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

Subarnarekha Sinha (subarna@eecs.berkeley.edu) *
Sunil P. Khatri (spkhatri@colorado.edu) †
Robert K. Brayton (brayton@eecs.berkeley.edu) *
Alberto L. Sangiovanni-Vincentelli (alberto@eecs.berkeley.edu) *

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 [9], performance-critical parts of the control were implemented using single PLAs. Recent work [8] 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. Several techniques based on redundancy addition and removal [4, 5] have been proposed to improve the area or the routability of a multi-level circuit by wire removal and/or substitution. In this paper, we focus on Sets of Pairs of Functions to be Distinguished (SPFDs) as a candidate technique for wire removal. The circuit is assumed to be implemented using a network of PLAs [8]. We show that SPFD-based wire removal can remove wires that redundancy removal based techniques cannot.

SPFDs [16] were introduced in the context of FPGA optimization. In [2] this technique was refined and adapted to multi-level networks, while its application to logic optimization was described in [14]. The authors of [14] reported a significant average wire reduction for technology-independent wire removal. However, when technology mapping (using a standard cell based flow) was performed on the resulting circuits, the benefits of wire removal were erased. In [7], it was confirmed that SPFD-based wire removal is not a useful technique for standard-cell based design flows. In addition, in [7], 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 [14]. 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

*CAD Research Group, Department of EECS, University of California, Berkeley, CA 94720.

†VLSI CAD Research Group, Department of ECE, Campus Box 425, University of Colorado, Boulder, CO 80309

0-7695-0801-4/00 $10.00 © 2000 IEEE

PLAs can be modeled as a multi-level network with multi-valued nodes. We extend the binary wire removal technique described in [14] 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 [14], 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 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 [8], 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.

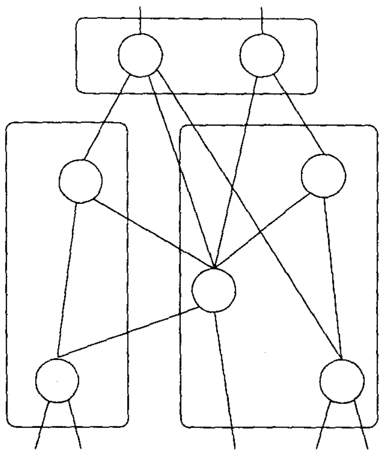


Figure 1. Multi-level circuit clustered into a network of PLAs.

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:*

$$g_1 \leq f \leq \overline{g}_2 \qquad (1)$$
$$g_2 \leq f \leq \overline{g}_1 \qquad (2)$$

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.*

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

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.

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 $\{(g_{1a},g_{1b}),(g_{2a},g_{2b}),\cdots,(g_{na},g_{nb})\}$ 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 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 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. In this sense, the SPFD of (η_i,η_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 (η_i,η_j) need not uniquely distinguish any minterms[2] (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 [14], 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.

For a detailed exposition on SPFDs and how they are computed and used for wire replacement, see [14].

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.*

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

[2]This is possible if all the pairs of minterms distinguished by (η_i,η_j) are distinguished by other wire(s), (η_i',η_j)

a	b	c	f	
0	0	0	1	$\leftarrow h_0^1$
0	0	1	2	$\leftarrow h_0^2$
0	1	0	2	$\leftarrow h_1^2$
0	1	1	0	$\leftarrow h_0^0$
1	0	0	1	$\leftarrow h_1^1$
1	1	0	2	$\leftarrow h_3^2$
1	0	1	2	$\leftarrow h_2^2$
1	1	1	0	$\leftarrow h_1^0$

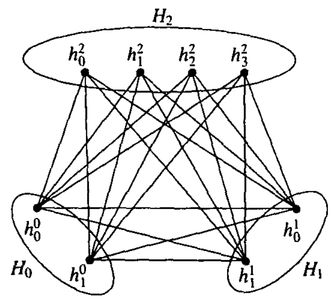


Figure 2. A Multi-valued SPFD.

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. A MV-SPFD with n connected components can be colored in $k_1!*\cdots*k_n!$, where k_i is the chromatic number of the ith connected component. Each different coloring of this graph represents a different incompletely specified multi-valued function (ISF). This allows us flexibility in implementing the multi-valued function, which can be exploited in many ways.

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

$$\begin{aligned} z_1 &= \bar{g}b+g\bar{b} \\ g &= \bar{a}b+a\bar{b} \\ z_2 &= b+c \end{aligned}$$

Running redundancy removal on this example results in no simplification. Now consider wire removal using (MV-)SPFDs[3]. The (MV-)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

[3]Since the network is binary, the binary SPFDs are the same as MV-SPFDs

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). Since, g is a single fanout node, hence its (MV-)SPFD is the same as the (MV-)SPFD of its fanout wire, (g,z_1). In general, the (MV)-SPFD of any node is the union of the (MV)-SPFDs of its fanout wires. Thus, the (MV-)SPFD of g (which is the set A') has two edges, both of which can be distinguished by (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 [14]), the new simplified circuit can be represented as:

$$\begin{aligned} z_1 &= a \\ z_2 &= b+c \end{aligned}$$

The additional flexibility that (MV-)SPFDs provide over CODCs [11] 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. Note that this example does not illustrate the differences between binary SPFDs and MV-SPFDs. In the next section, we will illustrate why MV-SPFDs, rather than binary SPFDs, are a more natural choice for a network of PLAs.

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 [3] can be generalized to MV-SPFD techniques using MDDs [15]. 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 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 η_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 η_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.

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

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 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.

Algorithm 1 describes our algorithm for multi-valued SPFD based wire removal. The steps of the algorithm are detailed below.

First we construct a multi-valued network $\mathcal{N}$ from the given network of PLAs, $\mathcal{M}$. Assume the PLA P has m inputs and n outputs. Figure 3-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 multi-valued node with 2^n values, and m multi-valued inputs, as shown in Figure 3-b.

For each multi-valued 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 multi-valued 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 re-assign the task of distinguishing edges of $\mathcal{S}_P(Y)$ to the fanins of P in the following steps.

- Fanins of P that have non-empty 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 $\mathcal{S}_P(Y)$.

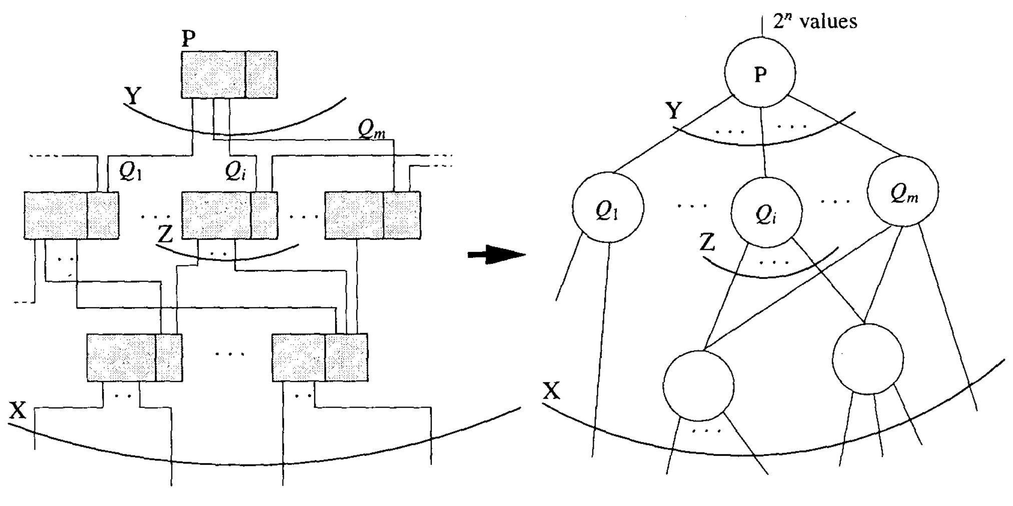


Figure 3. Multi-valued SPFD based wire removal.

- 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 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 $\mathcal{S}_P(Y)$ 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 [10] 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. 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,

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 1. Function Table of PLA A

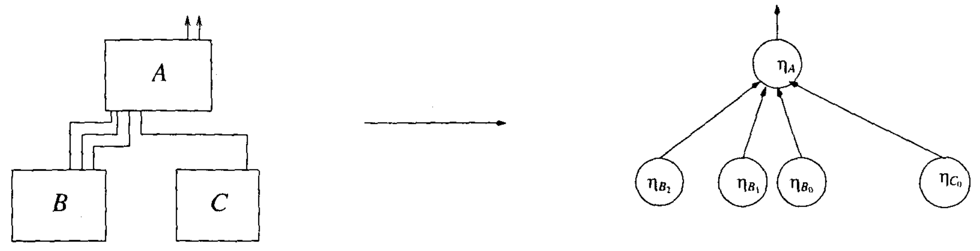
B_0	B_1	B_2	C_0	A
-	-	-	1	3
0	0	0	0	1
1	1	1	0	2

Table 2. Function Table of MV-node η_A

this can prove to be expensive. So we *block* the changes in the new function by its MV-CODCs [6] (a generalization of CODCs [11] for the multi-valued case). In other words, we only consider those changes at P which would be contained in its MV-CODC set. 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 Figure 4a). PLA B has 3 outputs and C has one input. The function of A is given in Table 1. The corresponding MV-network is shown in Figure 4b) and the functionality of η_A is shown in Table 2

So, the MV-SPFD of η_A is the set of edges $\{(1---,0000),(1---,0111\}),(0000,0111)\}$ (all the input minterms are in 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 (η_{C_0},η_A) is the set



PLA network

(a)

Corresponding MV network

(b)

Figure 4. Example.

Algorithm 1 MV-SPFD based wire removal

```
$\mathcal{N}$ = construct_mv_network($\mathcal{M}$)
for $P \in \mathcal{N}$ (in some topological order from PIs) do
    Construct $Y$ = fanins($P$)
    Compute $\mathcal{S}_P(Y)$ from its MVF
    $Y' = \phi$
    for all $Q_k \in$ fanins($P$) do
        $\mathcal{S}_{(Q_k,P)}$ = 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 $E \in \mathcal{S}_P(Y)$ do
        if $E \in \mathcal{S}_P(Y)$ is distinguished by some $y \in Y'$ then
            Remove $E$ from $\mathcal{S}_P(Y)$, and assign it to $y$
        end if
    end for
    for all fanins $Q_j \in Y \setminus Y'$ do
        $w(j) = 1/(\text{num\_fanouts}(Q_j)$
    end for
    Construct $W(Y \setminus Y', remaining\_edges(\mathcal{S}_P(Y)), w(j))$
    $Y''$ = solution of $W$
    $\hat{Y} = Y' \cup Y''$
    Construct $\hat{\mathcal{S}}_P(\hat{Y})$
    $C$ = color($\hat{\mathcal{S}}_P(\hat{Y})$)
    ESPRESSO-MV($C$) gives the new function of $P$
end for
```

B_2	C_0	A_0	A_1
-	1	1	1
0	0	0	1
1	0	1	0

Table 3. Modified Function Table of PLA A

$\{(1000,0000),(1111,0111)\}$. Since it is not empty, we cannot remove this wire. On the other hand, the minimum MV-SPFD of each of the input wires (η_{B_0},η_A), (η_{B_1},η_A) and (η_{B_2},η_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 (η_{B_0},η_{C_0}), (η_{B_1},η_{C_0}) or (η_{B_2},η_{C_0}). Suppose, we choose to retain (η_{B_2},η_{C_0}). Then, the modified function at A is shown in Table 3.

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 use the inputs $\{B_0,C_0\}$ and for A_1, we use the inputs, $\{B_1,C_0\}$. Then, the final PLA will have 3 inputs, unlike the final PLA in the previous case which has 2 inputs. 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 re-implement 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.

5.1 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 (η_i, η_j), we can replace it with the another wire (η_k, η_j), if all the edges of (η_i, η_j) are distinguished by (η_k, η_j). In the example given in Section 4, the wire (a, z_1) can replace the wire (g, z_1). However, if we use redundancy addition and removal based techniques [4] to generate alternate wires, we cannot get (a, z_1) as an alternate to (g, z_1).

Coloring of MV-SPFDs provides interesting possibilities. Each coloring of a MV-node would represent a different PLA encoding and thus different wiring connections between the PLA and its outputs. We are currently looking at ways to exploit this connection between coloring of an MV-SPFD and the resulting wiring changes at the output of a PLA.

6 Experimental Results

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 [14] 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 the algorithm described in Section 5. We do not perform wire replacement in this step; only wire removal is performed.

The clustering and wire removal code was written in SIS [13]. Placement of the network of PLAs was done using VPR [1], 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* [12].

The initial *blif* netlist for the benchmark circuit is clustered into nodes with up to 5 inputs. No redundancy removal is performed. 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 4 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 5 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 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. In all the experiments, there

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 4. Wire Removal Experiments - max width 40, max height 15

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 5. Wire Removal Experiments - max width 40, max height 20

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.

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 4 and Table 5 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 5). 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 multi-valued network, we could not compare our MV-SPFD based wire removal with any other technique.

Furthermore, the results reported in [7] 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 4, 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 5, 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 4 and 5. This is primarily due to the fact that the two combinations of maximum width and height used in Tables 4 and 5 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[5]. 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.

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. 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.

[5]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.

As mentioned in Section 5.1, 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] 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.

[2] R. Brayton. Understanding SPFDs: A new method for specifying flexibility. In *Workshop Notes, International Workshop on Logic Synthesis*, Tahoe City, CA, May 1997.

[3] R. Bryant. Graph-based Algorithms for Boolean Function Manipulation. *IEEE Trans. Comput.*, C-35:677–691, Aug. 1986.

[4] S. Chang, K. Cheng, N. Woo, and M. Marek-Sadowska. Layout driven logic synthesis for FPGAs. In *Proceedings of Design Automation Conference*, pages 308–13, 1994.

[5] L. Entera and K. Cheng. Sequential logic optimization by redundancy addition and removal. In *Proceedings of the International Conference on Computer-Aided Design*, pages 310–15, 1993.

[6] W. Jiang and R. Brayton. Don't cares and multi-valued logic minimization. In *Workshop Notes, International Workshop on Logic Synthesis*, Dana Point, CA, May-June 2000.

[7] 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.

[8] S. P. Khatri, R. K. Brayton, and A. Sangiovanni-Vincentelli. Cross-talk immune VLSI design using a network of PLAs embedded in a regular layout fabric. In *Proceedings of the International Conference on Computer-Aided Design*, Nov 2000. To appear.

[9] 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)*, pages 17–23, Oct 1998.

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

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

[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. Technical Report UCB/ERL M92/41, Electronics Research Lab, Univ. of California, Berkeley, CA 94720, May 1992.

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

[15] 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*, pages 92–95, Nov. 1990.

[16] 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*, pages 254–61, Nov 1996.