithms used in VLSI CAD design, verification, and testing. It is clear that the efficiency of any algorithm depends heavily on the underlying mathematical model. This fact has made the search for good models for Boolean function representation and manipulation a hot topic already for several decades. In the meantime the BDD model [1] has gained a wide acceptance and became a state-of-the-art data structure in modern VLSI CAD. First works on BDDs [2, 3, 4, 5] date back to 70s and even 50s. However, the model was not widely known until Bryant proposed a new data structure called *Reduced Ordered Binary Decision Diagrams* (ROBDDs) [6] in 1986. He showed simplicity of manipulation and proved the model canonicity what made it one of the most popular representations of Boolean functions.

During the last decade, many modifications to BDD model were proposed [7]. They were mostly aimed at fighting a memory explosion problem, which limits its usability on large designs. As a result, some similar new models appeared that can represent any Boolean circuit in linear space, e.g. Boolean Expression Diagrams [8]. There is a special kind of BDDs called *Structurally Synthesized BDDs* (SSBDDs), which is also known to be of linear complexity with respect to the number of logic gates in the circuit [9,10]. However, the latter statement has always been based on intuitive speculations and experimental observations only. In this paper, for the first time, we show in a formal way that the SSBDD model always has linear complexity with respect to the number of logic gates for any arbitrary combinational circuit. Moreover, we show that this model size is always less than the size of corresponding logic-level netlist. We also provide simple equations for exact calculation of SSBDD model size. Different equations use different parameters of combinational circuit. The simplest one is based on two basic parameters only: the total number of logic gates and the total number of signal lines (or half the number of un-collapsed stuck-at faults).

Besides linear complexity, the main characteristic feature and advantage of SSBDDs compared to other classes of BDDs is their ability of keeping structure of combinational circuits. Due to this fact the SSBDD model has been proposed as a data structure for various CAD problems like fast *deterministic* test pattern generation [9], efficient *design error* localization [11], *logic* and *multi-valued* simulation [12] for different purposes (like hazards investigation, *delay fault* analysis, and *fault cover* analysis in dynamic testing). Efficient algorithms for *timing* and *fault* simulation were proposed in [13, 14]. SSBDD model was introduced for the first time in [3, 5] as *Structural Alternative Graphs* and generalized as *multiple-valued* decision diagrams in [9]. Some more details on SSBDDs as well as their advantages in the logic-level simulation domain are considered in [10].

We continue our paper with definitions and terminology in Section 2. Section 3 is the main section where all the mentioned equations are proved and a method of the SSBDD model complexity estimation is considered. Finally, we bring conclusions in Section 4.

II. Definitions and terminology

In the following we will define some notions that are used throughout the paper. Since the SSBDD model is aimed at representation of logic level digital circuits rather than Boolean functions we define a *combinational circuit* as a network of basic Boolean logic gates, i.e. AND, NAND, OR, NOR, inverter, and buffer. This restriction on component base is important. We state that our technique holds for any combinational circuit composed of the specified set of 6 basic logic gates. Circuits, containing other types of gates have to be transformed into this component base before SSBDD model can be generated.

Def.1 A connection between output of a gate and input of another gate that does not contain fanout points is called a *signal line*. If a signal connection contains a fanout point then the fanout stem and all the branches are separate signal lines. All primary inputs and primary outputs are also signal lines.

It is not difficult to see that the number of signal lines is exactly a half of the number of un-collapsed stuck-at faults (two SAFs per signal line). The circuit in Fig. 1 has 3 primary inputs, 1 primary output and 8 signal lines.

Def.2 Any part of a combinational circuit is called a *tree-like subcircuit* if it contains no fanout points inside. A tree-like subcircuit always has a single output and one or more inputs. A single logic gate or a signal line is also a tree-like subcircuit. Any combinational circuit can be partitioned into a set of tree-like subcircuits (see Figure 2). It is not difficult to notice that if the elements of such a partition are tree-like subcircuits of a maximum size, then such a partition is unique.

Def.3 Such tree-like subcircuits of a maximum size are called *macros*.

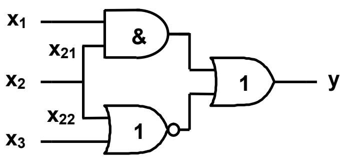

Fig 1 Combinational circuit

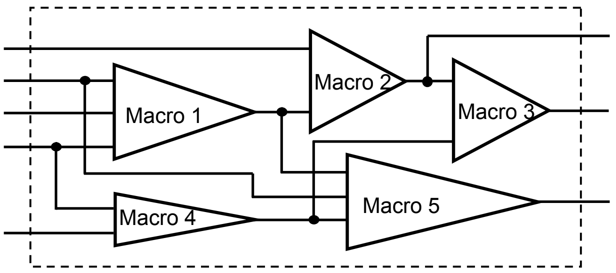

Fig 2 Partitioning of a combinational circuit

Def.4 A BDD that represents a Boolean function $y=f(X)$ over a set of Boolean variables $X=\{x_1, x_2, \ldots, x_n\}$ is a connected directed acyclic graph $G_y=(V,E)$ with a set of nodes V and a set of edges E that defines mapping from V to V. Set V consists of two types of nodes: internal (non-terminal) V^N and terminal V^T, $V=V^N \cup V^T$. A terminal node v^T is labeled by a constant {0,1} and is called *leaf*, while all non-terminal nodes $v \in M^N$ are labeled by variables $x \in X$, and have exactly two successor nodes. Let us denote the variable associated with internal node v as $x(v)$, then $low(v)$ is the successor of *v* at the value *x(v)=0* and *high*(*v*) is the successor of *v* at the value *x(v)=1*. An example of a BDD is given in Figure 3.

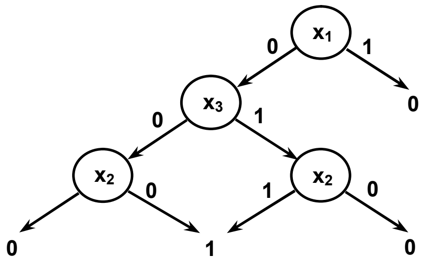

Figure 3:Binary Decision Diagram

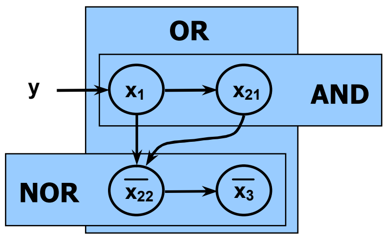

Figure 4: Illustration of the superposition principle

The majority of BDD modifications use *Shannon decomposition*: $y = \bar{x}f_{\bar{x}} + xf_x$. Here f_x and $f_{\bar{x}}$ are obtained from *y=f(x)* by replacing variable *x* by value 1 and 0 correspondingly. On the contrary, SSBDDs do not rely on Shannon decomposition. They are based upon the *equivalent parenthesis form* (EPF), that is, they describe a digital circuit structurally.

Def.5 A BDD is called *SSBDD*, if there is a *one-to-one correspondence* between non-terminal nodes of a BDD and signal paths of a combinational circuit (or a subcircuit). Non-terminal nodes of an SSBDD are labeled by indexed variables, which can be inverted or not. The SSBDD model is further defined by construction.

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 macros (Fig. 2). Each macro will be represented by its own SSBDD. That is, the whole circuit is represented not by a single SSBDD but by a set of separate SSBDDs connected by internal variables. This is the main clue why SSBDD model does not grow exponentially. It is also useful to notice that *the number of nodes in a particular SSBDD is equal to the number of inputs of the corresponding macro* and that *each node represents a whole signal path inside the macro from one of its inputs to the output*.

Figure 4 illustrates how an SSBDD for the combinational circuit from Fig. 1 is constructed. First, we set elementary BDDs: for AND and NOR gates. Then we use the superposition principle to combine them into a whole single SSBDD just as OR gate combines the two gates into a single circuit. For SSBDDs, it is agreed that the edge corresponding to the value *x(v)=0* always goes down while the edge corresponding to *x(v)=1* always goes right. Terminal nodes are not shown on this picture because it is also agreed that if *x(v)=0* and no edge goes down from the node *v*, then one gets to the terminal node labeled by 0. Similar reasoning holds in the case of missing “right” edge [9].

III. Calculation of SSBDD model size and complexity

Let us start with combinational circuits consisting of a *single macro* only. Then, we can generalize the case for an arbitrary circuit. Finally, we are going to estimate the model complexity.

A. Tree-Like Subcircuits

Let’s start with macros consisting of one-input gates only. From Figure 5 it can be seen that the number of SSBDD nodes for such circuits is exactly 1 independently of the number of one-input gates. The variable in this single node will be inverted or not depending on the number of inverters in such a one-input circuit.

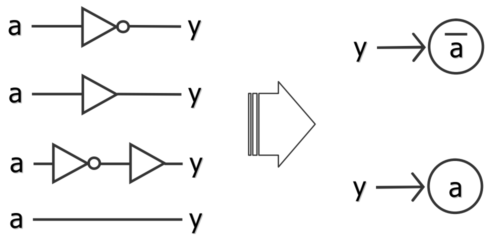

Fig 5 SSBDDs for one-input gates

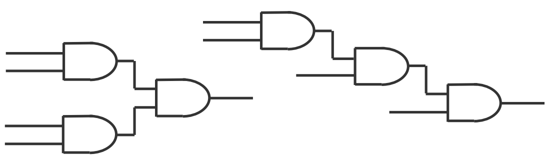
Fig 6 Macro consisting of two-input gates

The next figure shows different macros consisting of two-input gates. It is not difficult to see that the number of inputs of a tree-like circuit that consists only of two-input gates depends *on the number of gates only but not on their combination!* Indeed, if we add one more gate to any of the two circuits in Figure 6 it will add two inputs and remove one. Therefore, each additional gate adds one input in total while the very first one adds 2 inputs. Since the number of SSBDD nodes N_{nodes} is equal to the number of inputs of the corresponding macro, it can be calculated using the following equation where g is the number of 2-input gates in the macro: $N_{nodes} = g + 1$.

The same reasoning holds for macros consisting of 3-input gates. The only difference is that each additional gate adds 2 inputs except the very first one, which adds all 3 inputs (see Fig. 7). Then the size of such a macro can be calculated using the following equation: $N_{nodes} = 2g + 1$. If we continue increasing the number of inputs of the gates we will notice soon that the generalized equation for a macro consisting of *k-input* gates is the following: $N_{nodes} = (k-1)g + 1$.

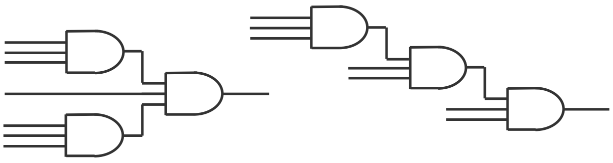
Fig 7 Macro consisting of three-input gates

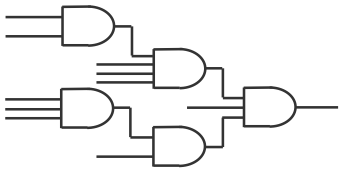
Fig 8 Macro consisting of different gates

So far we have considered quite an unusual situation where the whole circuit consists of a single macro, which in its turn consists only of gates with equal number of inputs. However, it is not difficult to generalize the equation for a macro consisting of arbitrary Boolean gates. Figure 8 clearly shows that, again, each *k-input* gate adds k inputs and takes one back (except, again, the first one). Therefore, we have to take the quantities of 1-input gates g_1, 2-input gates g_2, etc. into account separately. The generalized equation for a single macro consisting of arbitrary gates is the following:

$$N_{nodes} = 0g_1 + 1g_2 + 2g_3 + 3g_4 + \ldots + (n\text{-}1)g_n + 1$$

B. Arbitrary Combinational Circuits

Let us again consider an arbitrary combinational circuit that may consist of an arbitrary number of macros. One must not forget here, that each primary input, which is a stem of a fanout as well as each primary output that is a fanout branch, all of them are *separate macros* having one input and one output (see Figure 2). Therefore, they will be represented by a 1-node-SSBDD.

Theorem 1. The total number of SSBDD nodes for an arbitrary combinational circuit is

$$N_{nodes} = 0g_1 + 1g_2 + 2g_3 + 3g_4 + \ldots + (n\text{-}1)g_n + m \quad (1)$$

where m is the total number of macros and g_i is the number of i-input gates in the circuit.

Proof. In order to calculate the total number of nodes in the SSBDD model for an arbitrary combinational circuit consisting of *m* macros we have to sum up SSBDD sizes for each macro:

$$\begin{array}{l}
N_1 = 0g_{11} + 1g_{21} + 2g_{31} + 3g_{41} + \ldots + (n\text{-}1)g_{n1} + 1 \\
N_2 = 0g_{12} + 1g_{22} + 2g_{32} + 3g_{42} + \ldots + (n\text{-}1)g_{n2} + 1 \\
N_3 = 0g_{13} + 1g_{23} + 2g_{33} + 3g_{43} + \ldots + (n\text{-}1)g_{n3} + 1 \\
\vdots \\
\underline{N_m = 0g_{1m} + 1g_{2m} + 2g_{3m} + 3g_{4m} + \ldots + (n\text{-}1)g_{nm} + 1} \\
N_{nodes} = 0g_1 + 1g_2 + 2g_3 + 3g_4 + \ldots + (n\text{-}1)g_n + m
\end{array}$$

Here, N_j is the SSBDD size for j-th macro and g_{ij} is the number of i-input gates in j-th macro.■

The last equation allows the exact calculation of the SSBDD model size for any arbitrary combinational circuit if some detailed information (like number of gates of each type and number of fanout points) about the corresponding logic level netlist is provided. This can be done even when the model itself is not yet generated.

C. SSBDD Model Complexity

Based on the last equation, let us now estimate the SSBDD model complexity. It is not difficult to notice that, in the worst case, all the gates in the circuit have the maximum possible number of inputs *n* and the number of macros is equal to the number of gates. In this case we have:

$$N_{nodes} = (n\text{-}1)g_n + g_n = n \cdot g_n - g_n + g_n = n \cdot g_n$$

That is, the worst case complexity of the SSBDD model is *O(n·G)*, where *G* is the total number of logic gates in the circuit and *n* is the number of inputs of the largest gate. In reality *n is much less than G*. Moreover, *n* can be regarded as a constant when *G* approaches to infinity. Therefore, the worst case complexity of the SSBDD model is linear with respect to the number of logic gates.

D. Comparison of Sizes of the SSBDD Model and the Gate-Level Netlist

The gate-level netlist size can be characterized by the number of signal lines. Let us calculate this parameter in similar way as we have calculated the SSBDD size. It not hard to see (see Fig. 8), that the number of signal lines in a single macro is as follows:

$$N_{signals} = 1g_1 + 2g_2 + 3g_3 + 4g_4 + \ldots + n \cdot g_n + 1$$

We just need to sum up all the inputs of all the gates plus one primary output of the macro. If circuit consists of an arbitrary number of macros *m*, then the equation is the following:

$$N_{signals} = 1g_1 + 2g_2 + 3g_3 + 4g_4 + \ldots + n \cdot g_n + m \qquad (2)$$

If we compare equations (1) and (2) we will see that the size of SSBDD model is *always smaller* than the size of the corresponding gate-level netlist for any arbitrary combinational circuit.

E. SSBDD Size: Simple Equation

Theorem 2. The size of SSBDD model generated for an arbitrary combinational circuit is the difference between the number of signal lines and the number of gates in the circuit:

$$N_{nodes} = N_{signals} - N_{gates} \qquad (3)$$

Proof. Let us subtract equation (1) from equation (2).

$$N_{signals} - N_{nodes} = (1g_1 + 2g_2 + 3g_3 + 4g_4 + \ldots + n \cdot g_n + m) - (0g_1 + 1g_2 + 2g_3 + 3g_4 + \ldots + (n\text{-}1)g_n + m)$$

Subtracting term by term we will have the following:

$$N_{signals} - N_{nodes} = g_1 + g_2 + g_3 + g_4 + \ldots + g_n = N_{gates}$$

Therefore: $N_{nodes} = N_{signals} - N_{gates}$ ■

This result is important because it provides very facile method of estimation of the SSBDD model size based on just two main parameters of combinational circuits: the number of gates and the number of signal lines. Moreover, as it has been discussed above, the latter parameter is exactly a half of the total number of un-collapsed SAFs in the circuit, which is usually known. It also follows from the last equation that the SSBDD model is smaller than the gate-level netlist exactly by the number of gates.

IV. Conclusions

In this paper we showed in a formal way that a special type of BDDs called *Structurally Synthesized BDDs* (SSBDDs) always have linear complexity with respect to the number of logic gates for an arbitrary combinational circuit. Moreover, we showed that the SSBDD model complexity is even smaller than the complexity of corresponding logic-level netlist exactly by the number of logic gates.

We provide several equations for exact calculation of SSBDD model size. The simplest one is based on two basic parameters of combinational circuits: the number of logic gates and the number of signal lines (or half the number of un-collapsed stuck-at faults). It also represents a strict relationship between mentioned parameters and the number of nodes in SSBDD model.

References

[1] R. Drechsler, B. Becker. *Binary decision diagrams: Theory and implementation*, Kluwer Academic Publishers, Boston, 1998, 200 p.

[2] C.Y. Lee. Representation of switching circuits by binary decision diagrams. Bell System Technical Jour., 38:985-999, 1959

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

[4] S.B. Akers “Binary decision diagrams” IEEE Trans. On Comp., 27:509-516, 1978

[5] R. Ubar, “Beschreibung Digitaler Einrichtungen mit Alternativen Graphen für die Fehlerdiagnose,” Nachrichtentechnik/Elektronik, (30) 1980, H.3, pp.96-102.

[6] R. Bryant “Graph-based algorithms for Boolean function manipulation”, IEEE Transaction on Computers, 1986, vol. C-35, pp. 677-691.

[7] A.Narayan, “Recent Advances in BDD Based Representations for Boolean Functions: A Survey”, in Proc. 12th International Conference on VLSI Design, Goa, India, 1999, pp. 408-413.

[8] H.R. Andersen, H. Hulgaard “Boolean Expression Diagrams,” in *Proc. 12th IEEE Symposium on Logic in Computer Science*, Warsaw, Poland, June 29-July 2, 1997, pp. 88-98.

[9] R. Ubar, “Test Synthesis with Alternative Graphs,” IEEE D&T of Comp. Spring 1996, pp. 48-59.

[10] A. Jutman, J. Raik, R. Ubar, “SSBDDs: Advantageous Model and Efficient Algorithms for Digital Circuit Modeling, Simulation & Test,” in Proc. of 5th Int. Workshop on Boolean Problems, Freiberg, Germany, Sept. 19-20, 2002, pp. 157-166.

[11] A. Jutman, R. Ubar, “Design Error Diagnosis in Digital Circuits with Stuck-at Fault Model,” Journal of Microelectronics Reliability. Pergamon Press, Vol. 40, No 2, 2000, pp. 307-320.

[12] R. Ubar, “Multi-Valued Simulation of Digital Circuits with Structurally Synthesized Binary Decision Diagrams,” OPA, Gordon and Breach Publishers, Multiple Valued Logic, 1998, Vol.4, pp. 141-157.

[13] R. Ubar, A. Jutman, Z. Peng, “Timing Simulation of Digital Circuits with Binary Decision Diagrams”, in Proc. of DATE 2001 Conference, München, Germany, 2001, pp. 460-466.

[14] R. Ubar, “Parallel Critical Path Tracing Fault Simulation,” in Proc. of the 39. Int. Wiss. Kolloquium, Ilmenau, Germany, 1994, Band 1, pp. 399-404.