Elimination of false aggressors using the functional relationship for full-chip crosstalk analysis

Jae-Seok Yang, Jeong-Yeol Kim, Joon-Ho Choi, Moon-Hyun Yoo, Jeong-Taek Kong
CAE team, Memory Division, Dept. of Device Solution Network, Samsung Electronics
{sombrero, pass8000, foco, moonyoo, jkong}@samsung.co.kr

Abstract

As the portion of coupling capacitance increases in smaller process geometries, accurate coupled noise analysis is becoming more important in current design methodologies. We propose a method to determine whether aggressors can potentially switch simultaneously with the victim or not. The functional information is used to classify the aggressors. Our functional pruning algorithm inspects the conflict of the net states using CNF(Conjunction Normal Form) and BDD(Binary Decision Diagram). We present the experimental results on several industrial circuits. In the experiments, 6.4% of total aggressors are false and the accuracy of delay calculation can be improved up to 36.6%.

1. Introduction

As process geometries become smaller, the coupling capacitance of neighboring lines can contribute to a larger portion of the signal delay. When two coupled lines switch in the opposite direction, the interconnect delay increases. If they switch in the same direction, the interconnect delay decreases. The additional buffer insertion or the space increase of adjacent lines is required to prevent the crosstalk noise. These increase chip area[1] and power consumption. The crosstalk can cause not only delay variation but also functional failure[2],[3].

In noise analysis, all aggressors are assumed to switch simultaneously when we do not know the true switching relationship of the victim and aggressors. This assumption is conservative. The pessimism due to false aggressors has to be minimized because of the additional design cost to fix the noise.

The aggressor pruning can be accomplished by the functional and the temporal relationship. The functional information has been used to find a logically false aggressor[4]-[7]. In [4], the authors proposed an approach to find a pair of input vectors which maximize the crosstalk effect. A method to find the false aggressors using ATPG is presented in [5]. In [6], false aggressors are detected by a path sensitization procedure. Recently, the SLI(Simple Logic Implication) is used in [7]. However, these methods cannot be adopted into the industrial circuits having several million gates due to the high complexity. A heuristic method is required for full-chip crosstalk noise analysis.

In this paper, we present an efficient aggressor classification method. The newly proposed method limits the depth of backward search and uses a feature that two input cones of coupled nets must have a common portion to be logically related. If any common portion does not exist, the coupled lines are functionally independent. The CNF clause[8] is used to present the function of the victim and aggressor. The false aggressor problem is simply reduced to SAT(satisfiability) problem[9]. Finally, the aggressor classification is done using BDDs[10]. Our method can apply to the noise analysis as well as the delay calculation. We suppose the zero-delay model and it is extended to the variable real delay model in Section 2.3.

The aggressor classification method is explained in Section 2. We present a method to reduce the complexity in Section 3. In Section 4, the experimental results are shown. We conclude in Section 5.

2. Functional aggressor pruning

2.1. Problem definition

When an aggressor always switches in the same direction with a victim, the coupled lines are “in-phase relationship”. The coupled lines are “out of phase relationship” if an aggressor switches in the opposite direction. An aggressor is always true if the coupled lines are “independent relationship”. To classify the aggressor relationship, we propose the following statements assuming the zero-delay model.

Lemma 1: The “out of phase relationship” of the coupled lines is valid if both vectors satisfying the statement(1) and (2) exists.

$$At \quad t = 0, \quad aggressor = 'HIGH' \quad and \quad victim = 'LOW' \tag{1}$$

$$At \quad t = \infty, \quad aggressor = 'LOW' \quad and \quad victim = 'HIGH' \tag{2}$$

Lemma 2: The “in-phase relationship” of the coupled lines is valid if both vectors satisfying the statement(3) and (4) exists.

0-7695-1881-8/03 $17.00 © 2003 IEEE

$$At \quad t=0, \quad aggressor='LOW' \quad and \quad victim='LOW' \qquad (3)$$

$$At \quad t=\infty, \quad aggressor='HIGH' \quad and \quad victim='HIGH' \qquad (4)$$

The "independent relationship" is the case that both "out of phase relationship" and "in-phase relationship" are available.

An example that determines the relationship of coupled lines using BDD is shown in Figure 1. We simply consider the case that a victim has a rising transition. V1 is a victim and A1 is an aggressor of V1. The function of V1 is $N2 \bullet N3$ and the function of A1 is $N1 + N2$. The BDD descriptions of V1 and A1 are represented in Figure 2. The left branches are true states and the right branches are false states in BDD descriptions.

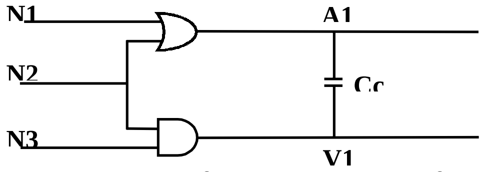


Figure 1. An example for the aggressor classification.

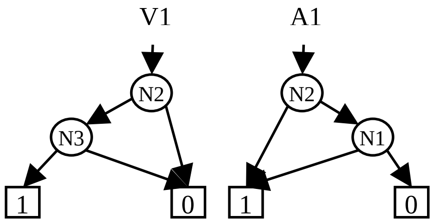


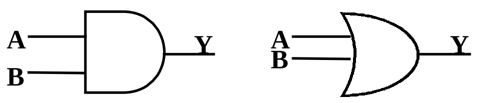
Figure 2. The BDD representation of V1 and A1.

The opposite direction switching must satisfy the Lemma 1. At t=0, A1 is high when N2 is high. V1 is low when N2 is high and N3 is low from the BDD in Figure 2. Thus, the input vector satisfying the statement (1) exists.

At t=∞, A1 is low when both N2 and N1 are low. V1 is high when both N2 and N3 are high. It is a contradiction that the value of N2 is high and low simultaneously. From this observation, we can conclude that the "out of phase relationship" of V1and A1 is invalid. In this case, A1 is a false aggressor and the coupling capacitance(Cc1) cannot affect the waveform of V1. We present a tidy method for finding the conflict in the next Section.

2.2. Aggressor classification using BDD

Usually, the output function of a gate is represented by inputs. However, when the function of a gate is represented by the CNF clause, we can see the valid input value for a specific output value. The modified CNF clause is obtained by exclusive-OR operation with the output signal. Simply, we call the exclusive-OR operation as the CNF. This notation is unusual but easy to implement. For instance, the CNF clause of a 2-input AND gate and OR gate is presented in Figure 3.



Function $Y = A \bullet B$ Function $Y = A + B$

CNF $Y \oplus (A \bullet B) = 0$ CNF $Y \oplus (A + B) = 0$

Figure 3. The CNF clauses of AND and OR.

The CNF clause is useful to classify the aggressor because the state of an internal node can be known for a specific output node value. If a node has to be a different value at the same time for the crosstalk transition, the crosstalk should not be occurred. To see the existence of a contradiction, we propose Lemma 3 and 4 using the CNF clauses of the victim and aggressor.

Lemma 3: The "out of phase relationship" of the coupled lines is valid if both solutions satisfying Boolean equation (5) and (6) exists.

$$\begin{gathered} When \quad aggressor='HIGH' \quad and \quad victim='LOW' \\ CNF\_of\_victim \; + \; CNF\_of\_aggressor \; = \; 0 \end{gathered} \qquad (5)$$

$$\begin{gathered} When \quad aggressor='LOW' \quad and \quad victim='HIGH' \\ CNF\_of\_victim \; + \; CNF\_of\_aggressor \; = \; 0 \end{gathered} \qquad (6)$$

Lemma 4: The "in-phase relation" of the coupled lines is valid if both solutions satisfying Boolean equation (7) and (8) exists.

$$\begin{gathered} When \quad aggressor='HIGH' \quad and \quad victim='HIGH' \\ CNF\_of\_victim \; + \; CNF\_of\_aggressor \; = \; 0 \end{gathered} \qquad (7)$$

$$\begin{gathered} When \quad aggressor='LOW' \quad and \quad victim='LOW' \\ CNF\_of\_victim \; + \; CNF\_of\_aggressor \; = \; 0 \end{gathered} \qquad (8)$$

In Figure 1, the CNF clauses of V1 and A1 are Boolean equation (9) and (10), respectively.

$$F(V1, N2, N3) = V1 \oplus (N2 \bullet N3) = 0 \qquad (9)$$

$$F(A1, N1, N2) = A1 \oplus (N1 + N2) = 0 \qquad (10)$$

The conjunction of Boolean equation (9) and (10) is Boolean equation (11) when V1 is low and A1 is high. Boolean equation (12) is when V1 is high and A1 is low. In Figure 4, the left branches present true states and the right branches present false states.

$$F(V1=0, A1=1, N1, N2, N3) = N2 \bullet N3 + \overline{(N1+N2)} = 0 \qquad (11)$$

$$F(V1=1, A1=0, N1, N2, N3) = \overline{N2 \bullet N3} + N1 + N2 = 0 \qquad (12)$$

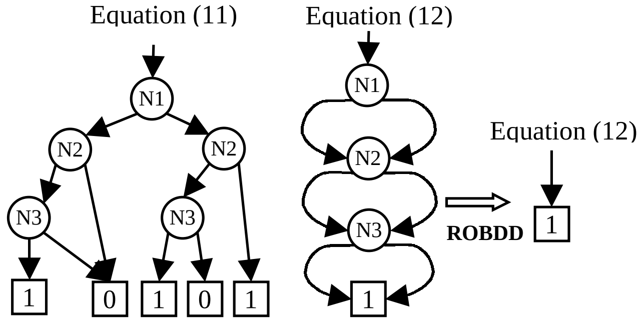


Figure 4. The BDD representation of CNF clause.

From Figure 4, we know that the solutions of Boolean equation (11) are {N1=1, N2=0}, {N1=1, N2=1, N3=0} and {N1=0, N2=1, N3=0}. It is clear that the equation(12) dose not have a solution because the condition at V1=1 and A1=0 is impossible. From Lemma 3, we can conclude that A1 is false. It is the same result in Section 2.1. We can classify false aggressors by building BDDs and reducing them.

2.3. Glitch consideration

A true aggressor may be false when the gate delay is ignored because of a glitch transition. Now, we expand the previous heuristic method to be a real delay model. In Figure 1, we assume that the delay of OR gate is 1~3ns and AND gate is 2~4ns. The gate delay normally has a variable range. The Boolean variable has a tag with a time range. To consider the glitch transition, the Boolean equation (11) and (12) should be modified as the following Boolean equations.

$$N2[-4\le t\le -2]\bullet N3[-4\le t\le -2]+\overline{(N1[-3\le t\le -1]+N2[-3\le t\le -1])}=0 \quad (13)$$

$$\overline{(N2[-4\le t\le -2]\bullet N3[-4\le t\le -2])}+N1[-3\le t\le -1]+N2[-3\le t\le -1]=0 \quad (14)$$

The Boolean equation(14) is true because each of N2 has a different time range. The variable states from Boolean equation(13) and (14) are presented in Figure 5. When the gate delay is considered, A1 may be a true aggressor.

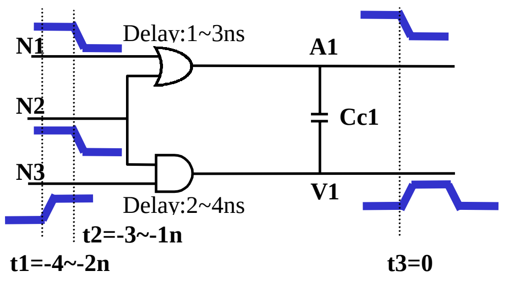


Figure 5. Aggressor classification under the non-zero delay model.

If t1=-2n and t2=-3n, V1 may be no transition. However, we assume that it is a true aggressor to have conservatism.

To be a true aggressor, the leaf node cannot include the glitch transition. It is an apparently true when the leaf nodes are flip-flop output or primary input. If there are all leaf nodes without glitches, the aggressor is true.

3. Search space reduction

As the number of nodes increases, the complexity of BDD increases exponentially. Thus, we need to limit the search space to save the run-time and memory usage. The connected lines by coupling capacitances are geometrically near at hand each other. The closeness of placement always does not mean a functional relationship. The coupled lines must have a common net in their input cone in order to have a functional relationship. We define a net as "common net" if the net belongs to both backward search cones from a victim and an aggressor.

If a common net contradicts to have a high value and a low value concurrently, the vector causing the crosstalk cannot exist. If there is no common net, the coupled lines are assumed to be "independent relation". In Table 1, we present the ratio to have a common net as the backward search depth increases. Design A is a DSP core block. Design B is the modem chip. As the backward search depth is increased, the possibility to have a common net is increased. To improve the ratio of pruned aggressors, we can increase the search depth.

Table 1. The portion of common nets.

Backward Search Depth	Design A		Design B	
	# of Common nets	Ratio	# of Common nets	Ratio
1	1,602	5.4%	16,738	10.5%
2	2,984	10.0%	40,686	25.6%
3	4,716	15.8%	46,248	29.1%
4	6,772	22.7%	53,242	33.5%
5	8,748	29.4%	60,076	37.8%

If any common net does not exist within a given backward search depth, we regard the aggressor as a true one. If a common net is near at the coupling capacitance, the functional relationship of the victim and aggressor is strong because it has little interference of the side inputs (Figure 6(a)). With a far common net (Figure 6(b)), the functional relationship is weak because the side inputs can cause a coupled interaction regardless of the common net state.

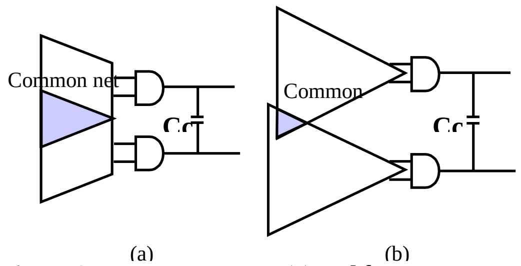


Figure 6. Near common net (a) and far common net (b).

4. Experimental results

CUDD package[11] is used to build BDD. The run-time is on a Blade1000 750MHz. We run the proposed method on five industrial circuits. The characteristics of the circuits are shown in Table 2. The design A and B are the same as shown in Table 1. The number of victims means the number of nets that have one or more aggressors.

The results of aggressor pruning are shown in Table 3 and 4. The transition of the opposite direction is assumed in the results. In design A, 3.5% of total aggressors are logically false when the backward search depth is five.

Table 2. The summary of the circuit information.

	A	B	C	D	E
# of instance	58K	198K	79K	296K	350K
# of net	60K	228K	85K	326K	372K
# of victim	27K	93K	37K	86K	121K
# of aggressor	48K	159K	76K	152K	209K
Aggr. per victim	1.79	1.72	2.04	1.75	1.73

The number of logically false aggressors increases as the search depth is increased. When the depth is larger than three, the increment of the pruning ratio slows down. This agrees with the assumption that a farther common net has a weaker functional relationship. We should determine the speed and functional pruning ratio by controlling the backward search depth. As the search depth becomes larger, the pruning ratio is increased while the run-time gets longer.

Table 3. The results of aggressor pruning for design A.

Backward Search Depth	False Aggressor	Ratio	Run-time
1	32	0.1%	34s
2	382	1.3%	42s
3	778	2.6%	57s
4	932	3.1%	67s
5	1,024	3.5%	79s

Several circuits are used to verify the proposed method in Table 4. In design B, the ratio of functionally false aggressor is 6.4% when the backward search depth is two. The largest design E(# of net : 372K) is executed within 660s. The run-time is a reasonable to be adopted as a crosstalk analysis tool.

Table 4. The results of aggressor pruning of four designs.

depth			B	C	D	E
1	Common net	Number	16,738	5,662	22,074	14,706
		Ratio	10.5%	7.5%	14.6%	7.0%
	False aggressor	Number	3,662	1,002	5,608	3,092
		Ratio	2.3%	1.3%	3.7%	1.5%
	Run-time		164s	136s	254s	414s
2	Common net	Number	40,686	10.968	45,396	24,912
		Ratio	25.6%	14.5%	30.0%	11.9%
	False aggressor	Number	10,188	2,722	8,994	6,904
		Ratio	6.4%	3.6%	5.9%	3.3%
	Run-time		251s	281s	383s	660s

Table 5.The delay reduction due to pruning in design C.

	Total aggressors	Pruned aggressors	W/O pruning delay(ns)	With pruning delay(ns)	Delay reduction
Net0	5	5	0.186	0.118	36.6%
Net1	1	1	0.432	0.349	19.0%
Net2	1	1	0.473	0.387	18.2%
Net3	1	1	0.537	0.441	17.8%
Net4	1	1	0.537	0.442	17.7%

In design C, all nets of having false aggressors are examined. The delay is measured using HSPICE. The partial circuit by cutting the coupled net is simulated. Net0 has five aggressors. All of them turn out to be false by functional pruning. The delay in Table 5 is the sum of the net delay and driver gate delay. The delay reduction of Net0 is 36.6%. If Net0 belongs to a critical path, the longest delay should be overestimated by 0.068ns without the proposed method.

5. Conclusion

In this paper, we propose the method to eliminate false aggressors for accurate crosstalk analysis. The advantages of the proposed method are summarized as: 1) the pessimism of crosstalk analysis should be reduced. 2) the over-shielding should be reduced. 3) the additional run-time is not needed.

We present the logical information of the coupled lines by the CNF clause. To determine the relationship, BDD is built from the CNF clause. The glitch transition is considered to guarantee the conservatism and the locality feature is useful to reduce the complexity. The algorithm works on the large industrial circuits in virtue of the efficiency. Consequently, the proposed method removes the false aggressors of 6.4% within a few minutes and the errors of delay calculation can be reduced up to 36.6%. The proposed method can improve the accuracy for the crosstalk analysis without additional time consumption.

6. References

[1] H. P. Tseng, L. Scheffer and C. Sechen, “Timing-and Crosstalk-Driven Area Routing”, IEEE Trans. Computer Aided Design, vol.20, no.4, Apr., 2001, pp. 528-544.

[2] K. L. Shepard and V. Narayanam, “Conquering Noise in Deep-Submicron Digital Ics”, IEEE Trans. Design and Test of Computers, Jan.-Mar., 1998, pp. 51-62.

[3] A. B. Kahng, S. Muddu and D. Vidhani, “Noise and Delay Uncertainty Studies for Coupled RC Interconnects”, Proc. Int. Conf. Asic/SOC, 1999, pp. 3-8.

[4] P. Chen and K. Keutzer, “Towards True Crosstalk Noise Analysis”, Proc. Int. Conf. Computer Aided Design, 1999, pp. 132-137.

[5] R. Arunachalam, R. D. Blanton and L. T. Pileggi, “False Coupling Interactions in Static Timing Analysis”, Proc. Design Automation Conference, 2001, pp. 726-731.

[6] T. Xiao and M. Marek-Sadowska, “Functional Correlation Analysis in Crosstalk Induced Critical Paths Identification”, Proc. Design Automation Conference, 2001, pp. 653-656.

[7] A. Glebov, S. Gavrilov, D. Blaauw, S. Sirichotiyakul and C. Oh, “False-Noise Analysis using Logic Implications”, Proc. Int. Conf. Computer Aided Design, 2001, pp. 515-521.

[8] T. Larrabee, “Test Pattern Generation Using Boolean Satisfiablity”, IEEE Trans. Computer Aided Design, v11, Jan., 1992, pp. 4-15.

[9] J. P. Marques Silva, K. A. Sakallah, "GRASP: A new search algorithm for satisfiability", Proc. ACM/IEEE Int. Conf. Computer-Aided Design, pp.220-227, Nov. 1996.

[10] R. E. Bryant, “Graph-Based Algorithms for Boolean Function Manupulation”, IEEE Trans. Computers, v35, 1986, pp. 677-691.

[11] F. Somenzi, CUDD: CU Decision Diagram Package.