False-Noise Analysis using Resolution Method

Alexey Glebov, Sergey Gavrilov, David Blaauw*, Vladimir Zolotov**, Rajendran Panda**, Chanhee Oh**

MicroStyle - Moscow, Russia, *University of Michigan, Ann Arbor, MI, **Motorola Inc. Austin, TX,

Abstract

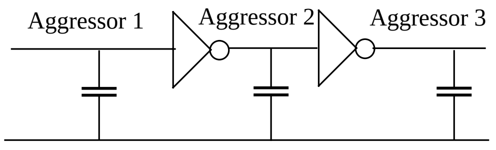
High-performance digital circuits are facing increasingly severe noise problems due to cross-coupled noise injection. Traditionally, noise analysis tools use the conservative assumption that all neighbors of a net can switch simultaneously, thereby producing the worst-case noise on a net. However, due to the logic correlations in the circuit, this worst-case noise may not be realizable, resulting in a so-called false noise failure. Since the problem has been shown to be NP-hard in general [2], exact solutions to this problem are not possible. In this paper, we therefore propose a new heuristic to eliminate false noise failures based on the resolution method [16]. It is shown that multi-variable logic relations can be computed directly from a transistor level description. Based on these generated logic relations, a characteristic ROBDD for a signal net and its neighboring nets is constructed. This ROBDD is then used to determine the set of neighboring nets that result in the maximum realizable noise on the net. The proposed approach was implemented and tested on industrial circuits. The results demonstrate the effectiveness of the approach to eliminate false noise failures.

1 Introduction

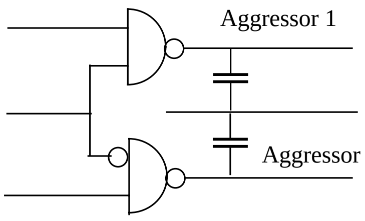
As process technology has advanced to deep submicron dimensions, noise in digital circuits has become a major concern. Noise can occur in a circuit through a number of different mechanisms - the most prominent of which is cross-coupling capacitance. Cross-coupling noise has become particularly critical as wire aspect ratios have increased, leading to tall and narrow wires that are closely spaced together. In noise analysis, the net on which noise is injected is referred to as the *victim* net, while the neighboring net that injects the noise is referred to as the *aggressor* net. Injected noise can be classified into two categories. If the victim net is not switching at the time of the noise injection, a noise pulse will result on the victim net which can propagate to a latch and change the state of the circuit. This noise type is referred to as a *functional noise*. On the other hand, if the victim net transitions at the time of noise injection, the delay of the victim transition is altered. This type of noise is referred to as *delay noise*.

Noise analysis tools typically make the assumption that all aggressor nets switch at the same time and in the same direction [1], [11]. Under this assumption, the noise injected from each aggressor combines, creating the maximum possible composite noise pulse on the victim net and yielding a conservative analysis. In practice, however, the timing and logic constraints present in the circuit may prevent all aggressors from switching in the same direction at the worst possible alignment time. Therefore, the noise reported by an analysis that does not account for timing and logic correlations can severely overestimate the actual noise realizable on a victim net and can create a so-called *false noise violation*. This is especially important when the number of aggressors for a victim is high (e.g. 10 or more), as is often the case.

Industrial noise analysis approaches have exploited timing correlations in circuits to reduce the pessimism of noise analysis by identifying situations where aggressor nets cannot switch at the same time (Figure 1(a)). To determine when a net can switch, the so-called *switching windows* are propagated in the circuit using static timing analysis [1], [2], [11]. After switching windows are identified for each aggressor, the possibility of overlap between timing windows for a set of aggressors is determined. However, this approach does not identify situations where a pair of aggressor nets, that can each switch individually at a particular time point, cannot *both* switch at that time due to logic relationships in the circuit. A simple example of such a situation is shown in Figure 1(b). Also the timing window based approach does not identify cases where nets cannot switch in the same direction, for instance when they are connected by an inverter as shown in Figure 1(a). Therefore, timing correlations will not remove all false noise failures, although it has been shown in practice to be relatively effective [1].



a) timing correlations between aggressors



b) logic correlations between aggressors

Figure 1. Logic relationships between aggressors

In order to identify all false noise failures, both timing and logic correlations of the circuit must be taken into account. In [2], it was shown that in general, this problem can be represented as a search for a worst-case 2-vector test using a Boolean Constraint Optimization formulation. In [3], a method based on compatible observability don't care sets was proposed. In [12], a method is proposed using a test pattern generation approach. However, all these methods have high computational complexity and cannot be applied to large problem sizes. Since noise primarily occurs in chip-level routes, it is critical to perform false noise analysis at this level in large designs, and hence heuristic methods must be employed.

In [13], an approach for false noise analysis was developed, based on so-called simple logic implications (SLI) [4]. An SLI expresses a logic relationship between two signals. Logic implications have been widely used in logic synthesis [5-8] as well as in peak current estimation [9]. In [13], pairwise SLIs are generated from ROBDD representations of the DC-connected components (DCCC) in the circuit. The generated pairwise implications are propagated in the circuit through forward and backward topological traversals. After computing SLIs, a constrained graph representation of the switching aggressors is build. It is then shown that the aggressor subset resulting in the maximal noise can be obtained by solving the maximum weighted independent set problem for the constraint graph.

The main limitation of an SLI based approach is that only pairwise implications are considered. In a circuit, many relations involving three or more circuit nodes exist, which can not be captured by SLIs. In this paper, we therefore present a new approach for false noise analysis that considers logic constraints between multiple nodes. Our approach is based on the well known resolution method [16] which has been widely used in mechanical theorem proving. In contrast to the implication based approach, the proposed method does not require extraction of logic descriptions for transistor level circuits, but can operate directly on transistor level circuits. This is particularly useful for circuits with large and complex DCCCs that are difficult or impossible to represent with their logic functions.

The proposed method use a zero-delay assumption which is valid only during the stationary state of the circuit before and after all transitions occur. Hence, this formulation for false noise analysis is conservative only for glitch-free circuits, obtained, for instance, through special transistor sizing methods [10]. The proposed approach was implemented and results are presented for a number of industrial test cases. It is shown that the total number of noise failures is reduced by up to 47%, demonstrating the effectiveness of the approach.

The remainder of this paper is organized as follows: Section 2 explains logic constraints and their application to false noise analysis. Section 3 explains the resolution method in connection with boolean relations. Sections 4 describes the logic constraints generation algorithm. Section 5 presents false noise analysis approach using logic constraints. Section 6 presents results, and Section 7 presents our concluding remarks.

2 Logic constraints and false noise analysis

Any circuit has many logic correlations between its signals. For noise analysis these correlations can be considered as logic constraints prohibiting circuit nets to have some combinations of signals. For false noise analysis it is especially important to find that a group of aggressor nets are prohibited from having simultaneous rising or falling transition if the victim net is at the given voltage level. Aggressor nets $(a_1,a_2,...,a_n)$ can not switch simultaneously in the same direction if one of the two signal combinations $(a_1=1,a_2=1,...,a_n=1)$ or $(a_1=0,a_2=0,..,a_n=0)$ is prohibited at the condition that the victim net is at the given state.

2.1 Representation of logic correlations

Logic correlations between circuit signals $(s_1,s_2,..., s_n)$ can be represented with a system of logic equations:

$$\begin{aligned} f_1(s_1, s_2, \ldots, s_n) &= c_1 \\ f_2(s_1, s_2, \ldots, s_n) &= c_2 \\ &\ldots \\ f_m(s_1, s_2, \ldots, s_n) &= c_m \end{aligned} \qquad \text{(EQ 1)}$$

where f_i are arbitrary functions and c_i are *0* or *1*.

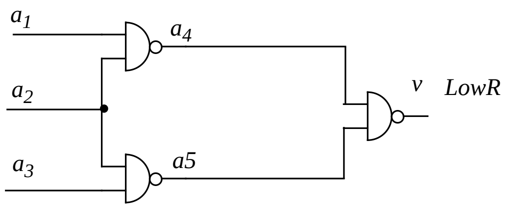
There are two functions convenient for expressing logic constraints: disjunction and conjunction. In the first case each constraint is $a_1+a_2+...+a_n=1$ and in the second case $a_1*a_2*...*a_n=0$ where each a_i is either s_i or $\bar{s}_i$. The first form is more common for resolution method used for theorem proving [16].For noise analysis the second form is more convenient. Each equation in the second form prohibits the signal combination where signals without negation are equal to *1* and signals with negation are equal to *0*. The system of constraint equations can be rewritten as a single equation:

$$\sum_{i=1}^{m} f_i = 0 \qquad \text{(EQ 2)}$$

Here each term f_i corresponds to one equation in (EQ1) and has the following form:

$$s^{\alpha_1}{}_{i_1} \cdot s^{\alpha_2}{}_{i_2} \cdot \ldots \cdot s^{\alpha_k}{}_{i_k} \qquad \text{(EQ 3)}$$

where each $s^{\alpha}{}_j$ means s_j if $\alpha = 0$ or $\bar{s}_j$ if $\alpha = 1$. For simple gates the logic constraints equation written in this way coincides with its characteristic equation in disjunctive form. For example logic constraints for 2-input AND gate with logic function $x=a*b$ can be written as $\bar{x}*a*b+x*\bar{a}+x*\bar{b}=0$ that is exactly its characteristic function. Here term $\bar{x}*a*b$ prohibits the combination *(x=0, a=1, b=1)*. Instead of writing a full equation in disjunctive form



Logic constraint useful for analyzing *LowR* noise at net *v*

$\bar{v}*\bar{a}_4$, $\bar{v}*\bar{a}_5$,

$\bar{a}_1*\bar{a}_4$, $\bar{a}_2*\bar{a}_4$, $\bar{a}_2*\bar{a}_5$, $\bar{a}_3*\bar{a}_5$

$a_1*a_2*a_4$, $a_2*a_3*a_5$

Figure 2. Example of circuit with logic constraints

we will simply specify the set of its terms. In Figure 2 we show an example of a simple circuit and some of its logic constraints. The constraints listed in the figure are relevant for analyzing the Low-Rise noise, meaning noise due to aggressor rising while the victim is low.

Comparing SLI used in [13] with our new approach we see that SLI is a 2-variable conjunctive term. For example the SLI *(a=1)->(b=0)* corresponds to the term $a*b$ and SLI *(x=1)->(y=1)* to the term $x*\bar{y}$. So the notation proposed here covers SLIs and generalizes them from bi-signal constraints to multi-signal constraints.

2.2 False noise analysis algorithm

The false noise analysis algorithm based on logic constraints is depicted in Figure 3. The input of the algorithm is a transistor level circuit and its critical noise clusters are specified by:

1. a single victim node v
2. a set of aggressor nodes $\{a_i\}$ that inject noise w_i
3. a noise type $t \in \{LowR, LowF, HighR, HighF, RiseR, RiseF, FallR, FallF\}$

The first four noise types correspond to functional noise where the victim net is either at a stable low state (*LowR* and *LowF*) or a stable high state (*HighR* and *HighF*), while the aggressor nets are rising (*LowR* and *HighR*) or falling (*LowF* and *HighF*). The second four noise types correspond to delay noise where the victim net is either rising (*RiseR* and *RiseF*) or falling (*FallR* and *FallF*) while the aggressor nets are rising (*RiseR* and *FallR*) or falling (*RiseF* and *FallF*).

The total noise is simply a sum of the contributions w_i of all aggressors. The goal of the algorithm is to derive logic correlations between the victim and aggressor nets and using them to find the subset of the aggressors providing the maximal sum of injected noise subject to the logic constraints. This problem is referred to as the *maximum realizable noise* problem and the set of aggressors responsible for the maximal realizable noise as the *maximal realizable aggressor set*.

Our false noise analysis algorithm has two stages: generating logic constraints for the circuit and analyzing each noise cluster. The logic constraints are generated by the resolution method from transistor level circuit representation and does not require preliminary logic function extraction. For noise analysis logic constraints of each noise cluster are represented in the form of ROBDD to simplify the search for maximum weighted set of aggressors satisfied to the logic constraints.

```
1. Read transistor level circuit description
2. Generate trivial logic constraints for MOS transistors
3. Derive logic constraints for DCCCs from transistor logic
   constraints by resolution technique
4. Derive circuit logic constraints applying resolution tech-
   nique to DCCCs logic constraints
5. For each noise cluster to be analyzed do the following
      2.1. Select logic constraints relevant to noise cluster
      2.2. Form logic constraint function in the form of
           ROBDD representing noise cluster logic constraints
      2.3. Find maximum weighted set of aggressors whose si-
           multaneous switching is not prohibited by the logic
           constraint function.
```

Figure 3. False noise analysis algorithm

3 Resolution method

The resolution method was originally proposed for mechanical theorem proving[16]. The resolution method (RM) is a method of deriving new boolean relations from existing ones. The traditional version of RM works with boolean relations in the form:

$$\prod_{i=1}^{m} f_i = 1 \qquad \text{(EQ 4)}$$

where each term f_i has the following form:

$$s^{\alpha_1}{}_{i_1} + s^{\alpha_2}{}_{i_2} + \ldots + s^{\alpha_k}{}_{i_k} \qquad \text{(EQ 5)}$$

where each $s^{\alpha}{}_j$ means a boolean variable s_j if $\alpha = 0$ or its inversion $\bar{s}_j$ if $\alpha = 1$. Using disjunctive form of terms is convenient for theorem proving because they correspond to assertances in the logic system. The resolution method uses the following derivation rule:

$$a + B = 1, \bar{a} + C = 1 \to B + C = 1 \qquad \text{(EQ 6)}$$

where B and C are any boolean functions. In the resolution method B and C are disjunctions of boolean variables some of which are taken with negation. Therefore the resulting relation $B+C=1$ is in the same form as the original ones. Traditionally the derivation rule is written in the simplified form by omitting equality signs and logic constant *1*:

$$a + B, \bar{a} + C \to B + C \qquad \text{(EQ 7)}$$

This notation can be interpreted as derivation of true logic sentence $B+C$ from the two true logic sentences $a+B$ and $\bar{a}+C$ thus helping theorem proving through sentence derivation.

Instead of the traditional version of RM, we use its modified formulation in which we work with logic constraints represented in the form given by (EQ2) and (EQ3). Each conjunctive term represents one prohibited combination of signals. The goal of the resolution method is deriving new logic constraints from the existing ones. The resolution rule is equivalently transformed into:

$$a \cdot B = 0, \bar{a} \cdot C = 0 \to B \cdot C = 0 \qquad \text{(EQ 8)}$$

where B and C are conjunctions of circuit signals or their inversions. Unlike the classical resolution method that derives new true sentences from the true ones our formulation is applied to the set of false sentences (logic constraints or prohibited signals combinations) and derives new false sentences. Both the formulations are equivalent. Similar to the resolution rule simplification (EQ7) we simplify our formulation of the resolution rule by omitting equality signs and the boolean constant *0*:

$$a \cdot B, \bar{a} \cdot C \to B \cdot C \qquad \text{(EQ 9)}$$

However we need to remember that it cannot be interpreted as derivation of a true sentence from two true premises. Instead it is deriving a logic constraint from two other logic constraints expressed as false sentences. An application of the resolution technique to deriving logic constraints is shown in Figure 4 and Figure 5 with examples of transistor and logic level circuits respectively.

The resolution rule is more general than the laws in SLI generation algorithm [13]. It covers transitive law, union rule, intersection rule (for both forward and lateral propagation) used there. Moreover the resolution rule can be used with logic constraints involving any number of signals.

4 Logic constraints generation

4.1 Transistor level constraints

In SLI based noise analysis approach [13] logic function is extracted for every DCCC output. This is easy only for small DCCCs like simple NAND and NOR gates. For large complex DCCCs full extraction of the logic function can be difficult or even impossible.

The resolution technique does not require full logic function extraction and can work directly at the transistor level. As a first step of the resolution technique we generate an initial set of logic constraints in the following way:

- For every n MOS transistor (*s* - source, *g* - gate, *d* - drain) two constraints are generated: $g*s*\bar{d}$, $g*\bar{s}*d$.
- For every p MOS transistor two constraints are generated: $\bar{g}*s*\bar{d}$, $\bar{g}*\bar{s}*d$.
- If a transistor terminal is connected to *Vdd* or ground, we generate a reduced set of constraints. For example, if source of p-type transistor is connected to Vdd, we generate constraint $\bar{g}*\bar{d}$. Similarly, if source of n-type transistor is connected to ground we generate constraint $g*d$.

All the initial constraints are obvious consequence of the MOS transistor operation in its *on* state. For example the constraint $g*s*\bar{d}$ for n MOS transistor means that if its gate is at high voltage it is impossible to keep its source at high voltage and drain at low.

After the initial constraints generation we apply the resolution rule multiple times for deriving new constraints. First we mark all the circuit nets that are involved in at least one noise cluster of interest. Then we process all the unmarked nets. Suppose that for net *N* we have n_1 constraints when *N* is high and n_0 constraints when *N* is low. If we make all possible resolutions to exclude signal *N*, then instead of n_0+n_1 old constraints we could have at most $n_0 n_1$ new ones. This number actually is much smaller because of the following cases:

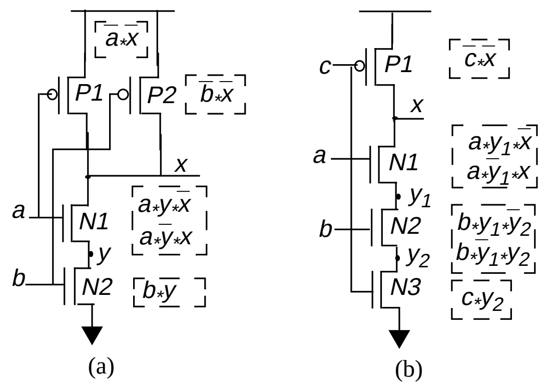
- A new constraint contains combination $a\bar{a}$ (tautology).
- A new constraint already exists in the original set, or is covered by an existing constraint, or covers existing constraints.
- New constraint can be merged with some existing constraint by the resolution rule. For example, $a*b*c$ and $a*b*\bar{c}$ are merged into $a*b$.

If the number of the new constraints is less than or equal to the number of old ones, the new constraints are used instead of the old ones, and net *N* is excluded from future analysis.

For some DCCCs or groups of DCCCs, the reduction using resolution is equivalent to logic function extraction. Figure 4 (a) shows this in an example of *NAND2* circuit with inputs *a* and *b*, output *x* and internal node *y*. Instead of five initial constraints $(\bar{a}*\bar{x},\bar{b}*\bar{x}, a*y, b*x*\bar{y}, b*\bar{x}*y)$ we build three new constraints $(\bar{a}*\bar{x},\bar{b}*\bar{x}, a*b*x)$ wherein the internal node *y* is eliminated. The new constraints express the gate's logic function. Another example of logic constraints generation is given in Figure 4 (b) where the resolution method is applied to a simple dynamic gate.

4.2 Logic level constraints

After completing constraints generation for all DCCCs we apply the resolution rule to constraints belonging to different DCCCs for generating logic level constraints. They differ in their selection of the constraints to which the resolution rule is applied. It is beneficial to exploit the fact that the constrains belonging to the DCCCs connected with their inputs and outputs often have common variables. Therefore it is convenient to use the algorithm similar to SLI propagation [13]. We derive new logic constraints by forward and backward propagation of constraints consisting of two literals



NAND2 logic constraints
$\bar{a}*\bar{x}$, $\bar{b}*\bar{x}$,
$b*y$, $a*\bar{y}*x$ -> $a*b*x$

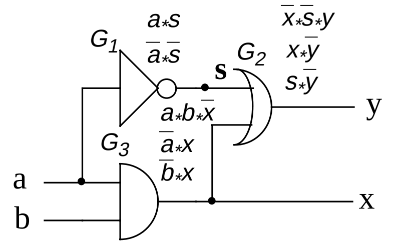
dynamic gate logic constraints
$\bar{c}*\bar{x}$
$c*y_2$, $b*y_1*\overline{y_2}$ -> $c*b*y_1$
$c*b*y_1$, $a*\overline{y_1}*x$ -> $c*a*b*x$

Figure 4. DCCC logic constraints calculation by resolution

(SLIs) through typologically ordered DCCCs and performing all possible resolution transformations. Our propagation algorithm differs from the original SLI propagation in the following:

- Transitive law and union rule are not used because false noise analysis does not require SLI that can be obtained in this way.
- Propagating SLIs through DCCC we use its constraints instead of its logic function.
- Propagating SLIs $(a_0*\bar{a}_1,\ a_0*\bar{a}_2,...,\ a_0*\bar{a}_{n-1})$ through DCCC with constraint $a_1*a_2*...a_n$, we generate new SLI a_0*a_n by applying the resolution rule multiple times.
- If a new constraint does not exist in the current set of constraints and cannot be derived from existing ones by transitive law, it is added to the set.
- If an existing constraint is covered by a new one, then we replace it with the new constraint.

Our propagation algorithm can generate constraints with more than 2 signals. We refer to a constraint involving *N* signals as N-LI. In Figure 5 we demonstrate that the lateral implication [13] *(y=0)->(b=0)* can be derived by the resolution technique.



G_1, G_2: $a*s$, $\bar{x}*\bar{s}\,\bar{y}$ -> $a*\bar{x}*\bar{y}$; $\bar{a}*\bar{s}$, $s*\bar{y}$ -> $\bar{a}*\bar{y}$ (R_1)
G_2, G_3: $a*b*\bar{x}$, $x*\bar{y}$ -> $a*b*\bar{y}$; (R_2)
R_1,R_2: $\bar{a}*\bar{y}$,$a*b*\bar{y}$ ->$\bar{y}*b*\bar{y}$ -> $b*\bar{y}$ (R_3)

Figure 5. Logic constraints derivation by resolution

5 Characteristic ROBDD and noise analysis

After logic constraints generation noise analysis is performed for every cluster for its respective noise type. In the SLI based algorithm [13], a constraint graph is formed based on generated SLIs, and then Maximum Weighted Independent Set problem is solved. In the proposed approach constraints involve many variables and a constraint graph turns into a hypergraph. Therefore, instead of the constraint graph, we construct ROBDD of the noise cluster constraints. We call it a noise cluster characteristic ROBDD.

Let a noise cluster contain a victim net *v* and aggressors nets $a_1,...,a_n$. The characteristic function of the cluster $f(v, a_1,..., a_n)$ is a function that equals *1* for combinations satisfying all the constraints and equals *0* otherwise.

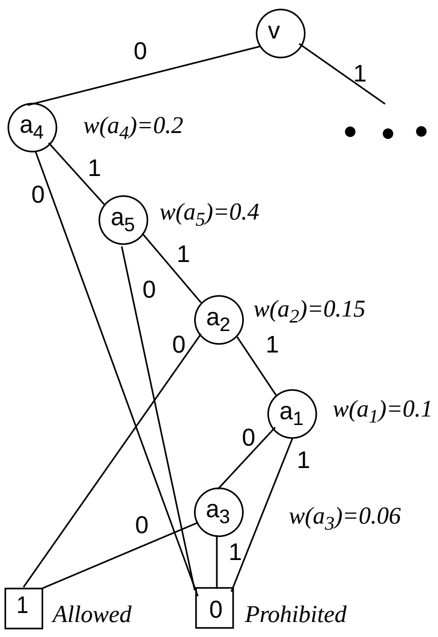
We construct characteristic ROBDD by the following recursive procedure. First we create a root vertex corresponding to the victim net *v*, and assign *v=0*. Then we make all possible conclusions from this assignment. For example, for a constraint $\bar{v}*\bar{b}$ we conclude *b=1*. Similarly a constraint such as $\bar{v}*c*d$ is reduced to $c*d$, and so on. After all the conclusions are made, we create a low-child of the root, i.e. the vertex corresponding to the aggressor a_1. Then we assign $a_1=0$ and again make all the conclusions throughout the circuit. If we meet a conflict (i.e. for some net *p* we obtain assignments *p=0* and *p=1*), then combination $(v=0,a_1=0)$ is prohibited by our constraints, and the low-child of the vertex a_1 is a terminal *0*-vertex. Then we try the next assignment $a_1=1$, otherwise we create a low-child of the vertex a_1 corresponding to the aggressor a_2, and so on. Thus we recursively build the BDD, and after subsequent reductions obtain the characteristic ROBDD. Using the characteristic ROBDD of the noise cluster we calculate the maximum noise of the given type by finding the maximum weighted set of the aggressors, for which simultaneous switching of the same type is not prohibited. It is the maximum weighted set of aggressors $\{a_{i1},a_{i2},...,a_{im}\}$, for which ROBDD has two paths from its root to the terminal 1-vertex $(v=V,a_{i1}=0,a_{i2}=0,...,a_{im}=0)$ and $(v=V,a_{i1}=1,a_{i2}=1,...,a_{im}=1)$ where *V* is the victim state corresponding to the analysed noise.

In Figure 5 we demonstrate the characteristic ROBDD for the circuit shown in Figure 2. The ROBDD describes the logic constraints for analyzing *LowR* noise injected into net *v* by nets *(a1,a2,a3,a4,a5)*. The noise injected by each aggressor is written near the correspondent ROBDD vertices. The maximum weighted aggressor set is *(a1, a3)* with total noise *0.16*.

6 Implementation and experimental results

The proposed false noise analysis algorithm is implemented in an industrial noise analysis tool called Clarinet [1]. The system was architected using a separate false noise analysis engine DiNo. First, the noise analysis tool performs the traditional noise analysis without using logic information. If generates a list of critical noise clusters where the total noise injected into the victim is higher than the tolerable noise threshold. The false noise analysis engine DiNo reads transistor level circuit description and a list of critical noise clusters. Then it generates logic constraints that could be useful for at least one critical cluster. For each critical cluster DiNo builds characteristic ROBDD and finds the maximum weighted aggressor set and the maximum feasible noise. The analysis can be performed both at the block and chip levels. At the block-level, the tool directly operates on the transistor level description of the circuit. At the chip-level, DiNo first pre-characterizes each gate in the library with a so-called *logic correlation black box*. These black boxes are then used in the chip-level generation of logic constraints. Figure 4 illustrates this methodology.

In Table 1, columns 2,3,4 and 5 we show the number of multi-signal constraints generated by resolution for ISCAS [15] bench-



Maximum Weighted Set of Aggressors for LowR noise is (a1, a3) with total weight 0.16

Figure 6. Characteristic ROBDD of noise cluster

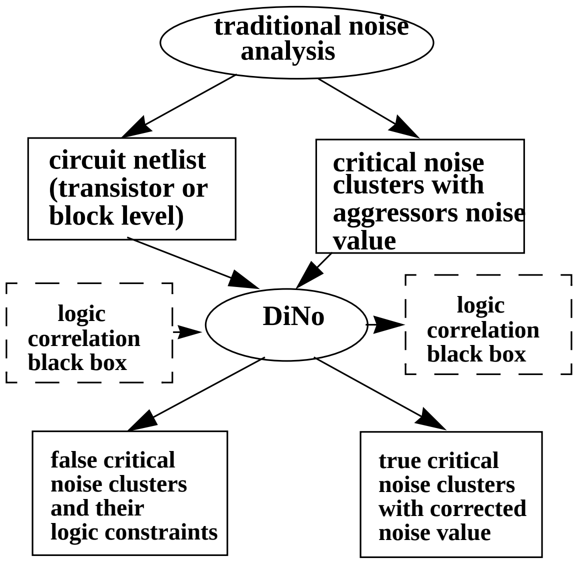

traditional noise analysis
circuit netlist (transistor or block level)
critical noise clusters with aggressors noise value
logic correlation black box
DiNo
logic correlation black box
false critical noise clusters and their logic constraints
true critical noise clusters with corrected noise value

Figure 7. Block diagram of the false noise analysis tool.

mark and industrial circuits. The results of the proposed method are compared with results of the SLI based approach [13]. The number of 2-LIs generated by the SLI method is shown in column 8. It can be seen that the proposed technique is effective in generating a larger set of constraints. The results in Table 2 demonstrate the effectiveness of the resolution based approach proposed in this paper. Column 3 shows the number of noise failures without false noise analysis. Column 4 shows the number of failures with false

circuit	#2-LI	#3-LI	#4-LI	#5-LI
c1355	873	72	49	3
c17	13	42	12	0
c1908	1151	42	70	15
c3540	1548	174	61	20
c499	669	49	63	35
c7552	589	106	30	5
c6288	587	715	434	213
plldriver	62	122	6	0
srot8	848	1614	155	29
xbar	541	168	16	0
srot16	2229	2591	1497	0
adder32	874	133	35	0
srot32	4233	6054	2742	19

Table 1. Number of generated constrains

circuit	clusters	failures	DiNo failures	% Reduction	Avg.Noise Reduction
c1355	107	214	160	25.234	35.891
c17	9	18	8	55.556	68.433
c1908	112	224	178	20.536	30.359
c3540	119	238	173	27.311	42.322
c499	100	200	154	23.000	32.389
c7552	118	236	209	11.441	20.279
c6288	118	236	192	18.644	34.461
plldriver	59	18	5	72.22	94.15
srot8	401	320	174	45.63	64.49
xbar	384	72	56	22.2	86.99
srot16	975	794	461	41.94	62.08
adder32	113	108	91	15.74	23.76
srot32	2005	2005	1104	44.94	53.70

Table 2. Noise analysis results

noise analysis as presented in this paper. The average number of aggressors in a cluster is about 10. The experiments were performed for noise threshold that is 5 times larger than the average noise of a single aggressor. It can be seen that as much as 72% of the original violations are identified as false violations by the proposed technique. Moreover the realizable noise is as much as 94% less than the noise reported without considering logic constraints.

7 Conclusions

In this paper, we presented a new approach for false noise analysis. We propose the use of resolution method for eliminating aggressor nets that cannot simultaneously switch. We have shown that the initial set of constraint terms in the resolution approach can be generated from transistor level circuit description. It was shown that single resolution rule can overlap some aspects of logic extraction and simple logic implication propagation. We explained how to generate a characteristic ROBDD for victim/aggressors cluster and then calculate worst-case combination of aggressors.The presented algorithms were implemented and tested on industrial circuits. Presented tables demonstrate the efficiency of proposed approach comparing with simple implication propagation method.

8 References

[1] R.Levy, et.al. “ClariNet: A noise analysis tool for deep submicron design”, DAC-2000, pp.233-238.

[2] P.Chen, K.Keutzer. “Towards True Crosstalk Noise Analysis”, ICCAD-99, pp.132-137.

[3] D.A.Kirkpatrick, A.L.Sangiovanni-Vincentelli. “Digital Sensitivity: Predicting Signal Interaction using Functional Analysis”, ICCAD-96, pp.536-541.

[4] F.M.Brown. “Boolean reasoning”, Kluwer Academic Publishers, 1990.

[5] G.Hachtel, R.Jacoby, P.Moceyunas, C.Morrison. “Performance Enhancements in BOLD using Implications”, ICCAD-88, pp.94-97.

[6] W.Kunz, P.R.Menon. “Multi-Level Logic Optimization by Implication Analysis”, ICCAD-94, pp.6-13.

[7] R.I.Bahar, et.al. “Symbolic Computation of Logic Implications for Technology-Dependent Low-Power Synthesis”, ISPLED-96.

[8] W.Long, Y.L.Wu, J.Bian. “IBAW: An Implication-Tree Based Alternative-Wiring Logic Transformation Algorithm”, ASPDAC-2000, pp.415-422.

[9] S.Bobba, I.N.Hajj. “Estimation of maximum current envelope for power bus analysis and design”, Int. Symp. on Phys. Des., 1998.

[10] A.Wroblewski, C.V.Schimpfle, J.A.Nossek. “Automated Transistor Sizing Algorithm for Minimizing Spurious Switching Activities in CMOS Circuits”, ISCAS-2000, pp.291-294.

[11] Shepard K.L. “Design methodologies for noise in digital integrated circuits”, Proc., DAC, 1998, pp. 94-99.

[12] A.Rubio, N.Itazaki, X.Xu and K.Kinoshita, “An Approach to the Analysis and Detection of Crosstalk Faults in Digital VLSI Circuits”, IEEE Trans. on CAD, Vol.13, No.3, 1997.

[13] A.Glebov, S.Gavrilov, D.Blaauw, S. Sirichotiyakul, C.Oh, V.Zolotov. “False Noise Analysis using Logic Implications”, ICCAD 2001, Nov. 2001, pp. 515-521.

[14] R.E.Bryant. “Graph-Based Algorithms for Boolean Function Manipulation”, IEEE Trans. on Computers, 1986, v.35, pp.677-691.

[15] F.Brglez, H.Fujiwara. “A Neutral List of 10 Combinational Benchmark Circuits”, Proc. IEEE ISCAS, IEEE Press, Pscataway, N.Y., 1985, pp.695-698.

[16] J.A.Robinson “A Machine-Oriented Logic Based on the Resolution Principle”, Journal of the ACM, 12(1): 23-41, 1965.