Circuit Structure Relations to Redundancy and Delay: The KMS Algorithm Revisited

Alexander Saldanha Robert K. Brayton Alberto L. Sangiovanni-Vincentelli
University of California - Berkeley CA

Abstract
Keutzer, Malik, and Saldanha (KMS) [1] have presented an algorithm that derives an equivalent non-redundant circuit implementation from a given redundant high-performance circuit, with no increase in delay (measured using viability analysis). In this paper we resolve the main bottlenecks in the KMS algorithm, arising due to an iterative loop of timing analysis, gate duplications, and redundancy removal. A circuit structure property based on path lengths is related to testability (redundancy) and delay. Based on this relationship, an efficient (single-pass) implementation of the KMS algorithm is presented. It consists of the transformation of any Boolean network to an equivalent circuit structure on which a single redundancy removal achieves the same effect as the original KMS algorithm.

1 Introduction
The relationship between the optimization criteria of area, performance, and testability in combinational circuits has been explored recently [1, 5]. While the relationship between the area and testability of a circuit has been known for some time, the relationship between the performance and testability of a circuit was first studied in [1]. There it is proved that redundancy is not necessary to reduce the delay of any circuit. An algorithm is provided that guarantees an irredundant implementation of any given redundant circuit with no increase in the delay. A principal drawback of the algorithm is the number of iterations of timing analysis, duplications, and redundancy removal that are performed. The algorithm operates on one longest false path at a time. Hence, it does not complete on circuits with large numbers of false paths.

Motivated by this observation, the relationship between circuit structure and path lengths versus testability and delay is explored. In particular, a new circuit structure property, termed L-path-disjoint, is defined that allows the simultaneous elimination of all (long) false paths via a single redundant multiple stuck-fault removal. An operation that transforms any circuit to possess this desired property is developed. The new single-pass KMS algorithm (referred to as the skMS algorithm) takes time proportional to the size of the circuit to perform these transformations. With efficient timing analysis [4] and redundancy removal techniques, the running time of the complete skMS algorithm is very small. The area (and fanout) increase of the algorithm is empirically shown to be very small. Thus, the skMS algorithm shows promise in eliminating the false path problem (and the accompanying reliability problems [3, 1]) in a typical synthesis scenario.

2 The KMS algorithm
This section is a review of the algorithm that derives an equivalent irredundant circuit at least as fast as a given redundant circuit [1]. First, a few basic definitions are reviewed.

Definition 2.1 A non-controlling value for a gate f is the value at its input which is not a controlling value for the gate. It is denoted as \(I(f)\). For example, 1 is a non-controlling value for an AND gate.

Definition 2.2 Let \(P = \{f_0, f_1, \ldots, f_m\}\) be a path. If \(P\) includes both a primary input and a primary output, \(P\) is an IO-path. The inputs of \(f_i\) other than \(f_{i-1}\) are called side-inputs of \(f_i\) along \(P\) and denoted as \(S(f_i, P)\). A path that starts at a primary input and ends at a side-input of \(P\) is a side-path of \(P\).

Definition 2.3 A path is statically sensitizable if there exists an input vector which sets all the side-inputs to the path to non-controlling values. The condition for static sensitization of a path \(P = \{f_0, f_1, \ldots, f_m\}\) composed of simple gates is

\[
\prod_{i=0}^{m} \prod_{f \in S(f_i, P)} (g = I(f_i)).
\]

Similar to the approach of [1], we use the notion of viability [2] in determining the conditions under which a path may contribute to the delay of a circuit. Refer to [2, 4] for the formal definition of viability. It is mentioned here that if a path is statically sensitizable then it is viable, though the converse is not true.

If any longest path in the circuit is (statically) sensitizable, the KMS algorithm invokes standard redundancy removal to derive an irredundant circuit. This follows simply because redundancy removal cannot increase the longest (and true) delay of the circuit. If all of the longest paths are not sensitizable, the KMS algorithm chooses one path, say \(P\), and attempts to remove a redundant fault on the first edge of \(P\). If \(P\) is fanout-free, i.e. each gate along \(P\) has fanout \(= 1\), then such a redundant connection always exists. If \(P\) is not fanout-free, duplication of some of the gates along \(P\) (gates duplicated include those between the first edge of \(P\) and the last gate along \(P\) with fanout \(> 1\)) is done to obtain a corresponding path \(P'\) that is fanout-free. The first edge of \(P'\) is now redundant, and is removed. It is proved in [1] that redundancy removal on the first edge of a longest path does not increase the delay of the circuit. Hence, by iterating the KMS algorithm, one path at a time in decreasing path length, an irredundant circuit, guaranteed to be no slower than the original is obtained.

An implementation of the original KMS algorithm is reported in [5]. The principal bottleneck of the algorithm is the number of iterations required. An iteration includes selecting a longest path, determining whether it is sensitizable or not; possible duplication and redundancy removal on the first edge of the path is performed if it is false. This algorithm fails to complete on all but small circuits or circuits with very few or no long false paths.

3 A single-pass algorithm
Due to the large number of false long paths in some circuits, it is imperative that any efficient algorithm must not explicitly
enumerate each false path while performing the KMS transformation. This section develops such an algorithm and proves its correctness.

**Definition 3.1** The set of all the paths beginning at connection \( c \) and terminating at a primary output is called the path-set of the connection \( c \), and is denoted \( PS_c \).

Note that the paths in the path-set of a connection are IO-paths only if the connection is from a primary input.

Consider a connection \( c \) from a primary input in a circuit \( \eta \). Additionally, assume the computed delay of \( \eta \) is less than \( L \) and let every path in \( PS_c \) be of length greater than or equal to \( L \). Note that every path in \( PS_c \) is an IO-path. On completion of the KMS algorithm, all the paths in the resulting circuit \( \eta' \), are of length less than or equal to \( L \). The KMS algorithm removes connections between primary inputs and some gates, i.e. only the first edge of any path is removed. Thus, \( c \) cannot exist in \( \eta' \), since every IO-path through \( c \) is of length greater than or equal to \( L \). This notion is captured formally by the next definition and theorems.

**Definition 3.2** A \( L \)-path-disjoint circuit is one where the paths in \( PS_c \), for any primary input connection \( c \), are either all of length \( \geq L \) or all of length \( < L \).

**Theorem 3.1** Let \( \eta \) be a circuit whose longest viable path is of length \( < L \). Let \( C = \{ c \} \) be the set of all primary input connections such that each path in \( PS_c \) is of length \( \geq L \) (and hence non-viable). Then any multiple stuck-fault composed of any combination of single stuck-0 or stuck-1 faults on each \( c \in C \) is redundant.

**Proof** Assume that some multiple stuck-fault \( F_C \) on \( C \) is irredundant. Let a vector \( v \) be a test for the fault. Consider the set of IO-paths \( P_{F_C} \) that propagate the fault effect to a primary output under vector \( v \). The length of each \( Q \in P_{F_C} \) is \( \geq L \) by assumption. Pick a path \( Q = \{ f_0, f_1, \ldots, f_k \} \in P_{F_C} \) with the property that for each \( f_i \in Q, f_i-1 \) under \( v \) is the earliest arriving input of \( f_i \) that propagates the fault effect. Such a path \( Q \) exists since the fault effect is propagated under \( v \) along some path in \( P_{F_C} \). Consider each gate \( f_i \) along \( Q \) in circuit \( \eta \) when vector \( v \) is applied. All the side-inputs to \( f_i \) that do not propagate the fault effect are at non-controlling values. The remaining side-inputs propagate the fault effect but each of them is available only at or after the time input \( f_i-1 \) arrives at \( f_i \). Hence these side-input are smoothed out (i.e. set to the non-controlling value) when considering the viability of the sub-path \( Q \) from \( f_0 \) up to \( f_i \). Hence the sub-path of \( Q \) up to \( f_i \) is viable. Since this is true at \( f_k \), \( Q \) is viable, contradicting that no path through \( c \) is viable.

**Theorem 3.3** Let \( \eta \) be a circuit such that all IO-paths in \( \eta \) of length \( \geq L \) are non-viable. Let \( C = \{ c \} \) be the set of all connections from a primary input such that each path in \( PS_c \) is of length \( \geq L \) (and hence non-viable). Let \( \eta' \) be the circuit resulting after asserting a multiple stuck-fault composed of any combination of the single stuck-0 or stuck-1 faults on each \( c \in C \). For any viable path \( \pi \) in \( \eta' \), the corresponding path \( \pi \) is viable in \( \eta \).

**Proof** Similar to the proof of Theorem 7.2 of [1]. The difference is that each gate may have several late-arriving side-inputs that are set to constant non-controlling values by the redundancy removal. However, these side-inputs always get smoothed out (i.e. set to the non-controlling value) and no change the viability conditions of any path \( \pi \) in \( \eta' \) from the viability of the corresponding path \( \pi \) in \( \eta \).

With Theorems 3.1 and 3.2 it is apparent that a multi-fault redundancy removal of the type specified can be done without increasing the computed delay (length of the longest viable path) or changing the logical behavior of the circuit. Moreover, no duplication is performed. Of course, not all circuits have such connections. A procedure is now described that transforms every circuit to a functionally equivalent \( L \)-path-disjoint circuit. In this case, all paths of length \( \geq L \) will be removed by the application of the above theorems. The only operation used is the duplication of gates and the transfer of some fanouts of a gate to its duplicate.

**Definition 3.3** The set of distinct path lengths from primary inputs to a gate \( f \) is denoted by \( atimes(f) \). The distinct path lengths from each gate \( f \) to the primary outputs is denoted by \( etimes(f) \).

If \( f \) is a primary input, \( atimes(f) \) is the single arrival time specified for \( f \). If \( f \) is a primary output, \( etimes(f) \) is 0. However, the algorithm could be generalized to take account of required times at the output; just take the maximum required time, \( R_{max} \), and think of a buffer on each output with delay \( R_{max} - R_f \), where \( R_f \) is the required time for \( f \). The algorithm to derive an \( L \)-path-disjoint network for a given network \( \eta \) is described in Figures 1 through 4. The main procedure, shown in Figure 1, consists of three phases.

The first phase computes the distinct paths lengths from primary inputs to each gate \( f \), and the distinct path lengths from each gate \( f \) to the primary outputs. This is done by simply performing a topological traversal from the inputs to the outputs for computing \( atimes(f) \), and a reverse topological traversal for computing \( etimes(f) \) (Figure 2).

The second phase consists of gate duplication and the transfer of some fanouts of a gate \( f \) to one or more duplicates of \( f \), and is shown in Figure 3. The essential operation performed on a gate with multiple fanout is the transfer of a set of fanouts connections of the gate to a duplicate gate. The gates are traversed in reverse topological order from primary outputs to primary inputs.

Let \( f \) be a gate that is to be processed by the algorithm. Let \( P_f \) be any path from a primary input up to \( f \). A duplication is only performed if there are at least two paths, \( Q_1 \) and \( Q_2 \).
Perform a delay trace on the network.

For each node $f$ in the network:
- If $f$ is a primary input, $\text{atimes}(f) = \{\}$.
- Else, compute $\text{atimes}(f)$ recursively.

To compute $\text{atimes}(f)$:
- For each fanout $g$ of $f$:
  - $\text{atimes}(g) = \text{atimes}(f)$.
  - Add $\text{atimes}(f)$ to $\text{atimes}(g)$ if $f$ is a primary output.

Perform the duplication as follows:
- For each gate $g$ in the network:
  - Compute $\text{atimes}(g)$.
  - If $\text{atimes}(g)$ is not a subset of $\text{atimes}(f)$, duplicate $g$.

The resulting network is an $L$-path-disjoint network if:
- For each primary output $f$, $\text{atimes}(f)$ is not a subset of $\text{atimes}(g)$ for any gate $g$.
- For any path $P$ from a primary input to a primary output, $|P| 

Figure 4: Setting constants on false paths

Figure 5: Example: Construction of an $L$-path-disjoint network

**Lemma 3.1** Consider gate $f$, in network $\eta$, that is processed by the algorithm of Figure 3. Assume that the lists $\text{atimes}(f)$ and $\text{etimes}(f)$ are computed using the procedure of Figure 2. Let $f'$ refer to gate $f$ or any of its duplicates. Then the following invariant is true for each $f'$: for each $f_{\text{atimes}} \in \text{atimes}(f')$, $f_{\text{atimes}} \neq f'$ and $f_{\text{etimes}} + \text{etimes}(f') \geq L^3$.

**Proof** By induction on the level of a node. The level of a node is the maximum number of nodes along any path from the node to the primary outputs. See [5] for details.

Since the invariant stated in Lemma 3.1 holds for each primary input, the final network is an $L$-path-disjoint network. Thus on completion of the second phase, the path-set of each first edge of any path of length $\geq L$ contains only paths of length $\leq L$. Hence, all such edges are set to constant 0 or 1 in the final phase of the single-pass algorithm (Figure 4).

Figure 5 illustrates the working of the algorithm on an example network. Each gate has unit delay and all primary inputs arrive at $t = 0$. Assume that the initial network (top of the figure) has a longest viable path of length 3. Hence an $L$-path-disjoint network for $L = 4$ is required. Following the first phase of the algorithm, the variables $\text{atimes}$ and $\text{etimes}$ are as follows:

\[
\begin{align*}
\text{atimes}(g1) &= \{1\} & \text{etimes}(g1) &= \{2, 3, 4\} \\
\text{atimes}(g2) &= \{1, 2\} & \text{etimes}(g2) &= \{2, 3\} \\
\text{atimes}(g3) &= \{1, 2, 3\} & \text{etimes}(g3) &= \{1, 2\}
\end{align*}
\]

There is no loss in generality in referring to the gates by a single representative $f'$, since, for each path from the primary inputs to $f$, there exists a corresponding path to some duplicate gate of $f$.

The notation $x + S < L$, for $x$ a scalar and $S$ a set, means that $x + s < L$ for all $s \in S$. $x + S \geq L$ has an analogous meaning.
duplication is done on these gates. Results of an implementation of the single-pass algorithm for gates 931 and 932 (second network from the top of the figure) are shown in Table 1. Each circuit the length of the longest sensitizable path, T, is first determined using an efficient timing analysis algorithm. The time to derive an L-path-disjoint network is proportional to the size of the original network. Although a tight area bound by the KMS algorithm is still open, empirically, the area penalty incurred is very small. In summary, application of the single-pass KMS algorithm performs the efficient removal of long false paths while guaranteeing that the delay does not increase, achieves full (single stuck-fault or multiple stuck-fault) testability, and increases the reliability of the circuit at a small expense in area. These results are a step towards resolving the question of whether it is ever necessary to synthesize circuits with long false paths.

<table>
<thead>
<tr>
<th>Name</th>
<th># Red.</th>
<th>Initial delay</th>
<th>Final delay</th>
<th># Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>cas 8.2</td>
<td>8</td>
<td>36.8</td>
<td>24.0</td>
<td>28.8</td>
</tr>
<tr>
<td>cas 12.2</td>
<td>16</td>
<td>69.6</td>
<td>36.0</td>
<td>30.4</td>
</tr>
<tr>
<td>cas 16.4</td>
<td>8</td>
<td>59.2</td>
<td>35.2</td>
<td>32.0</td>
</tr>
<tr>
<td>cas 32.4</td>
<td>16</td>
<td>114.4</td>
<td>47.2</td>
<td>42.0</td>
</tr>
<tr>
<td>cas 32.8</td>
<td>8</td>
<td>104.0</td>
<td>57.6</td>
<td>54.4</td>
</tr>
<tr>
<td>cas 64.4</td>
<td>16</td>
<td>206.0</td>
<td>69.8</td>
<td>72.1</td>
</tr>
</tbody>
</table>

The L-path-disjoint network is obtained by a reverse topological traversal. Since both g3 and g4 have single fanout, no duplication is done on these gates. g3 is duplicated to obtain gates g31 and g32 (second network from the top of the figure). g31 is connected to g4 while g32 is connected to g5. Now, etimes(g31) = [2], and etimes(g32) = [1], and the invariant of Lemma 3.1 is now satisfied for g31 and g32. Similarly, on duplicating g2 and g1 as shown in the figure, an L-path-disjoint network for L = 4 is obtained (bottom of the figure). Any multiple stuck-fault on the inputs a and b of g11, and c of gate g21 can now be removed. By the theorems discussed earlier in this section, the computed delay and logical behavior of the resulting circuit remains unchanged.

3.1 Results using the single-pass algorithm

Results of an implementation of the single-pass algorithm for redundancy removal, guaranteeing the delay does not increase, are shown in Table 1. Each circuit is composed of two-input gates and the delay of a gate is computed using a block delay of 1.0 plus a drive delay of 0.2 for each fanout connection. For each circuit the length of the longest sensitizable path, T, is first determined using an efficient timing analysis algorithm [4]. The smallest distinct path length L ≥ T is also known. Using the transformation of Figure 3, an L-path-disjoint circuit is derived from the original circuit. Finally, using Theorem 3.1 and 3.2, the first edge of every path of length L ≥ T is set to a constant. The topologically longest path in the resulting circuit is now of length L ≤ T (≤ T). Finally, standard redundancy removal is used to derive an irredundant circuit that is no slower than the original circuit. The algorithm provides the same effect as the KMS algorithm in a single-pass following the timing analysis phase. Often the resulting circuits are identical. The number of operations performed by the algorithm is linear in the number of connections of the original circuit and the number of distinct path lengths ≥ T in η.

The first column is the name of the circuit and the second indicates the initial number of redundancies. The third and fourth columns give the longest path delay (or topological delay) and the computed delay (using viability analysis) of the initial circuit, respectively. The fifth column gives the computed delay of the irredundant circuit obtained after application of the KMS algorithm. The last two columns compare the size of the initial redundant circuit against the irredundant circuit obtained by applying the KMS algorithm. The first set of experiments in the table are different examples [1]. The second set are optimized MCNC and ISCAS benchmark examples. Although not shown in the table, direct redundancy removal on the initial circuit leads to an increase in delay in several circuits. The single-pass algorithm completes on all the circuits experimented with. The CPU incurred DBC 5000, not including the timing analysis phase and the final redundancy removal, is only a few seconds even for the largest example. Similar results have been observed on mapped circuits that use technology dependent delay information, rather than the unit-fanout delay model of these experiments.

4 Conclusions

In this paper we resolve an important drawback of the KMS algorithm. On circuits with large numbers of false paths, the iterative loop of duplication and redundancy removal in the original algorithm leads to huge amounts of running time. Based on the notion of an L-path-disjoint network, where the paths of an input connection of the network are either all of length ≥ L or all of length < L, for some selected L, an efficient single pass algorithm is developed. The testability of a multiple stuck-fault on the L-path-disjoint network is related to the viability of paths (of length > L) in the original network. Thus, by a single multiple stuck-fault redundancy removal, any given redundant circuit with false paths is converted to a circuit with a longest sensitizable path. The time to derive an L-path-disjoint network is proportional to the size of the original network. Although a tight area bound by the KMS algorithm is still open, empirically, the area penalty incurred is very small. In summary, application of the single-pass KMS algorithm performs the efficient removal of long false paths while guaranteeing that the delay does not increase, achieves full (single stuck-fault or multiple stuck-fault) testability, and increases the reliability of the circuit at a small expense in area. These results are a step towards resolving the question of whether it is ever necessary to synthesize circuits with long false paths.

References


* A weak bound on nG for the area increase has been proved where G is the number of gates in the circuit and n is the number of distinct path lengths in the circuit between the longest visible path and the longest path length.