SPFD-based Wire Removal in a Network of PLAs

Sunil P. Khatri Subarnarekha Sinha Andreas Kuehlmann Robert K. Brayton

Alberto Sangiovanni-Vincentelli

Abstract

This paper describes the application of an SPFD-based wire removal technique for circuit implementations utilizing networks of PLAs. It has been shown that a design style based on a multi-level network of approximately equal-sized PLAs results in a dense, fast, and crosstalk-resistant layout. *Wire removal* is a technique where the total number of wires between individual circuit nodes is reduced, either by removing wires, or replacing them with other existing wires. The benefit of SPFD-based wire removal is shown to be insignificant when the circuit is mapped using standard cells. We demonstrate that this technique is very effective in the context of a network of PLAs. Further, we outline a technique for wire removal using multi-valued SPFDs which we expect will further improve the results.

1 Introduction

Programmable Logic Arrays (PLAs) are being rediscovered as an efficient implementation style for high-performance circuits. For example, in the recently introduced Gigahertz processor [1], performance critical parts of the control were implemented using flat PLAs. Recent work [2] demonstrates that a circuit implementation based on a network of approximately equal-sized PLAs yields a fast, compact, and cross-talk resistant design. The use of minimum-sized transistors in the PLA core results in a fast and dense layout, while a structured arrangement of wires guarantees an effective shielding among signals. The speed and area of each PLA in this design style was reported to be about 50% less than the corresponding standard-cell based implementation.

In order to reduce the area utilized by such a network, the removal of wires between individual PLAs is desired. This increases the freedom to place the PLAs and eliminates potential wire conjestion in the routing area. In this

paper, we focus on Sets of Pairs of Functions to be Distinguished (SPFDs) as a candidate technique for wire removal.

SPFDs were introduced in [3] in the context of FPGA optimization. In [4] this technique was refined and adapted to multi-level networks, while its application to logic optimization was described in [5]. The authors report a significant average wire reduction for technology-independent wire removal. However, when technology mapping is performed on the resulting circuits, the benefits of wire removal are erased.

In this work, we apply the wire removal algorithm of [5] to a network of PLAs, and demonstrate an approximate 20% reduction in wiring, which directly translates into a reduction of the layout area. This is because a separate technology mapping step is not required when the circuit is implemented as a network of PLAs. Further, we outline a new multi-valued SPFD computation that can be employed to perform wire removal in this context. This idea is motivated by the observation that the modeling of multi-output PLAs as multi-valued functions provides additional flexibility to optimize them.

The organization of the rest of this paper is as follows: In Section 2, we briefly describe the implementation style using a network of PLAs. Section 3 introduces Multi-valued SPFDs, while Section 4 outlines our multi-valued SPFD based technique for wire removal. Section 5 describes the wire replacement experiments we performed. Finally, Section 6 concludes the paper and gives an outline of future work.

2 Networks of PLAs

In [2], a new layout and design methodology was introduced, motivated by the goal to achieve fast and dense designs which are are not susceptible to cross-talk, an increasingly important design consideration in deep sub-micron (DSM) technologies. The circuit being implemented was decomposed into a network of medium-sized PLAs, each with between 5 and 10 inputs or outputs, and in the order of 20 product terms. It was shown that this size range for the PLAs constituted an optimal design point with respect to speed and density. Such PLAs were typically 50% faster, and about 40% smaller than a comparable standard-cell based implementation. A simple greedy algorithm was introduced

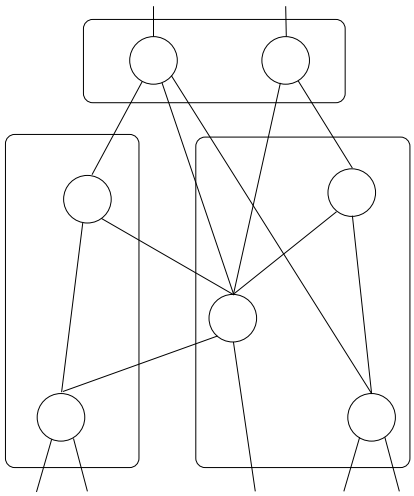


Figure 1: Multi-level circuit decomposed into a network of PLAs.

to decompose a multi-level circuit into a network of PLAs.

A sample multi-level circuit, with nodes shown as circles, is shown in Figure 1. The rectangular regions in this figure represent the clustering of circuit nodes into PLAs.

The layout of a circuit implemented in the network of PLAs style is shown in Figure 2. The dark rectangles in this figure correspond to the PLAs in the design.

3 Multi-valued SPFDs

Definition 1 *An* **SPFD** $\mathcal{F}(y)$ *on a domain* Y *is an undirected graph* (V,E) *where each* $v \in V$ *corresponds to a unique minterm* $v = (y_1, y_2, \cdots, y_k) \in Y$. *An edge* $(e = (v_1, v_2)) \in E$ *means that the minterms corresponding to the two vertices* v_1 *and* v_2 *must have different functional values.*

Figure 3 shows a multi-valued node H with k values, and its corresponding (MV-)SPFD. This SPFD is a set with k tuples $\{H_0, H_1, \cdots, H_{k-1}\}$. Each tuple H_i consists of several minterms $\{h_1^i, h_2^i, \cdots, h_{n_i}^i\}$, where $n_i \leq k$. Each minterm in H_i must be distinguished from (i.e. have different functional values than) minterms in each of the remaining $k-1$ tuples.

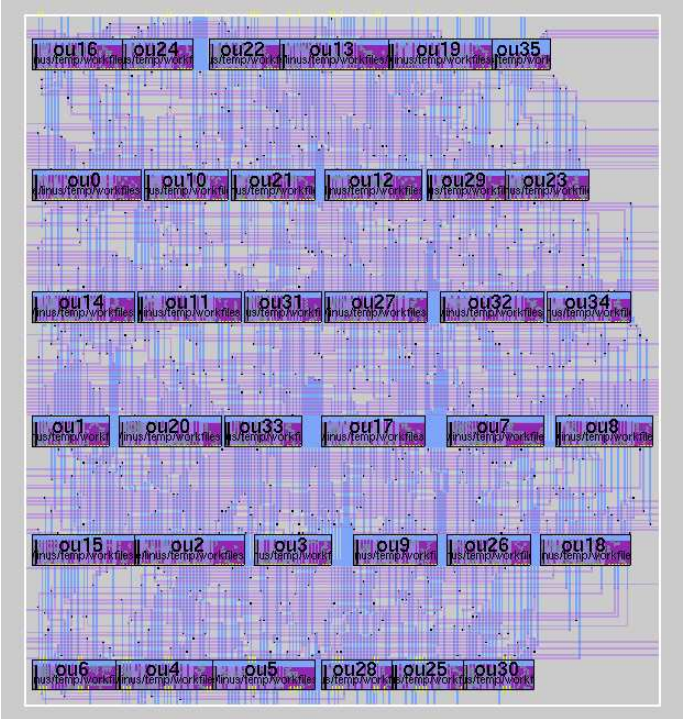


Figure 2: Sample Layout using Network of PLAs

a b c f

0 0 0 1 $\leftarrow h_0^1$
0 0 1 2 $\leftarrow h_0^2$
0 1 0 2 $\leftarrow h_1^2$
0 1 1 0 $\leftarrow h_0^0$
1 0 0 1 $\leftarrow h_1^1$
1 1 0 2 $\leftarrow h_3^2$
1 0 1 2 $\leftarrow h_2^2$
1 1 1 0 $\leftarrow h_1^0$

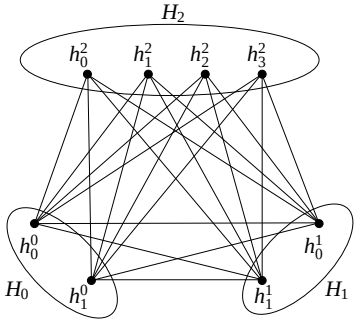


Figure 3: A Multi-valued SPFD.

Note that this definition of an SPFD is a Multi-valued generalization of the definition of [5]. For a detailed exposition of binary valued SPFDs, and how they can be used for network optimization, the reader is referred to [5].

Definition 2 *A function $F(y)$* **implements** *$\mathcal{F} = (\mathcal{V}, \mathcal{E})$ if $F(y)$ is a valid coloring of $\mathcal{F}$, i.e.*

$$F(y^1) \neq F(y^2), (y^1, y^2) \in E$$

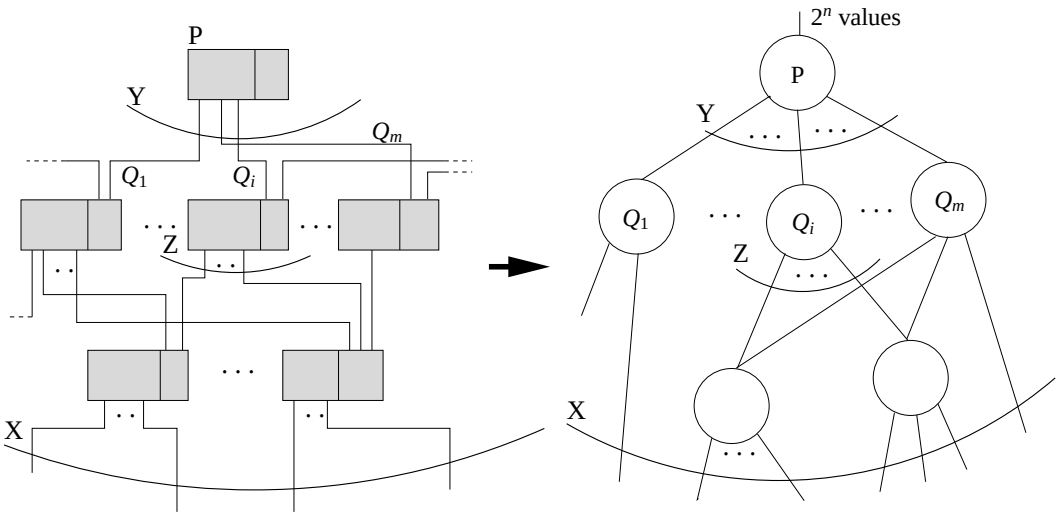
In other words, for a function F to satisfy $\mathcal{F}$, F assigns a different value to functions f_p^i and f_q^j, for $i \neq j$. This suggests that the *chromatic number* of an SPFD is minimum the number of values required to implement the SPFD using a multi-valued function.

4 Wire Removal using Multi-valued SPFDs

In the network of PLAs, each individual PLA is a multi-output, multi-input structure. Suppose a given PLA has k outputs. Then, it can be viewed as a single output node with 2^k values. At this point, a multi-valued SPFD can be computed for this node, and wires in its fanin can be removed using a SPFD based computation. This computation is described below.

Given a wire, (i, j), its SPFD represents the pairs of minterms that have to be distinguished by it. Thus, in a sense, the SPFD of (i, j) encodes the information content of a wire. We can replace a wire by another as long as the second wire has all the information of the original wire. A wire (s, j) can replace the wire (k, j) if all the minterms uniquely distinguished by the wire (k, j) are also distinguished by (s, j). In other words, the objective is to replace wire (k, j) from node n_k to n_j with a wire (s, j) from node n_s to n_j, such that the original SPFD at n_j is preserved, and some gain is realized by this change.

The procedure for removing wires in a PLA network is explained below. Consider the PLA P, which has m inputs and n outputs. Figure 4-a shows a sample network of PLAs, in which P resides. Each rectangle represents a PLA, with



a) A network of PLAs b) Its corresponding multi-valued network

Figure 4: Multi-valued SPFD based wire removal.

its AND (input) plane on the left, and the OR (output) plane on the right. The PLA *P* can equivalently be considered a multi-valued node with 2^n values, and *m* binary-valued inputs, as shown in Figure 4-b.

After computing the $\mathcal{P}(Y)$ of *P*, (here *Y* is the space of the fanins of *P*) we re-assign the task of distinguishing edges of $\mathcal{P}(Y)$ to the fanins of *P*, using the following procedure.

- Fanins of *P* that exclusively distinguish some edge of $\mathcal{P}(Y)$ are first identified, and the corresponding edges are assigned to these fanins.

- The remaining edges $\mathcal{P}(Y)$ are then assigned to the fanins of *P*. As far as possible, edges are assigned to fanins that already have SPFD edges assigned to them, until no more edges can be assigned.

- Of the remaining edges, consider those that can be distinguished by two fanins. Of these fanins, assign as many edges as possible to the fanin that distinguishes most remaining edges.

- Next consider the edges that can be distinguished by three fanins. Of these fanins, assign as many edges as possible to the fanin that distinguishes most remaining edges.

- Proceed in this fashion until no edges remain.

After this computation, if there are fanins that have no edges to distinguish, these can be removed, and the network can be accordingly simplified. The new fanin space of P is called $\hat{Y}$. The remaining fanins are re-implemented, by computing the image of their new SPFDs in the input space or a space corresponding to an intermediate cut of the network (we call this space X), and projecting this image back to the space of their fanins, Z. Now *espresso* [6] is called to get the new function at the fanin.

In the PLA context, re-implementing any fanin of P means modifying that specific output of the PLA Q, which is an input to P. This would in general result in a new implementation of Q. The only constraint is that the total number of product terms of Q after this modification is still bounded by the upper limit on product terms.

Now, P is modified. First the image of $\mathcal{P}(Y)$ is computed on the X space. This image is projected back to the $\hat{Y}$ space, to get $\hat{\mathcal{P}}(\hat{Y})$, the new SPFD of P in terms of its new fanins. Now we can simply run espresso on the new SPFD to get the new function of P.

The above method for performing wire removal is effective for the following reasons:

- It was observed [5] that whenever the nodes were simple, SPFD based optimizations resulted in little improvement. The multi-valued nodes corresponding to the PLAs are complex, since they usually constitute several inputs and outputs. As a result, the flexibility offered by SPFDs can be exploited to the fullest.

- In an SPFD-based computation, a logic node and its fanins are optimized simultaneously. When the fanins of a node F are modified, the logic function of F needs to be changed as well. In [5], to avoid the propagation of changes throughout the transitive fanout of F, the CODCs (compatible output don't-cares) [7] of the immediate fanouts of F are used to *block* the changes of F. For the application of MV-SPFDs in PLA-based wire removal, the changes to any node (i.e. a PLA) F do not need to be blocked by the don't cares of nodes in the fanout of F. This is because fanout PLAs can be easily re-implemented if F changes, as long as the total number of product terms in the fanout PLAs are bounded. This is expected to result in significantly more flexibility while optimizing any given PLA.

Circuit	Standard Cell			Network of PLAs		
	before WR	after WR	Ratio	before WR	after WR	Ratio
C432	1.063	1.042	0.980	1.414	1.217	0.861
C499	2.578	2.617	1.015	2.200	1.917	0.871
C880	2.460	2.593	1.054	3.809	2.876	0.755
rot	7.591	7.416	0.977	7.782	4.453	0.572

Table 1: Wire Removal without Prior Optimization

Circuit	Standard Cell			Network of PLAs		
	before WR	after WR	Ratio	before WR	after WR	Ratio
C432	0.906	0.998	1.102	1.403	1.299	0.926
C499	2.484	2.422	0.975	2.533	2.016	0.796
C880	2.515	2.456	0.977	2.082	1.730	0.831
rot	6.659	6.413	0.963	5.524	4.250	0.769

Table 2: Wire Removal after *script.rugged*

5 Experimental Results

In our experiments to validate the usefulness of SPFD-based wire removal for a network of PLAs, we utilize the *wire_replace* code which was described in [5]. The computation is done at the level of binary-valued SPFDs[1]

Table 1 reports the results of *wire_replace* on unoptimized circuits. In our tables, the final layout area of the circuit is measured in units of 10^6 square grids. All reported numbers include the area for the actual logic as well as routing.

Columns 2 and 3 report the results for a standard-cell based implementation, with and without *wire_replace*, respectively. Column 4 reports the ratio of the standard cell area after wire removal, to the area before wire removal. Columns 5 and 6 report the results for a PLA based implementation, with and without *wire_replace*, respectively. Column 7 reports the ratio of the PLA based area after wire removal, to the area before wire removal. Table 2 is organized in the same fashion, except that each circuit was first subjected to *script.rugged*, a technology-independent optimization script in SIS [8].

In essence, we note that for the standard-cell based methodology, *wire_replace* does not impact the overall layout area. This is because the benefits attained through wire removal at the technology independent level are negated by the technology mapping step. However, in the case of the network of PLAs, *wire_replace* results in a significant reduction

[1] It is expected that we will have significantly better results when performing wire removal using MV-SPFDs as described in Section 4, since the changes to a MV node (i.e. PLA) do not need to be blocked by the Don't Cares of its fanouts, and also because the MV nodes are more complex, allowing greater optimizational flexibility

in circuit area (23.5% for the non-optimized case, and 16.9% in the optimized case). This is due to the absence of a technology mapping step after wire removal. As a result, the benefits of wire removal are directly translated into a reduction in circuit area.

6 Conclusions and Future Work

In this paper we have demonstrated that SPFD based wire removal is a powerful technique for reducing the wiring, and therefore the overall layout area, of a circuit implemented as a network of PLAs. We show that the binary-valued wire removal algorithm of [5] provides a 20% reduction in wiring for a network of PLAs, while the same algorithm delivers no improvement for a standard-cell based implementation.

We have also outlined a multi-valued SPFD computation to perform wire removal for a network of PLAs. We plan to implement this in the future. It is expected to provide yet better results, since the changes in a MV node (i.e. a PLA) do not need to be blocked by the don't-cares of fanout nodes. Also, since each of the MV nodes are complex, we expect that then MV-SPFD based algorithm will have a larger flexibility in re-implementing an MV node.

References

[1] S. Posluszny, N. Aoki, D. Boerstler, J. Burns, S. Dhong, U. Ghoshal, P. Hofstee, D. LaPotin, K. Lee, D. Meltzer, H. Ngo, K. Nowka, J. Silberman, O. Takahashi, and I. Vo, "Design methodology for a 1.0 ghz microprocessor," in *Proceedings of the International Conference on Computer Design (ICCD)*, pp. 17–23, Oct 1998.

[2] S. Khatri, "A novel VLSI layout and design flow utilizing regular layout fabrics." Internal Study, University of California at Berkeley, Feb 1999.

[3] S. Yamashita, H. Sawada, and A. Nagoya, "A new method to express functional permissibilities for LUT based FPGAs and its applications," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 254–61, Nov 1996.

[4] R. Brayton, "Understanding SPFDs: A new method for specifying flexibility," in *Workshop Notes, International Workshop on Logic Synthesis*, (Tahoe City, CA), May 1997.

[5] S. Sinha and R. Brayton, "Implementation and use of SPFDs in optimizing boolean networks," in *Proceedings of the International Conference on Computer-Aided Design*, pp. 103–10, Nov 1998.

[6] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli, *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Publishers, 1984.

[7] H. Savoj, *Don't Cares in Multi-Level Network Optimization.* PhD thesis, University of California Berkeley, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA 94720, May 1992.

[8] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," Tech. Rep. UCB/ERL M92/41, Electronics Research Lab, Univ. of California, Berkeley, CA 94720, May 1992.