# On the Construction of Small Fully Testable Circuits with Low Depth

Görschwin Fey[1] Anna Bernasconi[2] Valentina Ciriani[3] Rolf Drechsler[1]

[1] Inst. of Computer Science
University of Bremen
28359 Bremen, Germany
{fey,drechsle}@informatik.uni-bremen.de

[2] Dep. of Computer Science
University of Pisa
56127 Pisa, Italy
annab@di.unipi.it

[3] Dep. of Information Technology
University of Milano
26013 Crema, Italy
ciriani@dti.unimi.it

## Abstract

*During synthesis of circuits for Boolean functions area, delay and testability are optimization goals that often contradict each other. Multi-level circuits are often quite small while circuits with low depth are often larger regarding the area requirements. A different optimization goal is good testability which can usually only be achieved by additional hardware overhead.*

*In this paper we propose a synthesis technique that allows to trade-off between area and delay. Moreover, the resulting circuits are 100% testable under the stuck-at fault model. The proposed approach relies on the combination of 100% testable circuits derived from binary decision diagrams and 2-SPP networks. Full testability under the stuck-at fault model is proven and experimental results show the trade-off between area and depth.*

## 1 Introduction

The synthesis of circuits is subject to many often contradicting objectives. Area and delay are supposed to be as small as possible while the testability of the circuit should be very good. While in the past testability was a secondary optimization goal only, today the importance of highly testable circuit structures has significantly grown. This is due to the shrinking feature size that makes structures more susceptible for functional failures introduced during the production process. Such functional flaws can often be detected by applying a functional fault model at the Boolean level and generating test patterns with respect to the fault model. Therefore good testability becomes another goal during synthesis. But often good testability can only be achieved at the cost of additional hardware. In this paper testability with respect to the *Stuck At Fault Model* (SAFM) is considered.

Several restrictions to the structure of a circuit have been considered to meet certain goals during synthesis (see e.g. [16, 11, 9, 12, 10, 6, 14, 13]). Promising approaches that already consider testability are circuits based on *Binary Decision Diagrams* (BDDs), so called BDD circuits [1], and *Sum of Pseudoproduct* (SPP) networks [6].

A BDD circuit is created by directly mapping the BDD of the Boolean function onto a network of multiplexers. As a result the depth of the circuit is quite large; in fact the depth is equal to the number of primary inputs. But as an advantage the circuits are quite compact for a large range of functions. At the cost of one additional input and one inverter any BDD circuit can be transformed into a circuit without untestable stuck-at faults [11].

SPP networks are 100% testable by construction if certain restrictions are met [8]. Due to implementation issues often 2-SPP networks are considered that restrict the number of input to XOR gates to two [6, 7]. As an advantage over BDD circuits, SPP networks have a depth of only 3 levels of logic gates. But the final circuit is often larger due to the tight restrictions on the network structure.

In this paper, an approach to combine both synthesis techniques is proposed. The resulting circuits are proven to be fully testable under the SAFM. Experimental results show that the new synthesis technique allows to trade-off size and depth of the final circuit.

The paper is structured as follows. In Section 2 the previous synthesis approaches are briefly reviewed. The new approach to combine both methods is presented in Section 3. Full testability of the resulting circuits under the SAFM is proven in Section 4. Experimental results are reported in Section 5 and, finally, the paper is summarized in Section 6.

## 2 Preliminaries

In this section, the *Stuck-At Fault Model* (SAFM) as well as BDD circuits and SPP networks are briefly reviewed.

### 2.1 Stuck-At Fault Model

Let $C$ be any combinational logic circuit, a fault in the SAFM [4] fixes exactly one input or output pin of a node in $C$ to the constant value 0 or 1, independently of the values applied to the primary inputs of the circuit. The construction of complete test sets requires the determination of the faults which are not testable, i.e. *redundant faults*. A node $v$ in $C$ is called *fully testable*, if there does not exist a redundant fault with fault location $v$. If all nodes in $C$ are fully testable, then $C$ is *fully testable*. Redundancies may also invalidate tests for testable faults and often correspond to locations of the circuit where area is wasted [4]. For this reason, synthesis procedures which result in non-redundant circuits are desirable.

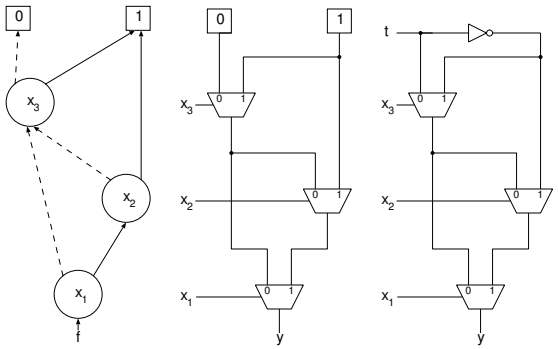
### 2.2 BDD-Circuits

A Boolean function $f : \mathbb{B}^n \rightarrow \mathbb{B}$ can be represented as a BDD [5]. A BDD is an acyclic graph where each node is either a leaf or an internal node. Each internal node $v$ has the two successors $low(v)$ and $high(v)$ and is labeled by a variable $x_i$. An internal node $v$ represents the Boolean function, i.e. the Shannon decomposition carried out at $v$:

$$f = \overline{x}_i f_{low(v)} + x_i f_{high(v)}$$

The leaves are labeled by 0 and 1 and represent the corresponding constant Boolean functions. A BDD is called *ordered* when the variables occur in the same order and at most once on all paths from an internal node to the leaves. The BDD is called *reduced* if it does not contain isomorphic subgraphs. A reduced ordered BDD is a canonical representation of Boolean functions that allows for efficient manipulation [5]. In the following the term BDD refers to reduced ordered BDDs. A simple BDD for the function $f = x_1x_2 + x_3$ is shown in Figure 1(a). Note that in contrast to the usual presentation, the BDD is shown upside down to illustrate the correspondence to the BDD circuit.

A BDD circuit can be derived by replacing each BDD node by a multiplexor as illustrated in Figure 1(b). As shown in [1], a BDD circuit is fully testable under the SAFM, if and only if both combinations 10 and 01 can be applied to the inputs of each multiplexer. It is possible to create fully testable BDD circuits by adding one additional input and one inverter. This was shown in [11] and has been implemented as the tool MuTaTe. Figure 1(c) illustrates this transformation.




(a) BDD graph (b) BDD circuit (c) Fully testable BDD circuit

**Figure 1. Creating a 100% testable BDD circuit**

### 2.3 SPP Networks

A *pseudoproduct* is a product (AND) of Exclusive OR (EXOR) factors, and an *SPP form* is a sum (OR) of pseudoproducts. Any arbitrary Boolean function can be expressed as a disjunction of pseudoproducts; this leads to a sum of pseudoproducts form. For example $(x_1 \oplus \overline{x}_2) \cdot x_5 \cdot (x_1 \oplus x_4 \oplus \overline{x}_7) + x_5 \cdot \overline{x}_4 + (x_1 \oplus x_3 \oplus x_4) \cdot (x_3 \oplus x_5 \oplus x_6)$ is an SPP expression.

*Sum of Pseudoproduct* (SPP) forms allow to represent Boolean functions with much shorter expressions than standard *Sum of Products* (SOP) forms.

Although SPP forms are compact, they have been defined for EXOR gates with unbounded fan-in that seem to be impractical in the current technology [18]. It follows that SPP-like forms with a fixed maximum number of literals in the EXOR factors are much more interesting. Therefore *k-SPP forms*, where the number of literals in the EXOR factors is restricted to an upper bound by a chosen constant $k$, have been introduced in [6, 7]. Experimental results show that the $k$-SPP synthesis algorithm generates still compact networks in a short time, even for $k = 2$. Therefore we focus on 2-SPP forms, that are still much more compact than SOP forms.

More formally, in a Boolean space $\{0, 1\}^n$ described by $n$ variables $x_1$, $x_2$, ..., $x_n$, a 2-*EXOR factor* is an EXOR with at most two variables, one of which possibly complemented (an EXOR with just one literal

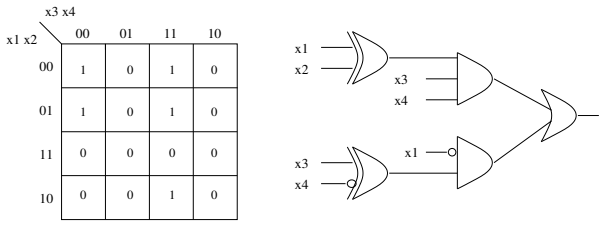



**Figure 2.** Karnaugh map of function $f$ with a 2-SPP cover $(x_1 \oplus x_2)x_3x_4 + \overline{x}_1(x_3 \oplus \overline{x}_4)$, minimal with respect to the number of 2-pseudoproducts, and the corresponding 2-SPP circuit representation.

corresponds to the literal itself). Given two Boolean variables $x_1, x_2$, all the possible 2-EXOR factors are essentially $x_1$, $\overline{x}_1$, $x_2$, $\overline{x}_2$, $(x_1 \oplus x_2)$ and $(x_1 \oplus \overline{x}_2)$ (in fact, $\overline{x}_1 \oplus x_2 = x_1 \oplus \overline{x}_2$, and $\overline{x}_1 \oplus \overline{x}_2 = x_1 \oplus x_2$).

**Definition 1.** *A* 2-pseudoproduct *is a product of* 2-*EXOR factors; and a* 2-SPP form *is a sum of* 2-*pseudoproducts.*

**Definition 2.** *A set of points whose characteristic function can be represented as a* 2*-pseudoproduct is a* 2-pseudocube.

This is a generalization of the concept of cubes. An SOP form is a particular 2-SPP form where each EXOR factor contains only one literal.

In the space $\{0,1\}^n$ the number of different 2-EXOR factors with exactly 2 literals is $2 \cdot \binom{n}{2} = n(n-1)$. Thus in the worst case, 2-SPP forms require a quadratic number of different 2-EXOR gates.

The 2-SPP synthesis problem can be stated as: *given a set of points in the Boolean space* $\{0,1\}^n$*, find its minimal cover composed of* 2*-pseudocubes*, where a minimal cover is represented by a sum of 2-pseudoproducts with a minimal number of literals or with a minimal number of 2-pseudoproducts.

**Example 1.** *For the function f represented by the Karnaugh map in Figure 2, the following 2-SPP cover is an expression minimal with respect to the number of* 2*-pseudoproducts:* $(x_1 \oplus x_2)x_3x_4 + \overline{x}_1(x_3 \oplus \overline{x}_4)$*. The 2-SPP circuit representation is shown on the right side of the figure. On the other hand, a 2-SPP form minimal with respect to the number of literals is* $\overline{x}_2x_3x_4 + \overline{x}_1(x_3 \oplus \overline{x}_4)$*. Finally, a minimal SOP form of such function is* $\overline{x}_2x_3x_4 + \overline{x}_1\overline{x}_3\overline{x}_4 + \overline{x}_1x_3x_4$*.*

In [7] a 2-SPP minimization algorithm is proposed. As in the Quine-McCluskey approach, the generation of prime 2-pseudoproducts is performed in steps of successive unions of 2-pseudoproducts. A minimal 2-SPP form is generated by choosing a minimal subset of prime 2-pseudoproducts that covers the original function.

A heuristic minimization procedure for 2-SPP forms based on the iteration of a suite of operations that generalize the expansion-irredundant-reduction cycle of heuristic SOP minimization is presented in [2]. The proposed procedure has been implemented with good results on industrial benchmarks, enabling us to minimize 2-SPP forms for which we cannot afford to compute an exact solution.

Beside synthesis, testability is a major aspect of the design process. In [8] it is shown that a 2-SPP network *minimal* with respect to the number of literals is fully testable under the SAFM. The proof of full testability presented in [8] exploits the properties of a minimal network, i.e. primality, irredundancy and minimality with respect to the number of literals. In [2] it has been shown that primality and minimality are not necessary for guaranteeing full testability. Indeed weaker properties are sufficient for obtaining fully testable 2-SPP networks. Thus the 2-SPP networks (possibly not optimal) derived with the heuristic presented in [2] are still fully testable under the SAFM.

Since the heuristic method generates 2-SPP forms that are compact and fully testable, we use this algorithm for synthesizing the 2-SPP part of the MuTaTe-SPP networks that are introduced in the next section.

## 3 Combining BDD Circuits and 2-SPP Networks

As an advantage 2-SPP networks and BDD circuits that are created according to MuTaTe are fully testable under the SAFM. The drawback of BDD circuits is the large depth whereas 2-SPP networks may have a large number of gates. By combining both techniques these problems can be mitigated while retaining full testability under the SAFM. An approach to combine both synthesis methods to create MuTaTe-SPP networks is introduced in this section.

To achieve full testability of a 2-SPP network, the primary inputs of the network must be controllable. The BDD network is testable according to functional arguments on the BDD. MuTaTe-SPP networks consist of an upper part that resembles the structure of the BDD circuit and a lower part that consists of a 2-SPP network.

Figures 3 and 4 illustrate how to derive a MuTaTe-SPP network from a given Boolean function $f$:

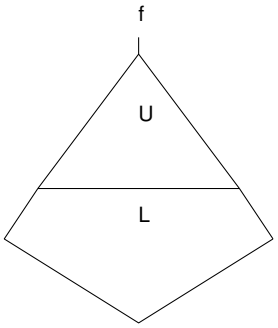



**Figure 3. Separating the BDD**

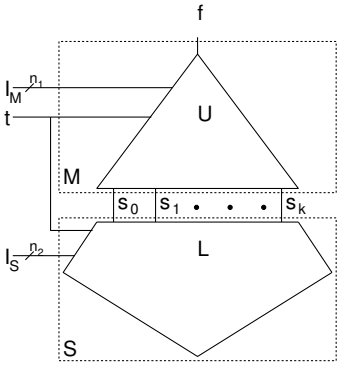



**Figure 4. Creating the circuit**

1. Create the BDD of the function.
2. Separate the BDD into an upper and a lower part.
3. Nodes in the upper part are collected in the set $U$, nodes in the lower part are collected in the set $L$ (see Figure 3).
4. Map the BDD onto a circuit according to MuTaTe.
5. Assemble all gates that correspond
   - to nodes in $U$ in a subcircuit $M$ and
   - to nodes in $L$ in a subcircuit $S$

   (the dotted boxes in Figure 4 show the subcircuits).
6. Resynthesize the subcircuit $S$ as a 2-SPP network.

Besides the test input $t$, a subset of the primary inputs is also passed to the subcircuit $S$. The outputs of the subcircuit $S$ are denoted by $s_0, \ldots, s_k$. The Boolean functions represented by these outputs are identical to the functions previously represented by nodes in the BDD.

How to separate the BDD into two parts to retrieve a small circuit that is not too deep is subject to heuristics. Here, a first approach is proposed. The BDD is minimized by standard reordering techniques, i.e. sifting [15], and a certain depth for the final circuit is chosen. By this, the number of levels of BDD variables that can be assembled in the upper part of the circuit to reduce the size of the 2-SPP network is fixed.

The experimental results show that trading area for delay, i.e. the depth of the circuit, becomes possible by using the new approach. Moreover, by construction these circuits are fully testable with respect to the SAFM as will be proven in the next section.

## 4 Testability

Before proving the full testability of the composed MuTaTe-SPP networks, we briefly recall the already mentioned testability results concerning BDD and 2-SPP circuits, with respect to the SAFM.

A BDD circuit is fully testable if and only if both combinations 10 and 01 can be applied to the inputs of each multiplexer cell [1]. Since at least one of the two combinations is always applicable, as a simple computation shows, 100% testability of a BDD circuit can be achieved by adding one additional input and one inverter as shown in [11] and has been implemented as the tool MuTaTe.

On the other hand, any 2-SPP circuit minimal with respect to the number of literals is fully testable under the SAFM [8]. Fortunately, as shown in [2], the minimality of the circuit is not necessary and indeed weaker properties are sufficient for obtaining fully testable 2-SPP networks. More precisely, a 2-SPP network is fully testable in the SAFM if and only if the corresponding algebraic expression is *(i) AND-irredundant*, i.e., the deletion of a factor in any 2-pseudoproduct changes the represented function; *(ii) OR-irredundant*, i.e., the deletion of any 2-pseudoproduct changes the represented function; and *(iii) EXOR-irredundant*, i.e., the deletion of any literal in a 2-EXOR factor changes the represented function. This result is of importance because the three irredundancy properties can be guaranteed even when the synthesis is performed applying heuristic algorithms, instead of exact minimization procedures. In particular the tool developed in [2], and applied in this paper to obtain MuTaTe-SPP networks, has been designed in order to yield, in reasonable computational time, *AND-OR-EXOR-irredundant*[1] 2-SPP expressions resulting in fully testable, although not minimal, networks.

Combining these testability results, we can prove:

**Theorem 1.** *MuTaTe-SPP networks are fully testable with respect to the SAFM.*

[1] A 2-SPP circuit that is AND-irredundant, OR-irredundant, and EXOR-irredundant is called *AND-OR-EXOR-irredundant*.

*Proof.* Let $C$ be a MuTaTe-SPP network computing a Boolean function $f$. As shown in Figure 4, $C$ is composed of a 2-SPP subcircuit $S$, with outputs $s_i$, $0 \leq i \leq k$, and a MuTaTe-subcircuit $M$, with primary output $f$. To prove the full testability of $C$ we must consider two cases.

1. **A fault occurs in the 2-SPP subcircuit $S$.**

   Since any AND-OR-EXOR-irredundant 2-SPP network is fully testable in the SAFM, and since all inputs of $S$ are primary inputs, any fault occurring in $S$ can be propagated to at least one output $s_i$ of the subcircuit. Now observe that by applying an appropriate combination of values to the input variables of the MuTaTe subcircuit $M$, any output of $S$ can be propagated through $M$, otherwise a SAFM on the corresponding signal line in the BDD circuit according to MuTaTe would not be testable either, in contradiction to the results from [11]. Thus the fault observed in one of the outputs of $S$ can be propagated to the primary output of $C$ and tested.

   Therefore, all faults in $S$ are testable.

2. **A fault occurs in the MuTaTe subcircuit $M$.**

   Now, suppose that a fault occurs in the subcircuit $M$ of the MuTaTe-SPP network $C$. All signal lines in $M$ directly correspond to lines in the intermediate BDD circuit that was constructed in step (4) of the algorithm in Section 3. This intermediate circuit is fully testable by construction and this means that both combinations 10 and 01 can be applied to the inputs of each of its multiplexer cells.

   Now observe that the outputs $s_0, \ldots, s_k$ of $S$ correspond to lines in the same intermediate fully testable BDD circuit as well. Therefore all values needed to get both combinations 10 and 01 at the inputs of each multiplexer in $M$ can be propagated along them, before the execution of steps (5) and (6) of the algorithm.

   But this property holds even after the execution of these two final steps. In fact, when $S$ is resynthesized as a 2-SPP network, the Boolean function computed at any of its outputs does not change. Therefore the propagation properties of the BDD structure are retained and the combinations 01 and 10 are still applicable to the inputs of any multiplexer in $M$.

   Thus, all faults in $M$ are testable.

□

## 5 Experimental Results

The experimental data provided in this section shows that the trade-off between area and delay can be controlled using MuTaTe-SPP networks. Moreover, a comparison to BDD circuits and 2-SPP networks is presented. All experiments were carried out on an AMD Athlon 64 3500+ (Linux, 1GB RAM). The benchmark circuits considered are taken from the LGSynth93 benchmarks suite, additionally symmetric Boolean functions are considered, e.g. *s10t45* has 10 inputs plus 1 test input for MuTaTe and evaluates to one if 4 or 5 inputs assume the value 1.

Experimental data to evaluate parameter settings for MuTaTe-SPP networks is shown in Table 1. For each circuit the number of inputs (*IN*) and outputs (*OUT*) is reported. Data for 2, 3 and 4 input variables that are considered in the BDD-part of the MuTaTe-SPP network is reported in columns *2 variables*, *3 variables* and *4 variables*, respectively. The variables are taken from the topmost levels of the BDD that initially represents the function of the circuit. For each configuration the number of gates (*NODES*), literals (*LITS*) and the depth of the circuit (*DEPTH*) as determined by SIS [17] are reported. The number of gates counts complex gates like multiplexers as a single gate.

The depth of a circuit can be calculated in a straightforward manner. If two variables of the BDD part are joined into a subcircuit $M$, the depth of $M$ is 2 or 3, since an additional inverter at an output may be necessary due to complement edges [3]. The subcircuit $S$ is composed of the EXOR factors and an SOP-node and, therefore, has a depth of 2. Thus, the depth of the circuit is 4 or 5. The same argumentation applies when joining 3 or 4 variables of the BDD into subcircuit $M$.

The number of variables considered in $M$ influences the size of the final circuits. In case of the LGSynth93 benchmarks, 7 out of the 13 circuits considered are most compact when 4 variables are joined into $M$. Trading area for depth works well for e.g., *9sym*, *ex1010* and *inc*. In general, the heuristic for partitioning the inputs between the BDD part and the SPP part has to be improved. For example often different a different splitting should be used for different outputs. Some outputs may not even depend on the inputs considered in the BDD part and therefore no reduction in size is achieved.

On the symmetric circuits that have a very compact representation as a BDD, including more variables in the BDD part of the circuit always yields a reduction in size. Thus, trading area for depth becomes possible.

Results in comparison to pure BDD circuits according to MuTaTe and plain SPP networks are considered next. The depth of the circuits is always equal to the number of inputs plus one for the MuTaTe circuit and two for the SPP network. The depth for circuits created using the new approach was considered above. Thus, the depth of MuTaTe-SPP networks is always less or equal to that of the MuTaTe circuit and larger than that of the SPP network. Figure 5 shows the number of literals for the different approaches for selected circuits. The lower the number of literals the smaller circuit. In all cases the circuit created by MuTaTe is smaller than the SPP network. Moreover in most cases, at least one of the combined MuTaTe-SPP circuits has a size that is between those of the MuTaTe circuit and the SPP network. For example, *ex1010* and *s10t45* ideally exhibit the expected behavior: a small BDD circuit, a large SPP network and with increasing number of variables in the BDD-part of the MuTaTe-SPP network the size of the circuit shrinks again.

In summary, the new approach can be applied to construct circuits that have a predefined depth by spending area to transform the initial BDD circuit into a MuTaTe-SPP network. The circuits are provably fully testable under the SAFM.

## 6 Conclusions

A new synthesis approach to combine SPP networks and BDD circuits has been proposed. By adjusting the number of variables in the BDD part of the circuit, area can be traded for depth. Complete testability with respect to the SAFM is achieved by construction.

The evaluation of better heuristics to separate the BDD part of the circuit from the SPP part based on the initial BDD representation remain future work. One promising approach is to take into account which primary inputs are in the support of which output. Also different ordering heuristics for the BDD that allow for a compact representation of the lower part as an SPP network are of interest. Here, using a weighted approach that counts the density of the ON-set in the lower part may be of interest.

## References


[1] B. Becker. Synthesis for testability: Binary decision diagrams. In *Symp. on Theoretical Aspects of Comp. Science*, volume 577 of *LNCS*, pages 501–512. Springer Verlag, 1992.

[2] A. Bernasconi, V. Ciriani, R. Drechsler, and T. Villa. Efficient Minimization of Fully Testable 2-SPP Networks. In *Design Automation and Test in Europe (DATE)*, pages 1300–1305, 2006.

[3] K. Brace, R. Rudell, and R. Bryant. Efficient implementation of a BDD package. In *Design Automation Conf.*, pages 40–45, 1990.

[4] M. Breuer and A. Friedman. *Diagnosis & reliable design of digital systems*. Computer Science Press, 1976.

[5] R. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Trans. on Comp.*, 35(8):677–691, 1986.

[6] V. Ciriani. Synthesis of SPP Three-Level Logic Networks using Affine Spaces. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(10):1310–1323, 2003.

[7] V. Ciriani and A. Bernasconi. 2-SPP: a Practical Trade-Off between SP and SPP Synthesis. In *5th International Workshop on Boolean Problems (IWSBP2002)*, pages 133–140, 2002.

[8] V. Ciriani, A. Bernasconi, and R. Drechsler. Testability of SPP three-level logic networks in static fault models. *IEEE Trans. on CAD*, 25(10):2241–2248, 2006.

[9] D. Debnath and T. Sasao. Multiple–Valued Minimization to Optimize PLAs with Output EXOR Gates. In *IEEE International Symposium on Multiple-Valued Logic*, pages 99–104, 1999.

[10] D. Debnath and Z. Vranesic. A Fast Algorithm for OR-AND-OR Synthesis. *IEEE Transactions on CAD*, 22(9):1166–1176, 2003.

[11] R. Drechsler, J. Shi, and G. Fey. Synthesis of fully testable circuits from BDDs. *IEEE Trans. on CAD*, 23(3):440–443, 2004.

[12] E. Dubrova, D. Miller, and J. Muzio. AOXMIN-MV: A Heuristic Algorithm for AND-OR-XOR Minimization. In *4th Int. Workshop on the Applications of the Reed Muller Expansion in circuit Design*, pages 37–54, 1999.

[13] R. Ishikawa, T. Hirayama, G. Koda, and K. Shimizu. New Three-Level Boolean Expression Based on EXOR Gates. *IEICE Transactions on Information and Systems*, (5):1214–1222, 2004.

[14] F. Luccio and L. Pagli. On a New Boolean Function with Applications. *IEEE Transactions on Computers*, 48(3):296–310, 1999.

[15] R. Rudell. Dynamic variable ordering for ordered binary decision diagrams. In *Int'l Conf. on CAD*, pages 42–47, 1993.

[16] R. Rudell and A. Sangiovanni-Vincentelli. Multiple-valued Minimization for PLA Optimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, CAD-6:727–750, Sept. 1987.

[17] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli. SIS: A system for sequential circuit synthesis. Technical report, University of Berkeley, 1992.

[18] N. Weste and K. Eshraghian. *Principles of CMOS VLSI Design*. Addison-Wesley Publishing Company, 1993.

**Table 1. Different parameters for the number of variables considered in $U$**

| circuit | | | 2 variables | | | 3 variables | | | 4 variables | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| name | IN | OUT | NODES | LITS | DEPTH | NODES | LITS | DEPTH | NODES | LITS | DEPTH |
| 9sym | 10 | 1 | 22 | 558 | 5 | 28 | 344 | 6 | 20 | 173 | 7 |
| 9symml | 10 | 1 | 23 | 555 | 5 | 24 | 315 | 6 | 20 | 173 | 7 |
| clip | 10 | 5 | 60 | 1221 | 5 | 82 | 1211 | 6 | 127 | 1326 | 7 |
| cm150a | 22 | 1 | 7 | 238 | 4 | 9 | 304 | 5 | 15 | 200 | 6 |
| cm152a | 12 | 1 | 12 | 72 | 4 | 17 | 68 | 5 | 25 | 76 | 6 |
| cm85a | 12 | 3 | 24 | 190 | 4 | 27 | 185 | 5 | 32 | 237 | 6 |
| ex1010 | 11 | 10 | 110 | 12066 | 5 | 182 | 11021 | 6 | 335 | 10329 | 7 |
| inc | 8 | 9 | 69 | 691 | 5 | 106 | 604 | 6 | 122 | 512 | 7 |
| majority | 6 | 1 | 8 | 36 | 4 | 10 | 37 | 5 | 11 | 38 | 6 |
| misex3 | 15 | 14 | 181 | 15961 | 5 | 268 | 17024 | 6 | 438 | 20070 | 7 |
| misex3c | 15 | 14 | 168 | 17220 | 5 | 264 | 18986 | 6 | 435 | 16609 | 7 |
| mux | 22 | 1 | 7 | 249 | 4 | 12 | 186 | 5 | 14 | 208 | 6 |
| table3 | 15 | 14 | 177 | 12908 | 5 | 236 | 14484 | 6 | 361 | 13887 | 7 |
| s10t45 | 11 | 1 | 41 | 1281 | 5 | 38 | 868 | 6 | 36 | 606 | 7 |
| s10t56 | 11 | 1 | 42 | 1444 | 5 | 38 | 948 | 6 | 35 | 584 | 7 |
| s11t3-7 | 12 | 1 | 49 | 1654 | 5 | 32 | 1056 | 6 | 37 | 562 | 7 |
| s11t34 | 12 | 1 | 50 | 1453 | 5 | 45 | 1048 | 6 | 43 | 694 | 7 |
| s11t45 | 12 | 1 | 51 | 2065 | 5 | 46 | 1501 | 6 | 43 | 941 | 7 |
| s11t56 | 12 | 1 | 51 | 2467 | 5 | 46 | 1725 | 6 | 43 | 1157 | 7 |
| s11t67 | 12 | 1 | 50 | 2499 | 5 | 46 | 1788 | 6 | 43 | 1047 | 7 |
| s12t4-8 | 13 | 1 | 59 | 3502 | 5 | 53 | 2650 | 6 | 49 | 1320 | 7 |
| s12t78 | 13 | 1 | 61 | 4423 | 5 | 54 | 3001 | 6 | 51 | 1896 | 7 |
| s7t23 | 8 | 1 | 19 | 177 | 5 | 17 | 126 | 6 | 18 | 77 | 7 |
| s8t34 | 9 | 1 | 26 | 393 | 5 | 21 | 243 | 6 | 22 | 159 | 7 |
| s9t56 | 10 | 1 | 34 | 811 | 5 | 30 | 497 | 6 | 28 | 279 | 7 |

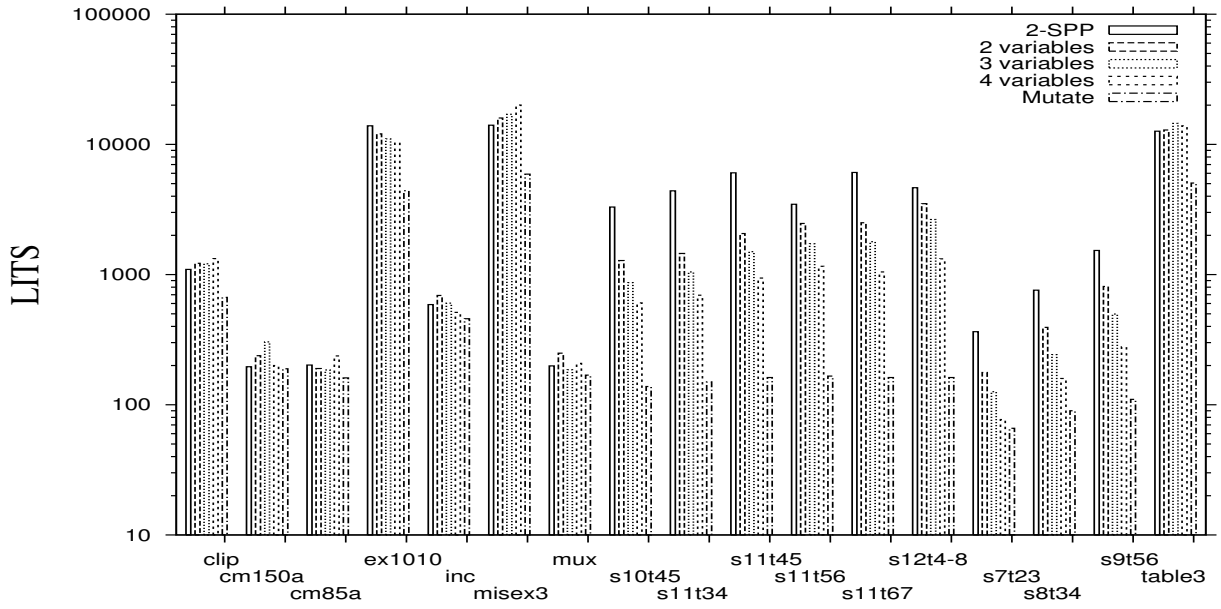



**Figure 5. Comparison to MuTaTe and 2-SPP networks**