BINARY AND MULTI-VALUED SPFD-BASED WIRE REMOVAL IN PLA NETWORKS

by

Sunil P. Khatri, Subarnarekha Sinha, Robert K. Brayton
And Alberto Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M99/51

29 October 1999

COVER

BINARY AND MULTI-VALUED SPFD-BASED WIRE REMOVAL IN PLA NETWORKS

by

Sunil P. Khatri, Subarnarekha Sinha, Robert K. Brayton
and Alberto Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M99/51

29 October 1999

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Binary and Multi-valued SPFD-based Wire Removal in PLA Networks

Sunil P. Khatri Subarnarekha Sinha Robert K. Brayton

Alberto Sangiovanni-Vincentelli

CAD Research Group, U.C. Berkeley

Abstract

This paper describes the application of binary and multi-valued SPFD-based wire removal techniques 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. Three separate wire removal experiments are performed. Either wire removal is invoked before clustering the original netlist into a network of PLAs, or after clustering, or both before and after clustering. For wire removal before clustering, binary SPFD-based wire removal is used. For wire removal after clustering, multi-valued SPFD-based wire removal is used since the multi-output PLAs can be viewed as multi-valued single output nodes. We demonstrate that these techniques are effective. The most effective approach is to perform wire removal both before and after clustering. Using these techniques, we obtain a reduction in placed and routed circuit area of about 11%. This reduction is significantly higher (about 20%) for the larger circuits we used in our experiments.

1 Introduction and Previous Work

Programmable Logic Arrays (PLAs) are being rediscovered as an efficient implementation style for high-performance circuits. For example, in the Gigahertz processor [1], performance-critical parts of the control were implemented using single 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 effective. This increases the freedom to place the PLAs and eliminates potential wire congestion 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 of [5] reported a significant average wire reduction for technology-independent wire removal. However, when technology mapping was performed on the resulting circuits, the benefits of wire removal were erased. In [6], binary SPFD-based wire removal was applied to a network of PLAs. However, this work did not utilize the power of multi-valued SPFDs for the task. Also, results were reported on a small set of benchmark circuits.

In our work, we perform both binary SPFD based wire removal as well as multi-valued SPFD based wire removal. Binary SPFD based wire removal is done in the manner described in [5]. This flavor of wire removal is performed before clustering the circuit into a network of PLAs.

In addition, we generalize the notion of SPFDs to multi-valued networks. We observe that (multi-output) PLAs can be modeled as multi-valued functions. Hence a network of PLAs can be modeled as a multi-level network with multi-valued nodes. We extend the binary wire removal technique described in [5] to the multi-valued case, and use this idea to perform wire removal for a network of PLAs. This flavor of wire removal is performed after the clustering of a circuit into a network of PLAs. We also observe that since each multi-valued node is more complex than the binary nodes encountered in [5], additional flexibility is obtained in optimizing them, as evidenced by our results. Although the full flexibility of multi-valued wire removal has not been exploited in our work, we still get good reductions in layout area.

The organization of this paper is as follows: In Section 2, we describe the circuit implementation style using a network of PLAs. Section 3 describes binary SPFDs and their use in removing wires in binary networks. Section 4

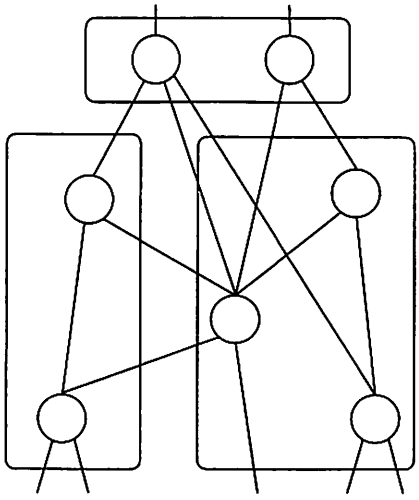


Figure 1: Multi-level circuit clustered into a network of PLAs.

introduces multi-valued SPFDs, while section 5 outlines our multi-valued SPFD based technique for wire removal. Section 6 describes the wire removal experiments performed. Finally, Section 7 concludes the paper and gives some directions for future work in this area.

2 Networks of PLAs

In [2], a new layout and design methodology was introduced, motivated by the goal of achieving fast and dense designs immune to cross-talk, an increasingly important design consideration in deep sub-micron (DSM) technologies. The circuit being implemented was clustered into a network of medium-sized PLAs, each with between 5 and 10 inputs or outputs, and approximately 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 to cluster 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.

3 Binary Sets of Pairs of Functions to be Distinguished

3.1 Definitions

Sets of Pairs of Functions to be Distinguished (SPFDs) are a new way to represent the flexibility of a node in a multi-level network. In this section we focus on SPFDs for binary valued nodes.

Definition 1 *A function f is said to* **distinguish** *a pair of functions g_1 and g_2 if either one of the following two conditions is satisfied:*

$$g_1 \leq f \leq \overline{g}_2 \tag{1}$$

$$g_2 \leq f \leq \overline{g}_1 \tag{2}$$

Note that this definition is symmetrical between g_1 and g_2. We can think of 1 and 2 specifying two incompletely specified functions, with g_1 as the onset and g_2 as the offset in 1 or vice-versa for 2.

Definition 2 *An* **SPFD**

$$\{(g_{1a}, g_{1b}), \ldots, (g_{na}, g_{nb})\}$$

is a set of pairs of functions to be distinguished.

Definition 3 *A function f* **satisfies** *an SPFD, if f distinguishes each pair of the set, i.e.*

$$[((g_{1a} \leq f \leq \overline{g}_{1b}) + (g_{1b} \leq f \leq \overline{g}_{1a})] \wedge \ldots \wedge$$

$$[(g_{na} \leq f \leq \overline{g}_{nb}) + (g_{nb} \leq f \leq \overline{g}_{na})]$$

Hence, an SPFD can be conveniently used to express the flexibility that can be used to implement a node in a network - the only condition required is that the function implemented at the node satisfies its node SPFD. Note that vertices

of a node's SPFD correspond to the on-set, off-set or dont-care minterms of the node function.

A trivial case is where the set is a single pair. In this case the SPFD represents two incompletely specified functions (ISF) where one is the complement of the other. If each of the $\{(g_{1a}, g_{1b}), (g_{2a}, g_{2b}), \ldots, (g_{na}, g_{nb})\}$ are pairwise disjoint, then the SPFD represents 2^n ISFs[1].

Classically, in computing the flexibility at a node in a Boolean network, don't cares are computed which represent a single ISF. These computations can be generalized so that SPFDs are obtained, which provide much more freedom in optimizing the node.

3.2 Wire Removal/Replacement Using SPFDs

The information content of a wire (which is effectively the set of pairs of minterms it can distinguish) in a network can be effectively represented by an SPFD. This allows SPFDs to help remove certain "difficult" wires in the network or to replace them by other wires. The technique of wire removal/replacement using SPFDs works as follows.

Consider a multi-level network, with some nodes η_i, η_j and η_k. Given a wire (η_i, η_j), its SPFD represents the pairs of minterms that have to be distinguished by it. Thus, in a sense, the SPFD of (η_i, η_j) encodes the information content required of that wire. If the wire (η_i, η_j) need not uniquely distinguish any minterms i.e. it has no unique information content required, we can remove it. We can also try to replace it by another wire as long as the second wire has all the information required of the original. So, a wire (η_s, η_j) can replace the wire (η_k, η_j) if all the minterms required to be distinguished by the wire (η_k, η_j) are also distinguished by (η_s, η_j). In other words, the objective is to replace wire (η_k, η_j) from node η_k to η_j with a wire (η_s, η_j) from node η_s to η_j, such that the original SPFD at η_j is preserved, and some gain is realized by this change. In the sequel, we shall refer to this technique as *wire_replace*. In [5], it was shown that there can be a substantial reduction in the number of wires (at the technology-independent level) in the network using the *wire_replace* algorithm. Note that *wire_replace* also removes wires whose SPFDs are empty.

[1]Note that an SPFD cannot represent a single function, it always represents at least a pair. Thus it cannot represent the function 1.

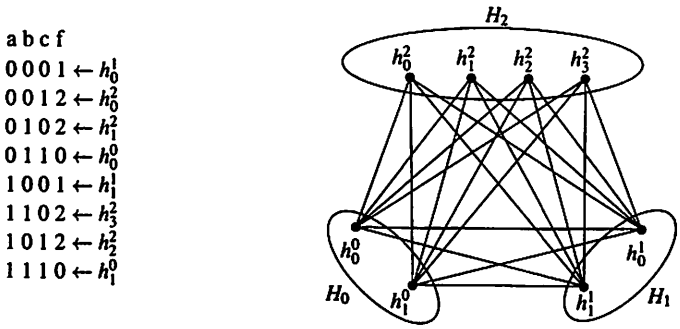


Figure 2: A Multi-valued SPFD.

For a detailed exposition on SPFDs and how they are computed and used for wire replacement, see [5].

4 Multi-valued SPFDs

We give a graph-theoretic definition of MV-SPFDs which is a generalization of the definition of binary SPFDs of the previous section.

Definition 4 *An* **MV-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 2 shows a multi-valued node H with k values, and its corresponding MV-SPFD. This MV-SPFD can be described as a set with k tuples $\{H_0, H_1, \cdots, H_{k-1}\}$. Each tuple H_i consists of several minterms $\{h_1^i, h_2^i, \cdots, h_{n_i}^i\}$. 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. Each H_i is also referred to as a *component.*

Definition 5 *A function* $F(y)$ **implements** $\mathcal{F} = (\mathcal{V}, \mathcal{E})$ *if* $F(y)$ *is a valid coloring of* $\mathcal{F}$, *i.e.*

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

For a function F to implement $\mathcal{F}$, F assigns a different value to minterms h_p^i and h_q^j, for $i \neq j$. Thus the *chromatic number* of an MV-SPFD is the minimum number of values required to implement the MV-SPFD using a multi-valued

function. Each different coloring of this graph represents a different incompletely specified multi-valued function (ISF). This is the source of flexibility of MV-SPFDs.

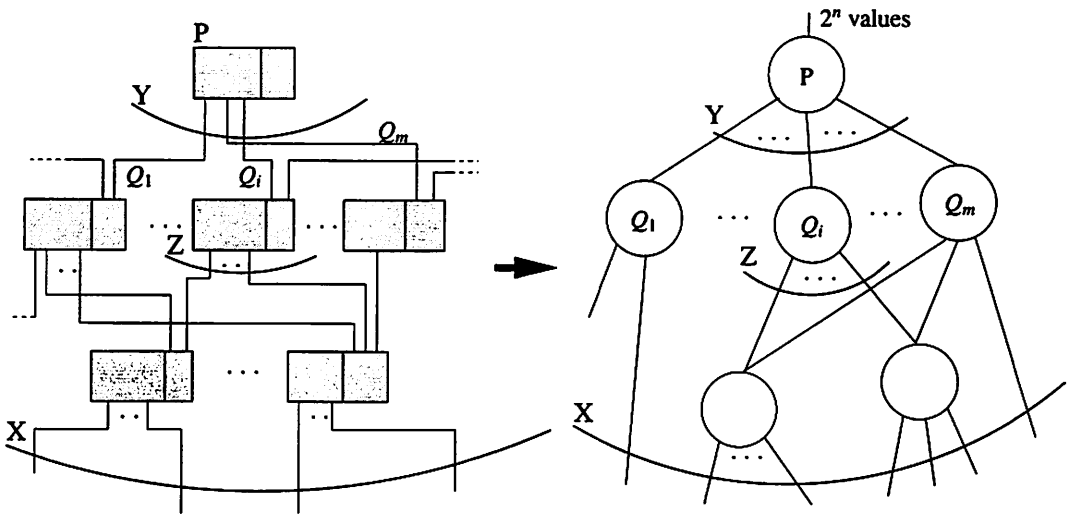
5 Wire Removal using Multi-valued SPFDs

In a network of PLAs, each individual PLA is a multi-input, multi-output structure. Suppose a given PLA has k outputs. In that case, it can be modeled as a single output node with 2^k values. A multi-valued SPFD can be computed for each node and can be used to remove wires in its fanin. A network of PLAs can be modeled as a multi-level network of multi-valued nodes. The binary SPFD techniques for computing and distributing SPFDs using BDDs [7] can be generalized to MV-SPFD techniques using MDDs [8]. The details of the computation are discussed below.

Consider a node η_j in a multi-level, multi-valued logic network. We know that the MV-SPFD of η_j represents the set of multi-valued minterms (henceforth equivalently referred to as minterms) that should be distinguished by η_j in order that it provides enough information to its fanouts. To achieve this, it is necessary that each pair of minterms in the MV-SPFD of η_j be distinguished by at least one of its fanin wires. Thus, the union of the MV-SPFDs of its fanin wires should cover the MV-SPFD of η_j. Alternately stated, just as in the binary case, the MV-SPFD of a node/wire gives the information content required of the node/wire. So, all the information contained in a node has to be provided by its fanins.

We define the **minimum MV-SPFD** of a wire (η_i,η_j) to be the set of pairs of minterms of η_j that must be distinguished exclusively by this wire. In order to ensure that all the pairs of minterms in the MV-SPFD of η_j are distinguished, the wire (η_i,η_j) must distinguish at least these pairs of minterms.

Given the MV-SPFD of the node η_j, we compute the minimum MV-SPFD of each fanin wire. If the minimum MV-SPFD of a fanin wire is not empty, then we cannot remove this wire since it uniquely distinguishes some pair of minterms in the MV-SPFD of the node η_j. On the other hand, if the MV-SPFD of a fanin wire is empty, it is a candidate for removal. However, we cannot simultaneously remove some or all fanin wires whose minimum MV-SPFDs are



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

Figure 3: Multi-valued SPFD based wire removal.

empty. This is because there could be two fanin wires (η_i, η_j) and (η_k, η_j) with empty minimum MV-SPFDs, such that both wires distinguish the pair of minterms (m_1, m_2) in the MV-SPFD of η_j, and no other fanin wire distinguishes this pair of minterms. In such a situation, at least one of these wires must be retained. If both wires are removed, (m_1, m_2) will not be included in the new MV-SPFD of η_j, and hence the resulting network will not be correct.

The procedure for removing wires in a PLA network is explained below. Consider the PLA P, which has m inputs and n outputs. Figure 3-a shows a sample network of PLAs, in which P resides. Each rectangle in this figure represents a PLA, with its AND (input) plane on the left, and the OR (output) plane on the right. The PLA P can be considered equivalently as a multi-valued node with 2^n values, and m multi-valued inputs, as shown in Figure 3-b.

The MV-SPFD of P, denoted as $\mathcal{P}(Y)$, is computed from its original multi-valued function (MVF). This MV-SPFD must distinguish every minterm in every component of its MVF from every minterm in every other component of its MVF. After computing $\mathcal{P}(Y)$, (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 have non-empty minimum MV-SPFDs, denoted as Y', are first identified.

- All the edges of $\mathcal{P}(Y)$ that are distinguished by these fanins are assigned to these fanins and are removed from

$\mathcal{P}(Y)$.

- A weighted covering problem is set up between the remaining fanins of P, $Y \setminus Y'$, and the edges of the modified $\mathcal{P}(Y)$. The fanins are weighted according to the following heuristic : the smaller the number of fanouts of a particular fanin, the greater its weight. This means that a fanin with a single fanout has the largest weight and so has the least likelihood of being included in the solution. Hence the corresponding wire is most likely to be removed. Let the solution of this weighted covering problem be Y''.

The new fanin space of P is the union of Y' and Y'' and will be subsequently referred to as $\hat{Y}$. Now, P is modified. First the image of $\mathcal{P}(Y)$ is computed on the primary input space X. 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. We use a coloring algorithm to obtain a new ISF at P. The connected components of the MV-SPFD are obtained and each component is colored appropriately to obtain a new ISF. Next we run Espresso-MV [9] to get the new minimized function of P.

We proceed in a topological order from the inputs to the outputs in the network and perform wire removal on each node in the network.

In the sequel, we refer to this algorithm as *mv_wire_replace*.

5.1 Controlling change

As mentioned in the previous section, any valid coloring of $\mathcal{P}(Y)$ can be used to obtain an incompletely specified MV function for P. But, if a node is changed, then its changes must be propagated throughout the transitive fanout of P. Although this can be done, in practice it can prove to be expensive. So we block the changes in the new function by its MV-CODCs [10] (a generalization of CODCs [11] for the multi-valued case). Thus, at any point in the algorithm, the *region of change* consists of a single node, and possibly its immediate fanins.

5.2 Multi-valued SPFDs vs binary SPFDs

Although MV-SPFDs are a generalization of binary SPFDs, there are some interesting points that they bring up.

- In [5], the authors discuss how binary SPFDs could run into the problem of non-bipartition i.e. there could arise situations where an SPFD can no longer be colored by two colors. This situation arises because during the SPFD computation, one cannot restrict the chromatic number of an SPFD without losing optimization flexibility. Since some binary SPFD algorithms exploit their bipartite nature, this could lead to inelegant solutions. MV-SPFDs are a simple and elegant way to handle this problem, since we do not need to restrict the chromatic number of an MV-SPFD.

- The coloring of MV-SPFDs gives rise to interesting possibilities. A binary SPFD with n connected components can be colored in 2^n ways. An MV-SPFD with n components can be colored in $k_1! * \cdots * k_n!$, where k_i is the chromatic number of the ith component. This flexibility can be exploited in many ways. In a network of PLAs, for instance, re-encoding a node could change the wiring connections between a node and its fanouts. So, if we expand the *region of change* to include a node *and* its fanouts, we can use some encoding algorithm to suitably modify the wiring between a node and its fanouts. This is a difficult problem and is currently being investigated.

6 Experimental Results

To validate the usefulness wire removal for a network of PLAs, we utilize the two SPFD-based wire removal techniques.

- For wire removal before clustering a circuit into a network of PLAs, we use the *wire_replace* code detailed in [5] and in Section 3.2. This computation is done at the level of binary-valued SPFDs, since the logic nodes are binary valued before clustering into PLAs.

- After clustering into a network of PLAs, each PLA can be viewed as a multi-valued node, as described in Section 5. At this point, multi-valued SPFD-based wire removal is invoked, using *mv_wire_replace* as described in Section 5.

The clustering and wire removal code was written in SIS [12]. Placement of the network of PLAs was done using VPR [13], an FPGA-based placement and routing tool. Since all PLAs in the network of PLAs have roughly the same size, VPR is a good choice for placement. However, routing is not done using VPR since it assumes an FPGA connection topology. Therefore, routing of the network of PLAs was performed using *wolfe* [14].

The initial *blif* netlist for the benchmark circuit is clustered into nodes with up to 5 inputs. This new netlist is the starting point for all wire removal experiments. We now perform one of 4 wire removal experiments:

- For *no wire removal*, (NOWR) we cluster the netlist into a network of PLAs. This network is now placed and routed as described above.

- For *wire removal after clustering*, (WRA) we follow the clustering step by a wire removal step, using multi-valued SPFD-based wire removal. The result of this step is then placed and routed.

- For *wire removal before clustering*, (WRB) we perform binary-valued SPFD-based wire removal on the netlist, and then cluster the resulting netlist into a network of PLAs. This network is then placed and routed.

- For *wire removal before and after clustering*, (WRBA) we perform binary-valued SPFD-based wire removal on the netlist, and then cluster the resulting netlist into a network of PLAs. This is followed by multi-valued SPFD-based wire removal. The resulting netlist is placed and routed as described above.

We constrain the clustering step by imposing a maximum width and maximum height constraint on the PLAs. In this section we report the results of experiments with two such combinations which utilize a PLA height constraint of 15 and 20, and a PLA width constraint of 40. The total number of outputs of each PLA is constrained to be no larger than 5.

Table 1 reports the results of wire removal on some benchmark circuits. All examples in this table use a PLA height constraint of 15, and a PLA width constraint of 40. Table 2 reports the results of wire removal where all examples use a PLA height constraint of 20 and a PLA width constraint of 40. Each PLA has 5 or less outputs in both

Circuit	NOWR	WRA	Improve %	WRB	WRBA	Improve %	NOWR-WRA%	NOWR-WRB%	NOWR-WRBA%	BEST
vda	11110252	9226156	16.96	10135268	8894264	12.24	16.96	8.78	19.95	19.95
frg2	6446700	5696732	11.63	5963900	5450492	8.61	11.63	7.49	15.45	15.45
C1908	5638308	5608980	0.52	4577608	4328608	5.44	0.52	18.81	23.23	23.23
apex6	4449572	4038000	9.25	4406096	4250136	3.54	9.25	0.98	4.48	9.25
x3.blif	4423560	4340580	1.88	4588100	4409520	3.89	1.88	-3.72	0.32	1.88
toolarge	4306372	4296808	0.22	4398404	4355300	0.98	0.22	-2.14	-1.14	0.22
x1	2046140	1874444	8.39	1805672	1859528	-2.98	8.39	11.75	9.12	11.75
x4	2040556	2127220	-4.25	1938784	1938784	0.00	-4.25	4.99	4.99	4.99
alu2	1624412	1473920	9.26	1647092	1629860	1.05	9.26	-1.40	-0.34	9.26
C432	1372412	1345680	1.95	1227940	1236696	-0.71	1.95	10.53	9.89	10.53
term1	1252120	1052248	15.96	960876	878036	8.62	15.96	23.26	29.88	29.88
apex7	991728	853688	13.92	952448	923612	3.03	13.92	3.96	6.87	13.92
ttt2	529568	448240	15.36	519304	508664	2.05	15.36	1.94	3.95	15.36
count	364000	315832	13.23	367920	315832	14.16	13.23	-1.08	13.23	13.23
pcle	272392	264808	2.78	272392	264808	2.78	2.78	0.00	2.78	2.78
decod	175192	175192	0.00	175192	175192	0.00	0.00	0.00	0.00	0.00
AVERAGE			7.32			3.92	7.32	5.26	8.92	11.35

Table 1: Wire Removal Experiments - max width 40, max height 15

cases. In both tables, the final layout area of the circuit is measured in units of square grids. All reported numbers include the area for the actual PLA logic plus the routing area. For each table, the first column reports the circuit name. The second column reports the resulting layout area using no wire removal (NOWR), while the third column reports layout area using MV-SPFD based wire removal after clustering the circuit into a network of PLAs (WRA). The fourth column reports the improvement in layout area by performing WRA (compared to the NOWR case). The fifth column contains layout area results when binary-valued SPFD based wire removal is performed before clustering into a network of PLAs (WRB). The sixth column reports layout area when SPFD based wire removal is performed both before and after clustering into a network of PLAs (WRBA). The seventh column reports the area improvement of the sixth column over the fifth. The eighth, ninth and tenth columns represent the percentage area improvements of WRA, WRB and WRBA over the NOWR case, respectively. Finally, the eleventh column represents the best area improvement from the preceding three columns.

We observe that the best area reduction using any flavor of wire removal is above 11% for both tables. Also note that the best area reduction is in excess of 19% for the three largest examples. This suggests that SPFD-based wire removal is very effective for larger circuits.

Circuit	NOWR	WRA	Improve %	WRB	WRBA	Improve %	NOWR-WRA%	NOWR-WRB%	NOWR-WRBA%	BEST
vda	12436252	10270940	17.41	11312344	9509604	15.94	17.41	9.04	23.53	23.53
frg2	5421528	4817176	11.15	5504856	5073732	7.83	11.15	-1.54	6.42	11.15
C1908	6681500	5834776	12.67	5179324	4491136	13.29	12.68	22.48	32.78	32.78
apex6	4856400	4692516	3.37	4773860	4456872	6.64	3.38	1.70	8.23	8.23
x3.blif	4992788	4487352	10.12	4655196	4745676	-1.94	10.12	6.76	4.95	10.12
toolarge	4714168	4681740	0.69	4670440	4679008	-0.18	0.69	0.93	0.75	0.93
x1	2110856	2098096	0.60	2069732	2103604	-1.64	0.60	1.95	0.34	1.95
x4	2061840	2028516	1.62	2158952	2230680	-3.32	1.62	-4.71	-8.19	1.62
alu2	1706600	1591744	6.73	1827148	1543940	15.50	6.73	-7.06	9.53	9.53
C432	1556960	1466576	5.81	1325088	–	–	5.81	14.89	–	14.89
term1	1300096	1126408	13.36	939400	858520	8.61	13.36	27.74	33.97	33.97
apex7	1030580	971280	5.75	1077320	1026528	4.71	5.75	-4.54	0.39	5.75
ttt2	599540	523240	12.73	595232	573344	3.68	12.73	0.72	4.37	12.73
count	483360	406192	15.96	483360	416368	13.86	15.97	0.00	13.86	15.97
pcle	310312	302728	2.44	310312	310312	0.00	2.44	0.00	0.00	2.44
decod	204472	204472	0.00	204472	204472	0.00	0.00	0.00	0.00	0.00
AVERAGE			7.53			5.19	7.53	4.27	8.18	11.60

Table 2: Wire Removal Experiments - max width 40, max height 20

Comparing the wire removal techniques in isolation, we observe that WRBA provides the best average improvement in area (8.92% and 8.18% for Table 1 and Table 2 respectively). In both these tables, WRBA improves on WRB by an average of 3.92% and 5.19% respectively. The least effective of the three wire removal flows is WRB.

Furthermore, the results reported in [6] indicated that wire removal applied to traditional standard-cell based designs results in no area improvement, since wire removal obtained by such techniques is negated by the technology mapping step required in such a design style. This suggests that using a network-of-PLAs design methodology has additional advantages over the standard-cell based design methodology. The reason for this is that in the network-of-PLAs design style, there is a more direct relationship between the cost function being optimized during synthesis, and the actual implementation of the logic. This is because there is no technology-mapping step required in this design style.

Among the three wire removal experiments conducted, the most effective are WRBA and WRA. These two experiments together contributed to a majority of the best case results (column 11). In Table 1, in the cases in which WRB contributed the best result, either WRA or WRBA had improvements very close to this. For the C432 example in Table 2, WRB contributed the best result, and the improvement provided by WRA trailed it significantly. However, WRAB was not able to complete on this example, so we are not sure if WRAB could have matched this result if the

example had completed.

We performed another study where all four experiments used a series of 9 values of maximum PLA height and width. The maximum height varied from 15 to 25 in steps of 5, and the maximum width varied from 40 to 60 in steps of 10. The maximum number of outputs was restricted to 5. We used the best area from each of these 9 cases for each example, and compared the results just as in the tables above. The results obtained were substantially similar to those reported in Tables 1 and 2. This is primarily due to the fact that the two combinations of maximum width and height used in Tables 1 and 2 accounted for the best results for most examples. In this study, the average best case area improvement due to any flavor of wire removal was 11.12%. WRBA once again was the most effective wire removal style, with an average improvement of 9.22%. WRA and WRB had an average improvement of 7.58% and 5.82% respectively. The detailed results of this experiment are not included, since they substantially track the results reported in this section.

7 Conclusions and Future Work

We have demonstrated that binary and multi-valued SPFD based wire removal are effective techniques for reducing the wiring, and therefore the overall layout area, of a circuit implemented as a network of PLAs. Our main findings are summarized below:

- Wire removal results in a best case layout area reduction on average of about 11%.
- This reduction increases to 19% or higher for larger examples, further suggesting the effectiveness of the technique.
- By choosing the best result among WRA and WRBA, we obtain an improvement which is almost as good as the best case improvement over all 3 wire removal styles. These two styles of wire removal account for the best case improvement in a majority of the examples.

In the future we plan to use wire removal after placement as well. After placement, we may have *critical wires* in the sense that if these wires are removed, there would be a reduction in layout area. Performing wire removal directed at such wires should further improve the results obtained.

Also, in our current implementation, the height of the PLAs is allowed to grow when we perform multi-valued SPFD-based wire removal. We plan to remove this restriction, which should probably result in further area savings.

As mentioned in Section 5.2, we also plan to investigate ideas to further exploit the flexibility of MV-SPFD based wire removal.

8 Acknowledgements

This research was supported partially by the SRC (under grant number 683), the GSRC/Marco center at Berkeley, and the California micro program with our industrial sponsors, Motorola, Fujitsu, Synopsys, and Cadence.

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, R. Brayton, and A. Sangiovanni-Vincentelli, "A VLSI design methodology using a network of PLAs embedded in a regular layout fabric," Tech. Rep. UCB/ERL M99/50, Electronics Research Laboratory, University of California, Berkeley, May 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] S. Khatri, S. Sinha, A. Kuehlmann, R. Brayton, and A. Sangiovanni-Vincentelli, "SPFD based wire removal in a network of PLAs," in *Workshop Notes, International Workshop on Logic Synthesis*, (Tahoe City, CA), May 1999.

[7] R. Bryant, "Graph-based Algorithms for Boolean Function Manipulation," *IEEE Trans. Computers*, vol. C-35, pp. 677–691, Aug. 1986.

[8] A. Srinivasan, T. Kam, S. Malik, and R. K. Brayton, "Algorithms for Discrete Function Manipulation," in *Proc. of the Intl. Conf. on Computer-Aided Design*, pp. 92–95, Nov. 1990.

[9] R. Rudell and A. Sangiovanni-Vincentelli, "Espresso-mv: Algorithms for multiple-valued logic minimization," in *Proceedings of the IEEE 1985 Custom Integrated Circuits Conference*, pp. 230–4, May 1985.

[10] W. Jiang and R. Brayton, "Don't cares and multi-valued logic minimization," *Internal Report, CAD Group, UC Berkeley*, May 1999.

[11] 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.

[12] 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.

[13] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Proceedings of the International Workshop on Field Programmable Logic and Applications*, 1997.

[14] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf Placement and Routing Package," *IEEE Journal of Solid-State Circuits*, 1985.