Crosstalk Alleviation for Dynamic PLAs

Tzyy-Kuen Tien Shih-Chieh Chang* Tong-Kai Tsai**
Department of Electrical Engineering
**Department of Computer Science and Information Engineering
National Chung Cheng University, Chiyai, Taiwan 621, Republic of China
*Department of Computer Science
National Tsing Hua University, HsingChu, Taiwan 300, Republic of China

Abstract

The dynamic PLA style has become popular in designing high performance microprocessors because of its high speed and predictable routing delay. However, like all other dynamic circuits, dynamic PLAs have suffered from the crosstalk noise problem. In this paper, we propose two techniques to alleviate crosstalk noise for dynamic PLAs. The first technique makes use of the fact that depending on the ordering of product lines, some crosstalk does not cause errors in outputs. A proper ordering can greatly reduce the number of lines affected by crosstalk noise. For those product lines which can be affected by crosstalk, we attempt to reduce the parallel length by re-ordering the input and output lines. We have performed experiments on a large set of MCNC benchmark circuits. The results show that after re-ordering, 86.7% of product lines become crosstalk immune and need not be considered for crosstalk prevention.

1. Introduction

The wire delay has become a dominating factor in deep sub-micron design. However, the wire delay is difficult to predict during the logic design step when the standard cell based design style is applied. Instead of using the standard cell style, it is now popular to employ the dynamic PLA style for the control logic of a high performance processor. It is not only because a dynamic PLA can provide high speed but also because the wire delay is easy to predict in the logic design phase. It was reported in literature [7] that a 1G Hz microprocessor utilizes the dynamic PLA style. In [4], the authors also showed that the delay of using a network of dynamic PLAs is 15% faster than that the delay of using the standard cell approach for some benchmark circuits. Our experiments on 0.18µ TSMC technology also showed that a X86 controller design using a dynamic PLA style could be twice faster than the standard cell design style.

This work is supported by NSC contract #89-2218-E-194-028.

Dynamic PLAs [1][3][9][10] are different from traditional **static** PLAs in that both the AND and OR planes are realized by the dynamic NOR structure because the NOR structure has smaller delay than the static AND/OR structure. Like other dynamic circuits, dynamic PLAs have large power consumption. Still some styles of dynamic PLAs [1][9][10] are more efficient in power consumption than others [3]. Figure 1 shows an example of Blair's dynamic PLA [1]. During the pre-charged phase, the clock signal is high which turns on both the *clocked N transistors* in the AND plane and *clocked P transistors* in the OR plane. Therefore, in the pre-charged phase, all product lines are pre-discharged to ground while all output lines are pre-charged to high. The feature of pre-discharge to ground is a design technique, which allows the Blair's PLA to have low power consumption. It is because that the probability is high for a product line (driven by a NOR gate) staying low in the evaluation phase.

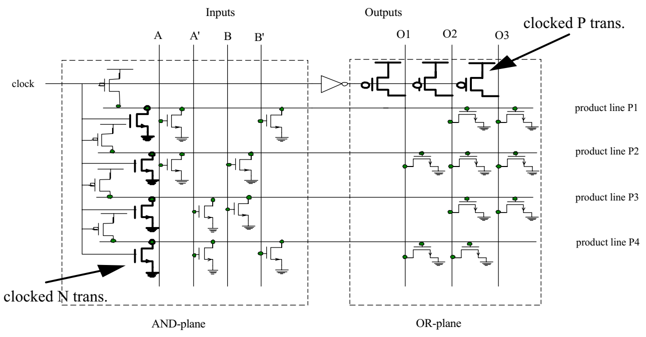

Figure 1 An example of Blair's PLA.

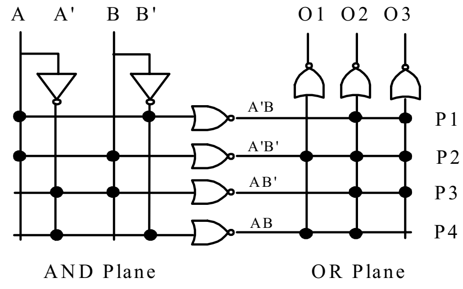

Figure 2 A shorthand example of Figure 1.

Despite high performance and predictable wiring delay, a dynamic PLA may suffer from the crosstalk problem. A crosstalk noise is directly proportional to the parallel length of two adjacent lines. Since product lines of a dynamic PLA may be parallel to the adjacent product lines on the same layout layer for a long distance, the noise can be large enough to cause erroneous outputs. A shorthand example of Figure 1 is shown in Figure 2. In the example, we have *O1'=P2+P4= (A+B)'+(A'+B')'= A'B'+AB*. Let us assume the crosstalk noise appears when product lines *P1* and *P3* make transitions from low to high while product lines *P2* and *P4* stay at low. The crosstalk noise produces noise glitches on product lines *P2* and *P4*, which may discharge output *O1*. If the crosstalk noise is large enough, it may cause functional error in *O1*.

To prevent the cross talk problem, Khatri [4] uses a static weak pull-up device to prevent the crosstalk noise in dynamic PLA. Xiao [11] prevents the cross talk noise by increasing transistor size or spacing for a given layout. Several efforts are proposed to reduce the crosstalk effect in the routing [2][4][5][8][11]. In general, most crosstalk prevention techniques may degrade performance and incur area and power overhead.

In this paper, we propose two techniques to alleviate the crosstalk problem for the dynamic PLAs [1][9][10], where a product line switches from low to high. We observe that some crosstalk, no matter how severe does not cause errors in outputs. For example in Figure 2, consider output line *O3=(P1+P2+P3)'*, which contains two adjacent product lines *P1* and *P2*. Assume product line *P2* switches from low to high and induces a noise glitch on product line *P1*. The output line *O3* will switch from high to low anyway disregarding whether there is a noise glitch in *P1*. In other words, the error or fault caused by a noise glitch in *P1* cannot propagate to *O3*. In this example, product line *P1* is positioned next to *P2* and output *O3* is not affected by the crosstalk noise. Therefore, by re-ordering the product lines, we may reduce the number of lines affected by crosstalk noise. In the other technique, we also observe that the parallel length between two adjacent lines depends on the input and output ordering. For those product lines which can be affected by crosstalk, we attempt to reduce the parallel length by re-ordering the input and output lines. Note that our techniques cannot completely eliminate the crosstalk problem and the protection techniques [11] are still needed to completely eliminate the cross talk. Also, we do not consider the crosstalk noise at the input (output) lines because the input (output) lines can be separated by ground or power lines as in [4][6]. We have performed experiments on a large set of MCNC benchmark circuits. The results show that after re-ordering, 86% of product lines become crosstalk immune and need not be considered for crosstalk prevention.

2. Crosstalk alleviation in dynamic PLA

In this section, we discuss two techniques to alleviate the crosstalk noise in a dynamic PLA. A product line is called an *aggressor* if its switching can induce noise on its neighborhood lines while a product line on which noise is induced is called a *victim* line. Except the two product lines at the top and bottom positions, each product line has two adjacent lines. Therefore, each product line may have two aggressors and also each aggressor may have two victim lines.

2.1. Product lines re-ordering to alleviate crosstalk noise

We now discuss the conditions that outputs are not affected by the crosstalk noise in dynamic PLAs. Consider again in Figure 2 that if aggressor *P2* induces a glitch on victim *P1*, output *O3* is not affected. One can find that if an aggressor and its victim are both inputs of a NOR gate, the gate's output is not affected by the crosstalk noise. We have the following definitions.

Definition 1 *(Output set)*: Assume that a dynamic PLA consists of *n* product lines *P1*, *P2*, …, *Pn*, and *m* output lines *O1*, *O2*, …, *Om*. We say the *output set* of a product line *Pi* is the set of outputs sharing product line *Pi*.

For example in Figure 2, the output set for product line *P1* is {*O2*, *O3*}, for *P2* is {*O1, O2, O3*}, for *P3* is {*O2, O3*} and for *P4* is {*O1, O2*}.

Definition 2 (*CT-immune*): A victim line *P1* is said to be *crosstalk immune* or *CT-immune to* its adjacent (aggressor) line *P2* if crosstalk noise from (aggressor) *P2* to (victim) *P1* never causes errors in the *P1's* output set. We also say that a product line *P1* is a *CT-immune line* if *P1* is CT-immune to both of its adjacent lines.

The main objective in the first technique is to find a good ordering of product lines to maximize the number of CT-immune lines or minimize non CT-immune lines. The reason is that a non CT-immune line may need special cares for preventing the crosstalk to occur. Note that all prevention methods for crosstalk incur additional area/power/performance penalty. Therefore, by minimizing the non CT-immune lines, we can reduce the additional area/power/performance cost. First, let us discuss the conditions for a product line to be a CT-immune line.

Consider Figure 2. Since *P2's* output set {*O1, O2, O3*} contains *P3's* output set {*O2, O3*}, we have product line *P3* CT-immune to *P2*. It is because that if aggressor *P2* switches from low to high, *P2's* outputs {*O1, O2, O3*} will switch from high to low so a noise glitch in victim *P3* does not cause errors. Also, if we re-order product lines in Figure 2 from *{P1, P2, P3, P4}* to *{P1, P3, P2, P4}*, line *P3* becomes CT-immune because *P3's* output set {*O2, O3*} is contained in *P2's* output set {*O1, O2, O3*} and *P1's* output set {*O2, O3*}. In such a case, we need not consider whether *P1* or *P2* may induce a crosstalk noise on *P3*.

From the example, we can find that by properly ordering *P3's* position in between *P1* and *P2*, product line *P3* becomes a CT-immune line. Therefore, different ordering may result in different number of CT-immune lines. To maximize the number of CT-immune lines, we first build a graph called a *CT-immune* graph. The graph is a directed graph. A node *pi* in the graph represents a product line *Pi* in a PLA. If product line *P1* is CT-immune to *P2*, there is an edge from the corresponding node *p2* to node *p1*.

Consider the example in Figure 2. Its CT-immune graph is shown in Figure 3(a). Since the output sets of product line *P1* and *P3* are the same. Therefore, product lines *P1* and *P3* are CT-immune to each other and there are directed edges from *p1* to *p3* and *p3* to *p1*. Also, *P3* is CT-immune to *P2* because the output set of *P2*, {*O1, O2, O3*} is a superset of *P3's* output set {*O2, O3*} so there is a directed edge from *p2* to *p3*.

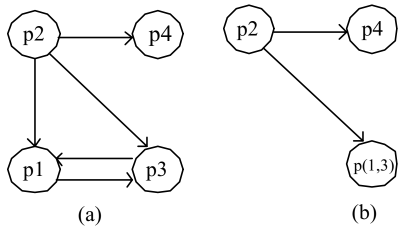

Figure 3 The CT-immune graph of Figure 1.

Basically, our algorithm is performed in two steps in Figure 4. In the first step, we look for a set of output lines whose output sets are the same. Those product lines with the same output set are CT-immune to each other and the corresponding nodes must have edges pointing to each other in the CT-immune graph. Intuitively *n* product lines, which are CT-immune to each other, should be placed together.

```
Product_Lines_Re-ordering()
{
  Construct CT-immune graph();
  Step1: Group the product lines();
  Step2: Find CT chains();
}
```

Figure 4 The algorithm of product lines re-ordering.

After grouping those lines that are CT-immune to each other, we then reduce the CT-immune graph to a new graph as follows. One new node replaces the corresponding nodes of the mutual CT-immune lines. The new node has the same incoming and outgoing edges of the replaced nodes. For example in Figure 3(a), since *P1* and *P3* have the same of output set, all outgoing or incoming edges of *p1* and *p3* are the same. The new node *p(1,3)* represents the grouping of product lines *P1* and *P3*. The updated graph is shown in Figure 3(b).

In the second step, we greedily give partial ordering of some product lines to obtain as many CT-immune lines as possible. After the first step, the best result we can get is to have an alternation of a CT-immune line followed by a non CT-immune line. We called such a chain, a *CT chain*. For example, in Figure 5, product lines *P2*, *P4* and *P6* are CT-immune lines while product lines *P1*, *P3*, *P5*, *P7* are non CT-immune lines. Our greedy algorithm attempts to give a partial ordering so that we can have a CT chain as long as possible. When a CT chain cannot grow any longer, we build another CT chain. We illustrate our greedy algorithm by the following example.

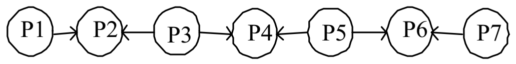

Figure 5 The example of a CT chain.

Consider the example in Figure 6(a) that there is no CT-immune line. In the first step of our algorithm, we finds that product line *P4* and *P6* have the same output set so they are grouped together. After grouping *P4* and *P6*, the updated CT-immune graph is shown in Figure 6(b). Then, in the second step, we attempt to build long CT chains. In this step, we first choose a product line with the maximum number of incoming edges in the updated CT-immune graph, i.e., the (grouped) product lines *P(4,6)*. We intend to build a CT chain around product lines *P(4,6)*. There are two directions, the "left" and the "right" directions, to grow a chain. Let us start to grow from the "right" side. Among the nodes in the graph which have an edge pointing to *p(4,6)*, we choose one with the largest outgoing edges which is *p1*. We then add *P1* to the CT chain and the chain becomes {*P(4,6)<-P1*} where the arrow indicates product lines *P(4,6)* is CT-immune to *P1*. Again among nodes to which *p1* has edges pointing, we choose one with the largest incoming degree which is *p2*. The CT chain becomes {*P(4,6)<-P1->P2*}. The chain growing process continues till we have {*P(4,6)<-P1->P2<-P7->P5*}. After that we try to grow the chain from the "left" side of *P(4,6)* using the similar process. Finally, the chain becomes {*P3->P(4,6)<-P1->P2<-P7->P5*}. One can find that this ordering has four CT-immune lines, which are {*P4, P6, P2, P5*}. The new ordering is shown in Figure 6(c). Note that in this example, we terminate the algorithm with one CT chain since all product lines are in the chain. In other cases, several CT chains may be needed.

2.2. Input and output lines re-ordering to alleviate crosstalk noise

In the previous section, we attempt to maximize the number of CT-immune lines. In this section for those non

CT-immune lines, we propose another technique to reduce the effect of the crosstalk noise.

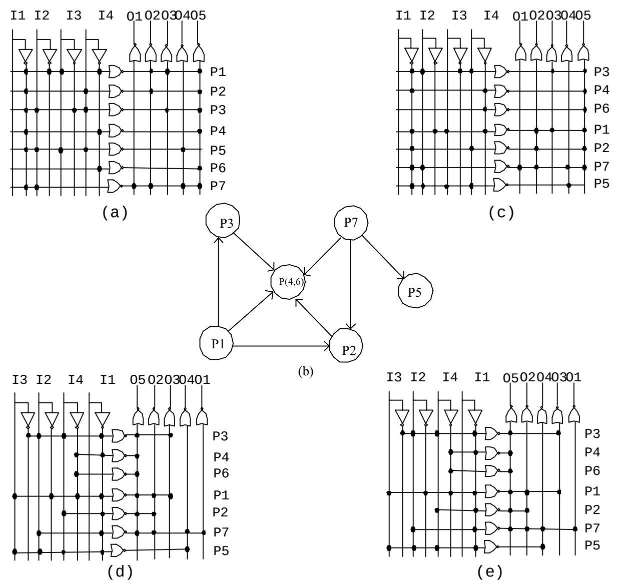

Figure 6(a) An example of a PLA (b) The CT-immune graph (c) The result of product lines re-ordering (d) The initial ordering of input/output line re-ordering (e) The result of input and output lines re-ordering.

In the PLA structure, product lines need not have the same length. For example, product line *P5* in Figure 6(c) can end till *O4*. The extra length from *O4* to *O5* can be removed. Therefore, the length of a product line depends on the ordering of input/output lines. For example let us consider the new ordering of {*O5, O2, O3, O4, O1*} in Figure 6(d). After the adjustment, the length of *P4* is greatly reduced because the portion of *P4* from *O2* to *O1* can be removed. The length reduction for *P4* can ease the crosstalk effect on its adjacent line *P3* because crosstalk noise is directly proportional to the parallel length between *P3* and *P4*.

In this section, we attempt to reorder the input/output lines so that the parallel length of those non CT-immune lines can be minimized. Since the layout of PLA is regular, we assume the spacing between adjacent input/output lines is 1. For each non CT-immune line, we compute the parallel length to its adjacent lines and then the cost is equal to the summation of all the parallel lengths of non CT-immune lines. For example, the parallel length of a non CT-immune line P3 is equal to 4 (three in the AND plane and one in the OR plane).

Our output-ordering algorithm consists of two steps. We first attempt to find a good initial ordering followed by an improvement step. In the initial ordering step, we order the output lines according to the number of product lines. For example in Figure 6(c), output line *O5* has the maximum number of product terms so it is ordered at the top of the ordering while output line *O1* has one product term so it is at the bottom of the ordering. After the first step, the initial ordering becomes {*O5, O2, O3, O4, O1*} as shown in Figure 6(d). The initial algorithm in general reduces the length for all product lines. Then, we adopt an iterative swapping algorithm to improve the results. For the same example, if output line *O3* and *O4* are swapped, the parallel length between *P7* and *P5*, a non CT-immune line is reduced while all other parallel lengths do not increase. The iteratively swapping is performed until there is no improvement. We use the same method to re-order the inputs. Figure 6(e) shows the final result of input/output lines re-ordering. Note that in the example the length of a product line goes from the AND plane to the OR plane. If inter-plane buffers are considered, the length on the AND plane and the OR plane should be considered separately.

3. Experimental Results

We have implemented the techniques presented in Section 2 to minimize the non CT-immune lines in the circuits and performed experiments on a large set of MCNC PLA benchmark. All the example circuits are minimized by ESPRESSO for two level logic optimization.

Table 1 shows the results of re-ordering the product lines. The first column shows the name of a circuit, the second column shows the number of product lines. In the third column, we show the number of non CT-immune lines before applying our algorithm. In the fourth column, we show the number of non CT-immune lines after our re-ordering algorithm. The last column shows the computation time performed on SUN workstation Ultra 10. For example, the number of product lines for circuit *x7dn* is 538. Before the re-ordering algorithm, the number of non CT-immune lines for circuit *x7dn* is 492. After the re-ordering, the number of non CT-immune lines is reduced to 28. On the average, the number of non CT-immune lines is only 13.3% of that in the product lines.

In Table 2, we also show the results of the input/output re-ordering technique. Before re-ordering, the second column shows the total parallel lengths of all non CT-immune lines described in Section 2.2. After the input/output re-ordering, the reduced cost is shown in the third column. The last column shows the computation time. For example the circuit *x7dn*, the total parallel length of non CT-immune lines to its adjacent line is 2026. After re-ordering, the total length becomes 776. On the average, the total parallel length of non CT-immune lines is reduced to 58% of that without input/output line re-ordering but after product line re-ordering.

4. Conclusion

This paper proposes two crosstalk alleviation techniques for dynamic PLA. We first propose the concept of CT-immune lines, which are immune from crosstalk noise. Then we propose a re-ordering technique for product lines to maximize the number of CT-immune lines. Furthermore, since the crosstalk noise is also directly proportional to the parallel length, we propose another input/output lines re-ordering technique so that the parallel lengths between the non CT-immune lines and their adjacent lines are greatly reduced. Our experiment results show that the presented techniques are very effective in reducing crosstalk noise in dynamic PLA.

References

[1] G. M. Blair, “PLA Design for Single-Clock”, *IEEE J. Solid-State Circuits*, pp. 1211-1213, 1992.

[2] K. Chaudhary, A. Onozawa, and E. S. Kuh, “A Specing Algorithm for Performance Enhancement and Cross-talk reduction”, *Proc. ICCAD*, pp. 697-702, 1993.

[3] Y. B. Dhong and C.P. Tsang, “High Speed CMOS POS PLA Using Predischarged OR Array and Charge Sharing AND Array”, *Trans. Circuits and Systems, Part II*, pp. 557-564, 1992.

[4] S. P. Khatri, R. K. Brayton, and A. Sangiovanni-Vincentelli, “Cross-talk Immune VLSI Design using a Network of PLAs Embedded in a Regular Layout Fabric”, *Proc. ICCAD*, pp. 412-418, 2000.

[5] K. Kim, U. Narayanan, and S. Kang, “Domino Logic Synthesis Minimizing Crosstalk”, *Proc. DAC*, pp. 280-285, 2000.

[6] S. P. Khatri, A. Mehrotra, R. K. Brayton, and A. Sangiovanni-Vincentelli, “A Novel VLSI Layout Fabric for Deep Sub-Micron Applications”, *Proc. DA*C, pp. 491-496, 1999.

[7] S. Posluszny and et al., “Design Methodology for a 1.0 ghz Microprocessor”, *Proc. ICCD*, pp. 17-23, 1998.

[8] A. Vittal and M. Marek-Sadowska, “Crosstalk Reduction for VLSI”, *IEEE Trans. CAD*, pp. 290-298, 1997.

[9] C. Wang and et al., “A Low-Power and High-Speed Dynamic PLA Circuit Configuration for Single-Clock CMOS”, *IEEE Trans. Circuits and Systems, Part I*, pp. 857-861, 1999.

[10] J. S. Wang, C. R. Chang, and Chingwei Yeh, “Analysis and Design of High-Speed and Low-Power CMOS PLA’s”, *IEEE J. of Solid-State Circuits,* vol. 36, pp. 1250-1262, Aug. 2001.

[11] T. Xiao and M. Marek-Sadowska, “Crosstalk Reduction by Transistor Sizing”, *Proc. ASP-DAC*, pp.137-140, 1999.

Table 1 The results of product line re-ordering

Circuit	# of product lines	Original # of non CT-immune lines	# of non CT-immune lines	Computation time (sec)
alu2	68	47	14	0.03
alu3	66	46	14	0.03
b10	100	94	30	0.08
b12	43	35	15	0.01
b3	211	193	43	0.45
b9	119	52	8	0.05
bc0	179	162	57	0.23
chkn	140	87	13	0.07
ex4	279	214	26	1.01
gary	107	102	35	0.09
ibm	173	166	32	0.25
in2	136	115	34	0.11
intb	631	230	12	1.51
max1024	274	196	25	0.35
max512	145	122	24	0.11
shift	100	100	30	0.07
vtx1	110	56	10	0.05
x6dn	82	63	18	0.04
x7dn	538	492	28	2.14
x9dn	120	62	12	0.05
Ratio	1	0.7274	0.1326	

Table 2 The results of input/output line re-ordering

Circuit	The parallel length of CT-immune lines after product line re-ordering	The parallel length of CT-immune lines after I/O line re-ordering	Computation time (sec)
alu2	187	144	0.04
alu3	136	133	0.05
b10	795	615	0.12
b12	329	231	0.02
b3	2545	1634	0.85
b9	100	94	0.13
bc0	2889	1453	0.4
chkn	350	273	0.24
ex4	2752	1082	8.71
gary	1022	837	0.12
ibm	1386	904	0.8
in2	1210	773	0.19
intb	188	182	2.3
max1024	415	319	0.47
max512	368	289	0.13
shift	850	485	0.14
vtx1	272	190	0.17
x6dn	862	370	0.14
x7dn	2026	776	9.48
x9dn	332	270	0.21
Ratio	1	0.5814	