# SPFD-Based Wire Removal in Standard-Cell and Network-of-PLA Circuits

Sunil P. Khatri, *Member, IEEE*, Subarnarekha Sinha, *Member, IEEE*, Robert K. Brayton, *Fellow, IEEE*, and Alberto L. Sangiovanni-Vincentelli, *Fellow, IEEE*

***Abstract*—Wire removal is a technique by which the total number of wires between individual circuit nodes is reduced, either by removing wires or replacing them with other new wires. The wire removal techniques we describe in this paper are based on both binary and multivalued sets of pairs of functions to be distinguished (SPFDs). Recently, it was shown that a design style based on a multilevel network of approximately equal-sized programmable logic arrays (PLAs) results in a dense, fast, and crosstalk-resistant layout. This paper describes the application of SPFD-based wire removal techniques for circuit implementations utilizing networks of PLAs as well as standard-cells. In our first set of wire removal experiments (which utilize binary SPFD-based wire removal), we demonstrate that the benefit of SPFD-based wire removal is 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. In the next set of wire removal experiments, we focus only on circuits implemented using a network of PLAs. Three separate wire removal experiments are performed. 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, multivalued SPFD-based wire removal is used since the multioutput PLAs can be viewed as multivalued 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.**

***Index Terms*—Logic, logic arrays, multivalued logic, programmable logic arrays, very large scale integration (VLSI).**

## I. 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 logic were implemented using single flat PLAs. Recent work [2] demonstrates that a circuit implementation based on a network of approximately equal-sized PLAs yields a fast, compact, and crosstalk-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.

*Wire removal* is a technique to reduce the total number of wires between individual circuit nodes, either by removing wires, or replacing them with other new wires. 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 congestion in the routing area. Several techniques based on redundancy addition and removal [3], [4] have been proposed to improve the area or the routability of a multilevel circuit by wire removal. In this paper, we focus on sets of pairs of functions to be distinguished (SPFDs) as a candidate technique for wire removal. We show that SPFD-based wire removal can remove wires that redundancy removal based techniques cannot. Our wire removal techniques utilize both binary SPFDs as well as multivalued SPFDs (MV-SPFDs).

SPFDs [5] were introduced in the context of field-programmable gate array (FPGA) optimization. In [6], this technique was refined and adapted to multilevel networks, while its application to logic optimization was described in [7]. The authors of [7] reported a significant average wire reduction for *technology-independent* wire removal.

We perform two separate experiments to test the effectiveness of SPFD-based wire removal. In our first experiment, we show that the benefit of binary SPFD-based wire removal is insignificant when the circuit is mapped using standard cells. For this experiment, we use the binary SPFD code of [7]. The authors of [7] report a significant average reduction in the number of wires for *technology-independent* wire removal. However, we show that when technology mapping is performed (using a standard-cell-based design flow) on the resulting circuits, the benefits of wire removal are erased. Binary SPFD-based wire removal *after* technology mapping is not effective because the resulting circuit after wire removal needs to be remapped, again erasing the benefits of wire removal. On the other hand, we demonstrate that binary SPFD-based wire removal is very effective in the context of a network of PLAs. In this experiment, we do not utilize MV-SPFDs. We apply the wire removal algorithm of [7] to a few circuits implemented using a network of PLAs, and demonstrate an approximate 13% reduction in the

Manuscript received November 22, 2002; revised June 12, 2003. This work was supported by the IEEE. This research was supported in part by the Survey Research Center (SRC) under Grant 683, in part by the Gigascale Silicon Research Center (GSRC)/Marco center at Berkeley, and in part by the California Micro program with industrial sponsors Motorola, Fujitsu, Synopsys, and Cadence. This paper was recommended by Associate Editor M. Sarrafzadeh.

S. P. Khatri is with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: sunil@ee.tamu.edu).

S. Sinha is with Synopsys, Inc., Mountain View, CA 94043 USA (e-mail: subarna@synopsys.com).

R. K. Brayton and A. L. Sangiovanni-Vincentelli are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94708 USA (e-mail: brayton@eecs.berkeley.edu; alberto@eecs.berkeley.edu).

Digital Object Identifier 10.1109/TCAD.2004.829821

number of wires, which directly translates into a reduction of the (placed and routed) layout area. This is because a separate technology mapping step is not required when the circuit is implemented as a network of PLAs.

Encouraged by the layout area reduction using binary SPFD-based wire removal for circuits implemented using a network of PLAs, we next generalize the notion of SPFDs to multivalued networks, and formulate an MV-SPFD-based wire removal scheme. We observe that (multioutput) PLAs can be naturally modeled as multivalued functions. Hence, a network of PLAs can be modeled as a multilevel network of multivalued nodes. We extend the binary wire removal technique described in [7] to the multivalued 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 multivalued node is more complex than the binary nodes encountered in [7], additional flexibility is obtained in optimizing them, as evidenced by our results.

In our second experiment, we focus on MV-SPFD-based wire removal for circuits implemented using a network of PLAs.[1] We describe three related wire removal experiments. Wire removal is invoked either 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. Binary SPFD-based wire removal is performed in the manner described in [7]. For wire removal after clustering, MV-SPFD-based wire removal is used since the multioutput PLAs can be viewed as multivalued 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% compared to a PLA network without wire removal. This reduction is significantly higher (about 20%) for the larger circuits we used in our experiments. Although the full flexibility of multivalued 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 II, we describe the circuit implementation style using a network of PLAs. Section III describes binary SPFDs and their use in removing wires in binary networks. Section IV introduces MV-SPFDs, while Section V outlines our MV-SPFD-based technique for wire removal. Section VI reports the experimental results for the two sets of wire removal experiments we conducted. Finally, Section VII concludes the paper and gives some directions for future work in this area.

## II. 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 not susceptible to crosstalk, an increasingly important design consideration in deep submicrometer (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 to decompose a multilevel circuit into a network of PLAs.

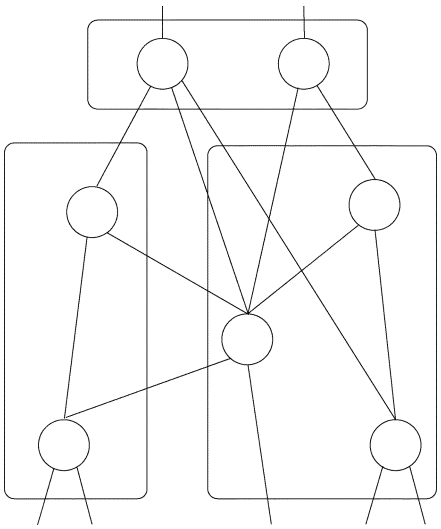

Fig. 1. Multilevel circuit decomposed into a network of PLAs.

A sample multilevel circuit, with nodes shown as circles, is shown in Fig. 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 Fig. 2. The dark rectangles in this figure correspond to the PLAs in the design.

## III. Binary SPFDs

### A. Definitions

SPFDs are a new way to represent the flexibility of a node in a multilevel network. In this section, we focus on SPFDs for binary valued nodes.

(Binary) SPFDs were introduced in [5] in the context of FPGA optimization. In [6], this technique was refined and adapted to multilevel networks, while its application to technology-independent logic optimization was described in [7].

*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:

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

$$g_2 \subseteq f \subseteq \overline{g}_1. \tag{2}$$

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

*Definition 2:* An **SPFD**

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

is a set of pairs of functions to be distinguished.

[1]Since SPFD-based wire removal was shown to be ineffective for standard-cell-based designs, we do not perform MV-SPFD-based wire removal for such designs. Also, a network of PLAs can be directly mapped to a multilevel network of multivalued nodes. There is no natural way to model a standard-cell-based design as a multivalued network of multivalued nodes. For these reasons, we restrict our attention to network of PLA-based designs in our second experiment.

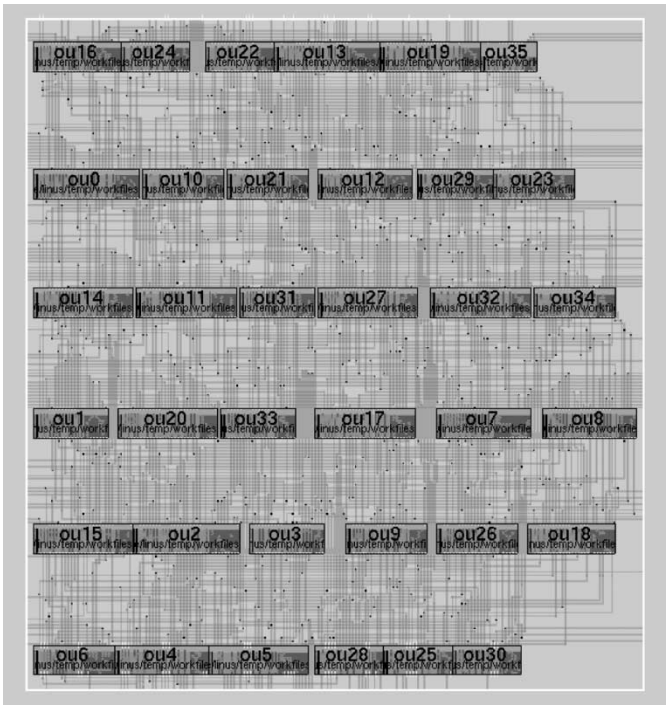


Fig. 2. Sample layout using network of PLAs.

We can think of an SPFD as an undirected graph with vertices

$$\{g_{1a}, g_{1b}, g_{2a}, g_{2b}, \ldots, g_{na}, g_{nb}\}. \tag{3}$$

This graph has edges

$$\{(g_{1a}, g_{1b}), (g_{2a}, g_{2b}), \ldots, (g_{na}, g_{nb})\}. \tag{4}$$

An edge $(g_{ia}, g_{ib})$ means that minterm $g_{ia}$ must be assigned a different functional value from minterm $g_{ib}$.

For example, consider $g_1 = ab$ and $g_2 = \overline{a}\overline{b}$. Now, if $f = a$, then we observe that condition 1 is satisfied. In other words, when $g_1$ is true, we can conclude that $f$ is also true, and that $g_2$ is false. Therefore, the function $f$ is able to distinguish the functions $g_1$ and $g_2$.

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

$$[(g_{1a} \subseteq f \subseteq \overline{g}_{1b}) + (g_{1b} \subseteq f \subseteq \overline{g}_{1a})] \wedge \ldots \wedge$$
$$[(g_{na} \subseteq f \subseteq \overline{g}_{nb}) + (g_{nb} \subseteq f \subseteq \overline{g}_{na})].$$

Thus, $f$ evaluates to a different value for each $g_{ia}$ and $g_{ib}$, where $i$ varies from 1 to $n$.

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 don't-care minterms of the node function. There are no edges incident on don't-care minterms.

If the SPFD consists of a single pair, it represents two incompletely specified functions (ISFs) where one is the complement of the other. If each of the

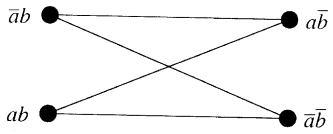


Fig. 3. Example SPFD.

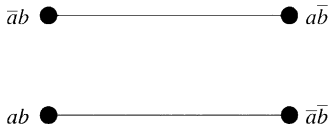


Fig. 4. Example SPFD after optimization.

$\{(g_{1a}, g_{1b}), (g_{2a}, g_{2b}), \ldots, (g_{na}, g_{nb})\}$ are pairwise disjoint, then the SPFD represents $2^n$ ISFs.[2]

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.

As an example to illustrate these ideas, consider the function $f(a, b) = b$. The SPFD of $f$ is obtained by considering $g_1 = f$ and $g_2 = \overline{f}$. The SPFD $\mathcal{F}$ of $f$ is a graph whose vertices are the points in $B^2$. Each onset minterm must be distinguished (have a different functional value) from each offset minterm, resulting in the SPFD $\mathcal{F}$ shown in Fig. 3.

If, during optimization, it is found that edges $(\overline{a}b, \overline{a}\overline{b})$ and $(ab, a\overline{b})$ are distinguished by another node in the multilevel network, the SPFD of $\mathcal{F}$ can be simplified to the graph shown in Fig. 4. Since the simplified SPFD of $\mathcal{F}$ has two connected components, there are four separate implementations at the node $f$ that satisfy the simplified SPFD $\mathcal{F}$. In general, each such implementation is an ISF.

### B. Wire Removal/Replacement Using Binary-Valued SPFDs

The information content of a wire (which is effectively the set of pairs of minterms it can distinguish) in a network can be efficiently 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 multilevel network, with some nodes $\eta_i, \eta_j$, and $\eta_k$. Given a wire $(\eta_i, \eta_j)$, its SPFD represents the pairs of minterms that have to be distinguished by it. In this sense, the SPFD of $(\eta_i, \eta_j)$ encodes the *information content* required of that wire (i.e., the set of pairs of minterms that must be distinguished by the wire). If the wire $(\eta_i, \eta_j)$ need not uniquely distinguish any minterms required of node $\eta_j$[3] (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 $(\eta_s, \eta_j)$ can replace the wire $(\eta_k, \eta_j)$ if all the minterms required to be distinguished by the wire $(\eta_k, \eta_j)$ are also distinguished by $(\eta_s, \eta_j)$. In other words, the objective is to replace wire $(\eta_k, \eta_j)$ from node $\eta_k$ to $\eta_j$ with a wire

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

[3]This is possible if all the pairs of minterms distinguished by $(\eta_i, \eta_j)$ are distinguished by other wire(s), $(\eta_i', \eta_j)$.

$(\eta_s, \eta_j)$ from node $\eta_s$ to $\eta_j$, such that the original SPFD at $\eta_j$ is covered by the union of the SPFDs of its inputs, and some gain is realized by this change. The cost function in this case is literal count. In the sequel, we shall refer to this technique as *wire_replace*. Note that in any iteration of *wire_replace*, at most one wire is removed from the fanin of any node. In [7], 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.

For a detailed exposition on SPFDs and how they are computed and used for wire replacement, see [7].

We use a simple example to illustrate why wire removal using SPFDs is more powerful than redundancy removal-based techniques. Consider the following example:

$$z_1 = \overline{g}b + g\overline{b}$$
$$g = \overline{a}b + a\overline{b}$$
$$z_2 = b + c.$$

Running redundancy removal on this example results in no simplification. Now, consider wire removal using SPFDs. The SPFD of the wire $(g, z_1)$ is given by the set $A = \{(00,10),(11,01)\}$ (in the set $A$, each minterm is of the form $gb$). Now, if we express the minterms of $A$ in terms of the primary inputs, $a$ and $b$, we get $A' = \{(00,10),(11,01)\}$ (the minterms in $A'$ are of the form $ab$). These computations are performed using binary decision diagram (BDD)-based image computations, and are described in detail in [7]. For the simple example above, we observe that the SPFD of the global function of $z_1$ (of the form $ab$) is $Z_1 = \{(11,01),(11,00),(00,10),(10,01)\}$. The SPFD of the global function of $g$ (of the form $ab$) is $G = \{(00,10),(10,11),(01,00),(11,01)\}$. Note that the SPFD of the edge $(g, z_1)$ is the set $A'$ of edges of $Z_1$ that are also distinguished by $G$, yielding $A' = \{(00,10),(11,01)\}$.

Since $g$ is a single fanout node, hence its SPFD is the same as the SPFD of its fanout wire, $(g, z_1)$. In general, the SPFD of any node is the union of the SPFDs of its fanout wires. Thus, the SPFD of $g$ (which is the set $A'$) has two edges, both of which can be distinguished by the wire $(a, g)$. Hence, the wire $(b, g)$ does not do anything and can be removed. If we now alter the functions of nodes $g$ and $z_1$ to reflect these changes (for details on how to do this, see [7]), the new simplified circuit can be represented as

$$z_1 = a$$
$$z_2 = b + c.$$

The additional flexibility that SPFDs provide over CODCs [8] or redundancy removal is due to the fact that we alter the function of each node *and* its fanins simultaneously. This allows minterms in the original onset and offset to be suitably swapped to get many different functions, some of which cannot be obtained by CODCs or redundancy removal.

## IV. MV-SPFDs

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

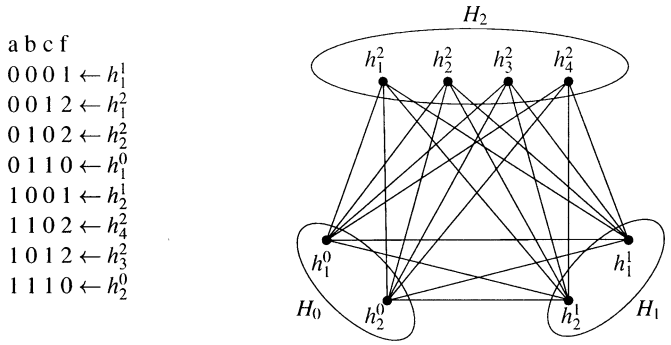


Fig. 5. MV-SPFD.

*Definition 4:* An **MV-SPFD** $\mathcal{F}(\mathrm{y})$ 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.

Fig. 5 shows a multivalued node $H$ with $k = 3$ 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 $\{h_1^i, h_2^i, \cdots, h_{n_i}^i\}$, where $\Sigma n_i = N$, the total number of minterms of $f$. 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** $\mathcal{F} = (V, E)$ 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 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 multivalued function. An MV-SPFD with $n$ connected components can be colored in

$$\left[C_{k_1}^N \cdot (k_1!)\right] \cdot \left[C_{k_2}^N \cdot (k_2!)\right] \cdots \left[C_{k_n}^N \cdot (k_n!)\right]$$

ways, where $k_i$ is the chromatic number of the $i$th connected component. Each different coloring of this graph represents a different incompletely specified multivalued function (ISF). This allows us flexibility in implementing the multivalued function, which can be exploited in many ways.

The MV-SPFD of a completely specified multivalued function is a complete k-partite graph. As we saw in the binary-valued SPFD example in Section III-A, it is possible that after optimization, a subset of edges of the MV-SPFD are removed. These edges are distinguished by some other node(s) in the multivalued, multilevel logic network.

In Section V, we will illustrate why MV-SPFDs, rather than binary SPFDs, are a more natural choice for a network of PLAs.

## V. Wire Removal Using MV-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. In this way, a network of PLAs can be modeled

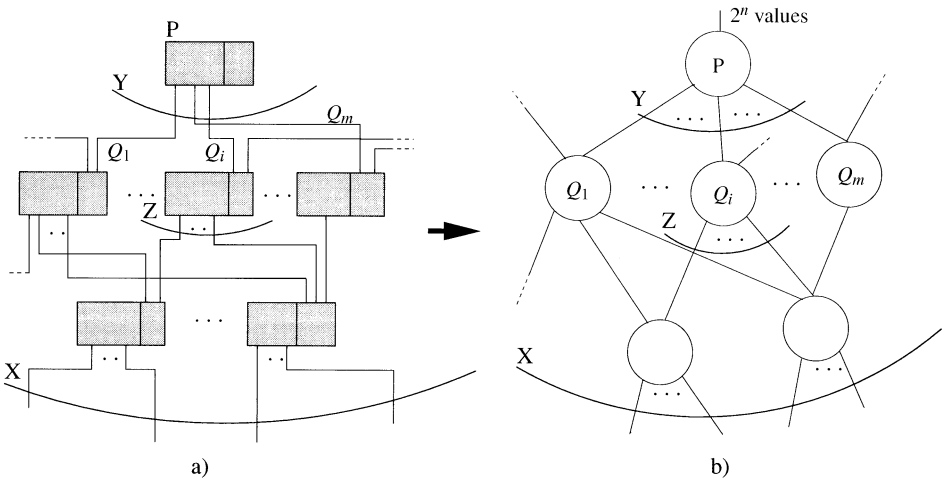


Fig. 6. MV-SPFD-based wire removal. (a) A network of PLAs. (b) Its corresponding multivalued network.

as a multilevel network of multivalued nodes. An MV-SPFD can be computed for each node and can be used to remove wires in its fanin. The binary SPFD techniques for computing and distributing SPFDs using BDDs [9] can be generalized to MV-SPFD techniques using multivalued decision diagrams (MDDs) [10]. The details of the computation are discussed below.

Consider a node $\eta_j$ in a multilevel, multivalued logic network. We know that the MV-SPFD of $\eta_j$ represents the set of multivalued minterms (henceforth equivalently referred to as minterms) that should be distinguished by $\eta_j$ in order to ensure that the functions of the primary outputs remains unchanged. To achieve this, it is necessary that each pair of minterms in the MV-SPFD of $\eta_j$ be distinguished by at least one of its fanin wires.[4] Thus, the union of the MV-SPFDs of its fanin wires should cover the MV-SPFD of $\eta_j$. Now, we can think of the pairs of minterms distinguished by the node/wire as the information content of the node/wire. In other words, 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 $(\eta_i, \eta_j)$ to be the set of pairs of minterms of $\eta_j$ that must be distinguished exclusively by this wire. In order to ensure that all the pairs of minterms in the MV-SPFD of $\eta_j$ are distinguished, the wire $(\eta_i, \eta_j)$ must distinguish at least these pairs of minterms.

Given the MV-SPFD of the node $\eta_j$, 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 $\eta_j$. 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 empty. This is because there could be two fanin wires $(\eta_i, \eta_j)$ and $(\eta_k, \eta_j)$ with empty minimum MV-SPFDs, such that both wires distinguish the pair of minterms $(m_1, m_2)$ in the MV-SPFD of $\eta_j$, 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 $\eta_j$, and, hence, the resulting network will not be correct.

Algorithm 1 describes our algorithm for MV-SPFD-based wire removal. The steps of the algorithm are detailed as follows.

First, we construct a multivalued network $\mathcal{N}$ from the given network of PLAs, $\mathcal{M}$. Assume the PLA $P$ has $m$ inputs and $n$ outputs. Fig. 6(a) shows the 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 multivalued function (MVF)[5] with $2^n$ values, and $m$ multivalued inputs, as shown in Fig. 6(b).

For each multivalued node $P$ in the network $\mathcal{N}$, in topological order from the PIs of the network, we perform the following steps.

- The MV-SPFD of $P$, denoted as $\mathcal{S}_P(Y)$, is computed from its original multivalued function (MVF). This MV-SPFD of $P$ distinguishes every minterm in every component of its MVF from every minterm in every other component of its MVF. After computing $\mathcal{S}_P(Y)$ (here, $Y$ is the space of the fanins of $P$), we reassign the task of distinguishing edges of $\mathcal{S}_P(Y)$ to the fanins of $P$ in the following steps.
- Fanins of $P$ that have nonempty minimum MV-SPFDs, denoted as $Y'$, are first identified.
- All the edges $e$ of $\mathcal{S}_P(Y)$ that are distinguished by these fanins are assigned to these fanins and are removed from $\mathcal{S}_P(Y)$.
- A weighted covering problem $W$ is set up between the remaining fanins of $P$, $Y \setminus Y'$, and the remaining edges of $\mathcal{S}_P(Y)$. 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

[4]We require that each pair be distinguished by a fanin wire, instead of any wire in the transitive fanin of $\eta_j$, to minimize the changes in the transitive fanin of a node.

[5]A multivalued function (formally, $\mathcal{F} : P_1 \times P_2 \times \ldots \times P_m \mapsto P_n$) of $m$ variables $X_1, X_2, \ldots, X_m$ can take on $P_n$ integer values $\{0, \ldots, |P_n| - 1\}$.

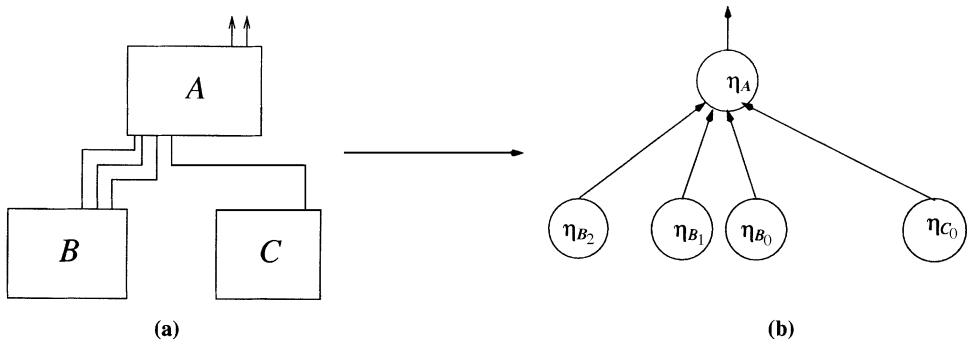


Fig. 7. Example. (a) PLA network. (b) Corresponding MV network.

be $Y''$. Note that we may remove more than one wire in one iteration of this algorithm (as contrasted to *wire_replace* where only one wire is removed per iteration).

- 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 $\mathcal{S}_P(Y)$ is computed on the primary input space $X$. This image is projected back to the $\hat{Y}$ space, to get $\hat{\mathcal{S}}_P(\hat{Y})$, 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 [11] 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. This procedure involves MDD-based image computations, and in general, it is not feasible for very large circuits. Typically, MV-SPFD-based wire removal (and binary SPFD-based wire replacement as well) work for circuits whose MDDs/BDDs can be computed. We are looking into overcoming these limitations by utilizing alternative methods to perform the image computations.

As mentioned earlier, any valid coloring of $\mathcal{S}_P(Y)$ 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 $P$. In practice, this can prove to be expensive since the entire network may need to be modified in general, when a single node is modified. To avoid this, we *block* the changes in the new function by its MV-CODCs [12] (a generalization of compatible observability don't cares (CODCs) [8] for the multivalued case). In other words, we only consider those changes at $P$ which would be contained in its MV-CODC set. Since such changes are by definition *compatible*, the nodes in the transitive fanout of $P$ need not be modified when $P$ is modified. Thus, at any point in the algorithm, the *region of change* consists of a single node, and possibly its immediate fanins.

Let us consider the following simple example to illustrate the working of the algorithm. Consider the network of PLAs shown in Fig. 7(a). PLA $B$ has three outputs and $C$ has one output. The function of $A$ is given in Table I. The corresponding MV network is shown in Fig. 7(b) and the functionality of $\eta_A$ is shown in Table II.

The MV-SPFD of $\eta_A$ is the set of edges $\{(1---, 0000), (1---, 0111), (0000, 0111)\}$(any input minterm is of the form $C_0B_0B_1B_2$). Now, the minimum MV-SPFD of a fanin wire are the set of edges that are exclusively distinguished by that wire. So, the minimum MV-SPFD of the wire $(\eta_{C_0}, \eta_A)$ is the set $\{(1000, 0000), (1111, 0111)\}$. Since it is not empty, we cannot remove this wire. On the other hand, the minimum MV-SPFD of each of the input wires $(\eta_{B_0}, \eta_A)$, $(\eta_{B_1}, \eta_A)$ and $(\eta_{B_2}, \eta_A)$ is empty as neither of these wires *uniquely* distinguishes the edge $(0000, 0111)$, although each of these wires distinguishes the edge $(0000, 0111)$. Hence, by the algorithm, we need to retain only one of $(\eta_{B_0}, \eta_{C_0})$, $(\eta_{B_1}, \eta_{C_0})$ or $(\eta_{B_2}, \eta_{C_0})$. Suppose we choose to retain $(\eta_{B_2}, \eta_{C_0})$. Then, the modified function at $A$ is shown in Table III.

TABLE I
FUNCTION TABLE OF PLA $A$

| $B_0$ | $B_1$ | $B_2$ | $C_0$ | $A_0$ | $A_1$ |
|---|---|---|---|---|---|
| - | - | - | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 |

TABLE II
FUNCTION TABLE OF MV NODE $\eta_A$

| $B_0$ | $B_1$ | $B_2$ | $C_0$ | $A$ |
|---|---|---|---|---|
| - | - | - | 1 | 3 |
| 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 2 |

TABLE III
MODIFIED FUNCTION TABLE OF PLA $A$

| $B_2$ | $C_0$ | $A_0$ | $A_1$ |
|---|---|---|---|
| - | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |

**Algorithm 1** MV-SPFD-Based Wire Removal
$\mathcal{N} = \text{construct\_mv\_network}(\mathcal{M})$
**for** $P \in \mathcal{N}$ (in some topological order from PIs) **do**
  Construct $Y = \text{fanins}(P)$
  Compute $\mathcal{S}_P(Y)$ from its MVF
  $Y' = \phi$
  **for all** $Q_k \in \text{fanins}(P)$ **do**
    $\mathcal{S}_{(Q_k,P)} = \text{minimum MV-SPFD of } e_k = (Q_k, P)$
    **if** $\mathcal{S}_{(Q_k,P)} \neq \phi$ **then**
      $Y' \leftarrow Y' \cup Q_k$

```
    end if
  end for
  for all E ∈ S_P(Y) do
    if E ∈ S_P(Y) is distinguished by some y ∈ Y'
  then
      Remove E from S_P(Y), and assign it to y
    end if
  end for
  for all fanins Q_j ∈ Y \ Y' do
    w(j) = 1/(num_fanouts(Q_j))
  end for
  Construct W(Y \ Y', remaining_edges(S_P(Y)), w(j))
  Y'' = solution of W
  Ŷ = Y' ∪ Y''
  Construct Ŝ_P(Ŷ)
  C = color(Ŝ_P(Ŷ))
  ESPRESSO-MV(C) gives the new function of P
end for
```

### A. Advantages of MV-SPFD-Based Wire Removal

There are several reasons why MV-SPFD-based wire removal is a better choice than binary SPFD-based wire removal for a network of PLA implementation of a circuit.

- In the example above, if we had used binary SPFDs, then we would have specified a binary SPFD at each output of the PLA separately. Then, the SPFDs of $A_0$ and $A_1$ are $\{(1---, 0000), (0111, 0000)\}$ and $\{(1---, 0111), (0000, 0111)\}$, respectively (the minterms are in the form $C_0B_0B_1B_2$). In order to distinguish the SPFD of $A_0$, we could use the inputs $\{B_0, C_0\}$ and for $A_1$, we could use the inputs, $\{B_1, C_0\}$.[6] Therefore, the final PLA will have three inputs, unlike the final PLA in the previous case which has two inputs.[7] So, it is more advantageous to look at all the outputs of a PLA at the same time and MV-SPFDs are ideal for that purpose.
- Also, with binary SPFDs, the actual encoding of the outputs becomes important. Thus, in our example, we would have a different answer if the outputs of $A$ are encoded differently. Even if this is the case, however, the MV-SPFD of the node remains unchanged (assuming that the three minterms have different functional values).
- Similarly, when we reimplement the PLA after removing the wires, it is better to consider the modified MV-SPFD of the entire PLA instead of considering the modified binary SPFDs of the outputs of the PLA separately. In the former case, it is possible to change the number of outputs of the PLA while in the latter case, this is not possible.
- It was observed [7] that whenever the nodes were complex, SPFD-based optimizations resulted in better results. The multivalued 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, since complex nodes typically increase the likelihood of having a large number of disconnected components in the SPFDs of the fanins.

[6]Note that the inputs to distinguish $A_0$ and $A_1$ are selected independently of each other.

[7]This is because we analyze all values of the (multivalued) output simultaneously (i.e., we implicitly analyze all output bits of the PLA simultaneously).

### B. Extensions

In this paper, we only do wire removal using MV-SPFDs. However, we can easily extend the algorithm to perform wire replacement. Thus, given the minimum MV-SPFD of a wire $(\eta_i, \eta_j)$, we can replace it with another wire $(\eta_k, \eta_j)$, if all the edges of $(\eta_i, \eta_j)$ are distinguished by $(\eta_k, \eta_j)$. In the example given in Section IV, the wire $(a, z_1)$ can replace the wire $(g, z_1)$. However, if we use redundancy addition and removal-based techniques [3] to generate alternate wires, we cannot get $(a, z_1)$ as an alternate to $(g, z_1)$.

In SPFD-based optimizations, when the fanins of a node $F$ are modified, the logic function of $F$ needs to be changed as well. In our implementation of (binary or MV) SPFD-based wire removal, when $F$ changes, we avoid the propagation of changes throughout the transitive fanout of $F$ by blocking these changes with the (binary or MV) CODCs of $F$. In general, in the application of MV-SPFD-based wire removal to circuits implemented using a network of PLAs, the changes to any node (i.e., a PLA) $F$ do not need to be blocked by its MV-CODCs. This is because fanout PLAs can be easily reimplemented if $F$ changes, as long as the total number of product terms in the fanout PLAs are bounded. This can result in significantly more flexibility while optimizing any given PLA.

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 $[C_{k_1}^N \cdot (k_1!)] \cdot [C_{k_2}^N \cdot (k_2!)] \cdots [C_{k_n}^N \cdot (k_n!)]$ ways, where $k_i$ is the chromatic number of the $i$th component. Each coloring of an MV-node would represent a different PLA encoding and thus different wiring connections between the PLA and its outputs. 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 an encoding algorithm to suitably modify the wiring between a node and its fanouts. This is a difficult problem and is currently being investigated.

## VI. Experimental Results

In Section VI-A, we perform experiments with binary SPFD-based wire removal. We demonstrate the utility of this technique in a network of PLAs, and show that it is not useful in a traditional standard-cell-based implementation style.

In Section VI-B, we perform binary and MV-SPFD-based wire removal experiments in a network of PLAs, and demonstrate the effectiveness of these techniques.

### A. Experiment 1

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 [7]. The computation is done using binary-valued SPFDs.

TABLE IV
WIRE REMOVAL WITHOUT PRIOR OPTIMIZATION

| Circuit | Standard Cell | | | Network of PLAs | | |
|---|---|---|---|---|---|---|
| | before WR | after WR | Ratio | before WR | after WR | Ratio |
| C432 | 1996.08 | 1956.64 | 0.980 | 2655.18 | 2285.26 | 0.861 |
| C499 | 4840.91 | 4914.14 | 1.015 | 4131.11 | 3599.70 | 0.871 |
| C880 | 4619.33 | 4869.08 | 1.054 | 7152.46 | 5400.49 | 0.755 |
| rot | 14254.21 | 13925.60 | 0.977 | 14612.87 | 8361.74 | 0.572 |
| alu2 | 4104.82 | 4144.26 | 1.010 | 3141.52 | 1626.16 | 0.518 |
| AVG | | | 1.007 | | | 0.715 |

TABLE V
WIRE REMOVAL AFTER *SCRIPT.RUGGED*

| Circuit | Standard Cell | | | Network of PLAs | | |
|---|---|---|---|---|---|---|
| | before WR | after WR | Ratio | before WR | after WR | Ratio |
| C432 | 1701.27 | 1874.02 | 1.102 | 2634.52 | 2439.23 | 0.926 |
| C499 | 4664.40 | 4547.98 | 0.975 | 4756.41 | 3785.60 | 0.796 |
| C880 | 4722.61 | 4611.82 | 0.977 | 3909.53 | 3248.56 | 0.831 |
| rot | 12504.12 | 12042.19 | 0.963 | 10372.84 | 7980.56 | 0.769 |
| alu2 | 3323.67 | 3030.73 | 0.912 | 1757.60 | 1774.50 | 1.010 |
| C1355 | 4964.84 | 5011.79 | 1.009 | 3995.91 | 3635.38 | 0.910 |
| C1908 | 5385.47 | 5528.18 | 1.026 | 4553.61 | 3836.30 | 0.842 |
| AVG | | | 0.995 | | | 0.869 |

Table IV reports the results of *wire_replace* on unoptimized circuits. In our tables, the final layout area of the circuit is measured in units of square microns. 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 V is organized in the same fashion, except that each circuit was first subjected to *script.rugged*, a technology-independent optimization script in SIS [13].

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 in circuit area (28.5% for the nonoptimized case, and 13.1% 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.

### B. Experiment 2

To validate the usefulness of 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 [7] and in Section III-B. This computation is done using 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 multivalued node, as described in Section V. At this point, MV-SPFD-based wire removal is invoked, using the algorithm described in Section V. We do not perform wire replacement in this step; only wire removal is performed.

  The reason for not performing binary-valued SPFD-based wire removal clustering are described in Section V-A. MV-SPFD-based wire removal is more powerful, and is a generalization of binary SPFD-based wire removal.

The clustering and wire removal code was written in SIS [13]. Placement of the network of PLAs was done using VPR [14], 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* [15].

The initial *blif* netlist for the benchmark circuit is clustered into nodes with up to five inputs. No redundancy removal is performed. Now the nodes of the resulting network is sorted in depth-first manner. The resulting array of nodes is sorted in *levelization*[8] order, and placed into an array $L$.

Now we greedily construct the logic in each PLA $P$, by successively combining nodes from $L$ into a single PLA. Next, we call a *PLA folding* routine which attempts to fold the inputs of $P$ so as to implement a more complex PLA in the same area. Finally, we check that the final PLA, after folding and simplification using *espresso*, satisfies the maximum width[9] and height[10] constraints, respectively. If so, we attempt to include another node from $L$ into $P$, otherwise we append the last PLA (satisfying the height and width constraints) to the result.

Nodes in the fanout of the nodes $N^*$ which are combined into the PLA $P$ are favored when attempting to include new nodes into $P$ (in an attempt to reduce the wiring between PLAs). If such nodes are not available, the first unmatched node from $L$ is returned.

The resulting PLA netlist is the starting point for all wire removal experiments. We now perform one of four 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 MV-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

[8]Primary inputs are assigned a level 0, and other nodes are assigned a level which is one larger than the maximum level of all their fanins.

[9]Based on the physical design of the PLAs we used, the width of the PLA is $4m + 2n$ (where $m$ is the number of inputs and $n$ the number of outputs of the PLA). Any values of $n$ and $m$ are permitted as long as the width of the PLA is less than the width constraint value. See [2] for details on the design of our PLAs.

[10]The height of the PLA is the number of cubes in the PLA.

TABLE VI
WIRE REMOVAL EXPERIMENTS—MAX WIDTH 40, MAX HEIGHT 15

| Circuit | NOWR | WRA | Improve % | WRB | WRBA | Improve % | WRA% | WRB% | WRBA% | BEST% |
|---|---|---|---|---|---|---|---|---|---|---|
| vda | 20862.11 | 17331.89 | 16.96 | 19040.67 | 16693.44 | 12.24 | 16.96 | 8.78 | 19.95 | 19.95 |
| frg2 | 12111.67 | 10703.33 | 11.63 | 11191.56 | 10233.89 | 8.61 | 11.63 | 7.49 | 15.45 | 15.45 |
| C1908 | 10590.67 | 10534.33 | 0.52 | 8600.22 | 8130.78 | 5.44 | 0.52 | 18.81 | 23.23 | 23.23 |
| apex6 | 8356.11 | 7586.22 | 9.25 | 8281.00 | 7980.56 | 3.54 | 9.25 | 0.98 | 4.48 | 9.25 |
| x3.blif | 8299.78 | 8149.56 | 1.88 | 8619.00 | 8281.00 | 3.89 | 1.88 | -3.72 | 0.32 | 1.88 |
| toolarge | 8093.22 | 8074.44 | 0.22 | 8262.22 | 8187.11 | 0.98 | 0.22 | -2.14 | -1.14 | 0.22 |
| x1 | 3849.44 | 3511.44 | 8.39 | 3398.78 | 3492.67 | -2.98 | 8.39 | 11.75 | 9.12 | 11.75 |
| x4 | 3830.67 | 3999.67 | -4.25 | 3642.89 | 3642.89 | 0.00 | -4.25 | 4.99 | 4.99 | 4.99 |
| alu2 | 3042.00 | 2760.33 | 9.26 | 3098.33 | 3060.78 | 1.05 | 9.26 | -1.40 | -0.34 | 9.26 |
| C432 | 2572.56 | 2535.00 | 1.95 | 2309.67 | 2328.44 | -0.71 | 1.95 | 10.53 | 9.89 | 10.53 |
| term1 | 2347.22 | 1971.67 | 15.96 | 1802.67 | 1652.44 | 8.62 | 15.96 | 23.26 | 29.88 | 29.88 |
| apex7 | 1859.00 | 1596.11 | 13.92 | 1783.89 | 1727.56 | 3.03 | 13.92 | 3.96 | 6.87 | 13.92 |
| ttt2 | 995.22 | 845.00 | 15.36 | 976.44 | 957.67 | 2.05 | 15.36 | 1.94 | 3.95 | 15.36 |
| count | 676.00 | 600.89 | 13.23 | 694.78 | 600.89 | 14.16 | 13.23 | -1.08 | 13.23 | 13.23 |
| pcle | 507.00 | 488.22 | 2.78 | 507.00 | 488.22 | 2.78 | 2.78 | 0.00 | 2.78 | 2.78 |
| decod | 338.00 | 338.00 | 0.00 | 338.00 | 338.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| AVERAGE | | | 7.32 | | | 3.92 | 7.32 | 5.26 | 8.92 | 11.35 |

TABLE VII
WIRE REMOVAL EXPERIMENTS—MAX WIDTH 40, MAX HEIGHT 20

| Circuit | NOWR | WRA | Improve % | WRB | WRBA | Improve % | WRA% | WRB% | WRBA% | BEST% |
|---|---|---|---|---|---|---|---|---|---|---|
| vda | 23359.56 | 19284.78 | 17.41 | 21237.67 | 17857.67 | 15.94 | 17.41 | 9.04 | 23.53 | 23.53 |
| frg2 | 10177.56 | 9050.89 | 11.15 | 10327.78 | 9520.33 | 7.83 | 11.15 | -1.54 | 6.42 | 11.15 |
| C1908 | 12543.56 | 10947.44 | 12.67 | 9726.89 | 8431.22 | 13.29 | 12.68 | 22.48 | 32.78 | 32.78 |
| apex6 | 9126.00 | 8806.78 | 3.37 | 8957.00 | 8374.89 | 6.64 | 3.38 | 1.70 | 8.23 | 8.23 |
| x3.blif | 9370.11 | 8431.22 | 10.12 | 8750.44 | 8919.44 | -1.94 | 10.12 | 6.76 | 4.95 | 10.12 |
| toolarge | 8844.33 | 8788.00 | 0.69 | 8769.22 | 8788.00 | -0.18 | 0.69 | 0.93 | 0.75 | 0.93 |
| x1 | 3962.11 | 3943.33 | 0.60 | 3887.00 | 3943.33 | -1.64 | 0.60 | 1.95 | 0.34 | 1.95 |
| x4 | 3868.22 | 3811.89 | 1.62 | 4056.00 | 4187.44 | -3.32 | 1.62 | -4.71 | -8.19 | 1.62 |
| alu2 | 3211.00 | 2985.67 | 6.73 | 3436.33 | 2891.78 | 15.50 | 6.73 | -7.06 | 9.53 | 9.53 |
| C432 | 2929.33 | 2760.33 | 5.81 | 2497.44 | 0.00 | – | 5.81 | 14.89 | – | 14.89 |
| term1 | 2441.11 | 2121.89 | 13.36 | 1765.11 | 1614.89 | 8.61 | 13.36 | 27.74 | 33.97 | 33.97 |
| apex7 | 1934.11 | 1821.44 | 5.75 | 2028.00 | 1934.11 | 4.71 | 5.75 | -4.54 | 0.39 | 5.75 |
| ttt2 | 1126.67 | 976.44 | 12.73 | 1126.67 | 1070.33 | 3.68 | 12.73 | 0.72 | 4.37 | 12.73 |
| count | 901.33 | 769.89 | 15.96 | 901.33 | 788.67 | 13.86 | 15.97 | 0.00 | 13.86 | 15.97 |
| pcle | 582.11 | 563.33 | 2.44 | 582.11 | 582.11 | 0.00 | 2.44 | 0.00 | 0.00 | 2.44 |
| decod | 375.56 | 375.56 | 0.00 | 375.56 | 375.56 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| AVERAGE | | | 7.53 | | | 5.19 | 7.53 | 4.27 | 8.18 | 11.60 |

netlist, and then cluster the resulting netlist into a network of PLAs. This is followed by MV-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 VI 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 VII 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 five or less outputs in both cases. In both tables, the final layout area of the circuit is measured in units of square micrometers. 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 11th 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. In all the experiments, there was no situation where an entire PLA was removed (by removing the output wires of any PLA, the entire PLA may be removed). So

the number of PLAs remained unchanged in all cases, and the area reduction is entirely due to wire removal.

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 VI and Table VII, 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. In general, WRB is least effective, since it utilizes binary SPFD-based techniques. WRA provides better results, since it utilizes MV-SPFD-based techniques which we predicted would be more effective than binary SPFD-based techniques (Section V). In fact, WRBA gives the best results on average, providing further evidence that MV-SPFD-based wire removal is very effective and can improve on the wire removal performed using binary SPFDs. Due to the lack of standard techniques to do optimization on a multivalued network, we could not compare our MV-SPFD-based wire removal with any other technique.

Furthermore, the results reported in Section VI-A 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 VI, 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 VII, 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 nine values of maximum PLA height and width. The maximum height varied from 15 to 25 in steps of five, and the maximum width varied from 40 to 60 in steps of ten. The maximum number of outputs was restricted to five. We used the best area from each of these nine 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 VI and VII. This is primarily due to the fact that the two combinations of maximum width and height used in Tables VI and VII 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.

In the above experiments, all wire removal is performed before placement and routing of the PLAs. Thus, there is a possibility of an increase in circuit delay.[11] This can be effectively addressed by performing wire removal *after* an initial placement is obtained, and then not modifying the placement after wire removal. This would guarantee that circuit delays do not increase. We did not perform experiments based on this idea.

## VII. Conclusion 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. Our first experiment show that the binary-valued wire removal algorithm of [7] provides a 13% reduction in wiring for a network of PLAs, while the same algorithm delivers no improvement for a standard-cell-based implementation. This is true regardless of whether logic optimization is performed on the netlist or not.

In our second wire removal experiment, we focus exclusively on circuits implemented using a network of PLAs. We demonstrate the use of a combination of binary and MV-SPFD-based wire removal on a larger set of examples. Results show that we can obtain significant area savings, especially for large designs implemented as a network of PLAs.

The findings of this experiment are summarized as follows.

- 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 three wire removal styles. These two styles of wire removal account for the best case improvement in a majority of the examples.
- Also, since each of the MV nodes are complex, the MV-SPFD-based algorithm has a larger flexibility in reimplementing an MV node.

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 which targets such wires should further improve the results obtained. In [16], the authors report a similarly motivated scheme, which targets rewiring of delay-critical wires in a design.

Also, in our current implementation, the height of the PLAs is allowed to grow when we perform MV-SPFD-based wire removal. We plan to remove this restriction, which should probably result in further area savings. All the MV-SPFD computations are done using MDDs, which limit the applicability of the technique for some large circuits. We are looking at alternate ways to make the computations more rugged.

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

[11]Even though the technology-independent delay is unchanged by our technique, there is a possibility that a wire on the critical path is not removed by wire removal, and after placement and routing, it can become longer. This results in a greater circuit delay.

## 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 *Proc. Int. Conf. Computer Design (ICCD)*, Oct. 1998, pp. 17–23.

[2] S. P. Khatri, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Cross-talk immune VLSI design using a network of PLA's embedded in a regular layout fabric," in *Proc. Int. Conf. Computer-Aided Design*, Santa Clara, CA, Nov. 2000, pp. 412–418.

[3] S. Chang, K. Cheng, N. Woo, and M. Marek-Sadowska, "Layout driven logic synthesis for FPGAs," in *Proc. Design Automation Conf.*, 1994, pp. 308–313.

[4] L. Entera and K. Cheng, "Sequential logic optimization by redundancy addition and removal," in *Proc. Int. Conf. Computer-Aided Design*, 1993, pp. 310–315.

[5] S. Yamashita, H. Sawada, and A. Nagoya, "A new method to express functional permissibilities for LUT based FPGA's and its applications," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1996, pp. 254–261.

[6] R. Brayton, "Understanding SPFDs: a new method for specifying flexibility," in *Workshop Notes, Int. Workshop Logic Synthesis*, Tahoe City, CA, May 1997.

[7] S. Sinha and R. Brayton, "Implementation and use of SPFD's in optimizing Boolean networks," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1998, pp. 103–110.

[8] H. Savoj, "Don't cares in multi-level network optimization," Ph.D. dissertation, Electronics Res. Lab., College of Engineering, Univ. California, Berkeley, May 1992.

[9] R. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, pp. 677–691, Aug. 1986.

[10] A. Srinivasan, T. Kam, S. Malik, and R. K. Brayton, "Algorithms for discrete function manipulation," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1990, pp. 92–95.

[11] R. Rudell and A. Sangiovanni-Vincentelli, "Espresso-MV: algorithms for multiple-valued logic minimization," in *Proc. IEEE 1985 Custom Integrated Circuits Conf.*, May 1985, pp. 230–234.

[12] W. Jiang and R. Brayton, "Don't cares and multi-valued logic network minimization," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2000, pp. 520–525.

[13] 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," Electronics Research Laboratory, Univ. of California, Berkeley, Tech. Rep. UCB/ERL M92/41, May 1992.

[14] V. Betz and J. Rose, "VPR: a new packing, placement and routing tool for FPGA research," in *Proc. Int. Workshop Field Programmable Logic and Applications*, 1997, pp. 213–222.

[15] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf placement and routing package," *IEEE J. Solid-State Circuits*, vol. SSC-20, pp. 510–522, Apr. 1985.

[16] J. Cong, J. Lin, and W. Long, "A new enhanced SPFD rewiring algorithm," in *Proc. ICCAD02*, Nov. 2002, pp. 672–678.


**Sunil P. Khatri** (M'98) received the B.Tech degree in electrical engineering from the Indian Institute of Technology, Kanpur, India in 1987, the M.S. degree in electrical and computer engineering from the University of Texas, Austin, in 1989, and the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, in 1999.

From 1989 through 1993, he worked at Motorola, Inc., where he was a member of the design teams of the MC88110 and PowerPC 603 RISC microprocessors. In January 2000, he joined the Electrical and Computer Engineering Department, University of Colorado, Boulder, as an Assistant Professor. He is currently an Assistant Professor in electrical engineering at Texas A&M University, College Station. His research interests include logic synthesis, physical design automation and novel VLSI design flows to address deep submicrometer design issues such as power and crosstalk. He has coauthored over 25 technical publications, five United States Patent awards, and a book titled *Cross-talk Noise Immune VLSI Design using Regular Layout Fabrics* (Norwell, MA: Kluwer, 2001), and has presented numerous invited talks.

**Subarnarekha Sinha** received the B.Tech. degree in electronics and electrical communication engineering from the Indian Institute of Technology, Kharagpur, in 1996 and the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley, in 2002.

She is currently with Synopsys, Inc. Her research interests include VLSI logic synthesis.

**Robert K. Brayton** (M'75–SM'78–F'81) received the B.S.E.E. degree from Iowa State University, Ames, in 1956 and the Ph.D. degree in mathematics from the Massachusetts Institute of Technology, Cambridge, in 1961.

From 1961 to 1987, he was a member of the Mathematical Sciences Department of the IBM T. J. Watson Research Center, Yorktown Heights, NY. In 1987, he joined the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, where he is the Cadence Distinguished Professor of Engineering and the director of the SRC Center of Excellence for Design Sciences. He held the Edgar L. and Harold H. Buttner Endowed Chair in Electrical Engineering at University of California, Berkeley from 1996 to 1999. He has authored over 400 technical papers and nine books. Past contributions have been in analysis of nonlinear networks, electrical simulation and optimization of circuits, and asynchronous synthesis. His current research involves combinational and sequential logic synthesis for area/performance/testability, formal design verification, and logical/physical synthesis for DSM designs.

Dr. Brayton is a member of the National Academy of Engineering, and a Fellow of the AAAS. He received the 1991 IEEE CAS Technical Achievement Award, and five best paper awards, including the 1971 IEEE Guilleman-Cauer award, and the 1987 ISCAS Darlington award. He received the CAS Golden Jubilee Medal and the IEEE Millennium Medal in 2000. He was the editor of the *Journal on Formal Methods in Systems Design* from 1992 to 1996.

**Alberto L. Sangiovanni-Vincentelli** (M'74–M'77–SM'81–F'83) received the Dott. Ing. degree (summa cum laude) in electrical engineering and computer science from the Politecnico di Milano, Milan, Italy, in 1971.

He holds the Edgar L. and Harold H. Buttner Chair of Electrical Engineering and Computer Sciences at the University of California, Berkeley, where he has been on the Faculty since 1976. From 1980 to 1981, he spent a year as a Visiting Scientist with the Mathematical Sciences Division of the IBM T. J. Watson Research Center, Yorktown Heights, NY. In 1987, he was a Visiting Professor with the Massachusetts Institute of Technology, Cambridge. He co-founded Cadence Design Systems (where he is currently the Chief Technology Advisor and Member of the Board of Directors), Synopsys, Inc., (where he was the Chair of the Technical Advisory Board), and Comsilica, a startup in the wireless communication area (where he is currently the Chairman of the Board). He also founded the Cadence Berkeley Laboratories and the Kawasaki Berkeley Concept Research Center, where he is Chairman of the Board. He was a Director of ViewLogic and Pie Design Systems. He is currently a Member of the Board of Directors of Sonics, Inc., Softface, and Accent. He has consulted for a number of U.S. companies, including IBM, Intel, AT&T, GTE, GE, Harris, Nynex, Teknekron, DEC, HP, Japanese companies, including Kawasaki Steel, Fujitsu, Sony and Hitachi, and European companies including SGS-Thomson Microelectronics, Alcatel, Diamler-Benz, Magneti-Marelli, BMW, and Bull. He is the Scientific Director of the Project on Advanced Research on Architectures and Design of Electronic Systems, a European Group of Economic Interest. He is on the Advisory Board of the Lester Center of the Haas School of Business and of the Center for Western European Studies and a member of the Berkeley Roundtable of the International Economy. He has authored or coauthored over 530 papers and 14 books in the area of design methodologies, large-scale systems, embedded controllers, hybrid systems, and tools.

Dr. Sangiovanni-Vincentelli is a member of the National Academy of Engineering. He received the Distinguished Teaching Award of the University of California in 1981, the Guillemin-Cauer Award in 1982, the Darlington award in 1987, and the 1995 Graduate Teaching Award of the IEEE. He was the Technical Program Chairperson of the International Conference on Computer-Aided Design and is currently General Chair and was also the Executive Vice-President of the IEEE Circuits and Systems Society.