Scan-chain Optimization Algorithms for Multiple Scan-paths

Susumu Kobayashi  Masato Edahiro  Mikio Kubo
C&C Media Research Laboratories  
NEC Corporation  
Kawasaki, Japan  
Logistics and Information Engineering  
Tokyo University of Mercantile Marine  
Tokyo, Japan

Abstract—This paper presents an algorithm framework for the scan-chain optimization problem in multiple-scan design methodology. It also presents algorithms we propose based on the framework; these are the first algorithms ever proposed for multiple-scan designing. Experiments using actual design data show that, for ten scan-paths, our algorithms achieved a 90% reduction in scan-test time at the expense of a 7% total scan-path length increase as compared with the length of a single optimized scan-path.

I. INTRODUCTION

As circuits become larger and higher in density, the problem of increased scan-test time becomes crucial when scan design methodology [1] is employed. Multiple-scan design methodology, in which multiple scan-paths exist, is an effective method to reduce scan-test time because it reduces the number of flip-flops included in a scan-path.

Although scan design methodology is widely used in the design for testability, this method often causes a winibility problem because long routing wires connecting scan flip-flops (FF) spread over the entire layout area. Since the connection-order of FFs in a scan-path does not change circuit functionality, it is possible to decrease the scan-path’s wire length by optimizing the connection-order after the placement process. This optimization is called scan-chain optimization.

Previously a scan-chain optimization method for the purpose of minimizing routing area overhead was proposed in [2]. In this method, selecting and chaining of partial scan flip-flops are performed simultaneously by a matching-based algorithm.

It is assumed that multiple-scan design methodology will be widely used in the future to reduce scan-test time. In this methodology, there are multiple scan-in pins (SIP) and scan-out pins (SOP), and thus multiple scan-paths are constructed. The multi-scan-chain optimization problem constructs multiple scan-paths so as to minimize the total scan-path length. This problem essentially contains three types of optimization: 1) constructing pairs of SIPs and SOPs (each pair is associated with a scan-path), 2) assigning FFs to scan-paths under the condition that the number of FFs should be balanced among scan-paths, and 3) deciding connection-order of FFs in each scan-path. There is an intuitive method of constructing multiple scan-paths in which, first, an optimized single scan-path is constructed, the scan-path is evenly divided, and then each path is connected to an SIP and an SOP. However, this method might produce long extra wires connected to the SIPs or SOPs, because the optimized single scan-path is divided regardless of the locations of SIPs and SOPs.

Thus, in order to minimize the extra wire length generated for multiple scan-paths, we must optimize three terms—SIP-SOP matching, FF assignment, and FF ordering—so as to minimize the total scan-path length. These three terms greatly depend on each other, and it is extremely difficult to optimize these terms appropriately. As far as we know, the multi-scan-chain optimization problem has not been studied yet.

For this problem, we propose an algorithm framework that consists of four phases: 1) SIP-SOP matching, 2) FF assignment, 3) FF ordering, 4) FF exchanging among scan-paths for improvement. We also propose one or two algorithms for each phase.

In our experiments, the proposed algorithms reduced scan-test time by approximately 90%, and the resultant total scan-path length was only 7% longer than the single optimized scan-path length.

II. MULTI-SCAN-CHAIN OPTIMIZATION PROBLEM

The multi-scan-chain optimization problem is defined as follows:

Given locations of $k$ scan-in pins (SIP), $k$ scan-out pins (SOP), and $n$ FFs, assign SIPs, SOPs, and FFs to $k$ scan-paths and decide connection-order in each scan-path so that total wire length of all the scan-paths is minimized (Fig.1).

Each scan-path must satisfy the scan-path conditions described below.

Scan-path conditions:

- A scan-path must include only one SIP and only one SOP.
- A scan-path must begin with the SIP, pass once through all FFs in the path, and end with the SOP.
Each FF is assigned to one scan-path.

The difference in the number of FFs among scan-paths should be at most one. (We refer to this fourth condition as the balancing condition from here on.)

Note that the balancing condition is established to reduce scan-test time as much as possible, as scan-test time usually depends on the maximum number of FFs in a scan-path.

III. Algorithms

Our multi-scan-chain optimization method includes assignment of SIPs and SOPs to scan-paths (pairing of SIPs and SOPs), assignment of FFs to the scan-paths, and optimization of connection-order of FFs in each scan-path. We propose a four-phase algorithm framework (Fig.2) for the multi-scan-chain optimization problem as follows:

Phase 1: Determine SIP-SOP pairs. (Each pair is assigned to a scan-path without duplication.)

Phase 2: Assign FFs to scan-paths.

Phase 3: Determine connection-order of FFs in each scan-path.

Phase 4: Reduce the total scan-path length by exchanging FFs among scan-paths.

For each phase we propose one or two methods, which are described in the following sections.

A. Phase 1: Determination of SIP-SOP pairs

If SIP-SOP pairs are not specified, it is necessary to determine them appropriately. We propose a method of determining SIP-SOP pairs which minimizes the maximum distance between SIP and SOP.

When FFs are not assigned to scan-paths, it is hard to determine SIP-SOP pairs appropriately so that total scan-path length is minimized. Intuitively, however, long SIP-SOP distance might have a bad influence on the scan-path length. We therefore intend to minimize the maximum SIP-SOP distance.

The proposed method (for Phase 1) shown in Fig.3 is described as follows:

First, a list $\Lambda$ of all the pairs of SIP and SOP is constructed, and $\Lambda$ is sorted in the increasing order of SIP-SOP distance. Next, the first pair in $\Lambda$ is added to a list $\Theta$ (Fig.3(a)), which is initially empty, and the pair is deleted from $\Lambda$. This operation is executed one by one according to the order in $\Lambda$ (Fig.3(b)). After each operation, if the pairs in $\Theta$ have a complete SIP-SOP matching, the algorithm terminates (Fig.3(c)). Clearly this method gives the optimal solution in terms of minimizing the maximum SIP-SOP distance.

B. Phase 2: Assignment of FFs to scan-paths

In order to minimize scan-path length, it is desirable that FFs included in the same scan-path should be placed within a small area. For this purpose, we propose a method for Phase 2 which formulates the problem into a minimum cost flow problem and employs a conventional minimum cost flow algorithm [9]. We call the method the Minimum-Cost-Flow-based Method (MCFM).

In this method, we define distance $d_p(i,j)$ between an
FF $i$ and a scan-path $j$ as the minimum Manhattan distance between $i$ and any point on the line segment from SIP to SOP of $j$ (Fig.4), where the location of an FF is defined as the center of the FF’s input and output pins.

In the following we explain MCFM using Fig.5.

MCFM first calculates the distance $d_p$ for all pairs of FF and scan-path. Next MCFM constructs a graph $G_f$, which is described as follows (Fig.5):

Let $F$ be a set of all FFs and $P$ be a set of all scan-paths. $G_f$ is a graph where

- each component of $F$ and $P$ is represented as a vertex,
- in addition to the $F$ and $P$ vertices, there are one source vertex and one sink vertex,
- and there are directed edges from the source vertex to all vertices in $F$, from each vertex in $F$ to all vertices in $P$, and from each vertex in $P$ to the sink vertex.

Let $E_1, E_2, E_3$ be a set of all edges from the source vertex to vertices in $F$, from vertices in $F$ to vertices in $P$, and from vertices in $P$ to the sink vertex, respectively. Then, MCFM specifies cost $c_s$ and upper and lower limits of flow $f_u, f_l$ as follows:

- $c_s = 0, f_u = 1, f_l = 0$, for all edges in $E_1$,  
- $c_s = d_p$ (for corresponding FF and scan-path), $f_u = \infty, f_l = 0$, for all edges in $E_2$,  
- $c_s = 0, f_u = n_1, f_l = n_1$, for all edges in $E_3$, where $n_1 = \lceil N_f/N_p \rceil$, $N_f$ is the number of FFs, $N_p$ is the number of scan-paths, and $n_1 = \lceil N_f/N_p \rceil$. (This means that the difference between $f_u$ and $f_l$ is at most one.)

Note that, for all edges $e$ in $E_2$, cost $c_s$ is specified as $d_p(i, j)$, where $i$ is the FF whose corresponding vertex is the starting vertex of $e$ and $j$ is the scan-path whose corresponding vertex is the ending vertex of $e$, and $n_1 (= \lceil N_f/N_p \rceil)$ and $n_2 (= \lceil N_f/N_p \rceil)$ are the minimum and maximum numbers of FFs that can be included in a scan-path under the balancing condition, respectively.

MCFM then finds the minimum cost flow of $N_f$ on $G_f$. To solve the minimum cost flow problem with positive upper and lower limits, we use Goldberg’s algorithm [5].

After solving the minimum cost flow problem, MCFM assigns each FF $i$ to the scan-path $j$ if and only if the edge $(e_i, e_j)$ has flow of one, where $e_i$ and $e_j$ are the corresponding vertex of $i$ and $j$ respectively.

It is easy to see that the minimum cost flow gives the minimized total sum of $d_p$. (Note that $d_p$ represents the suitability of assigning an FF to a scan-path, as described above.) As a result, MCFM gives the best solution from the point of view of $d_p$, although it does not necessarily ensure the minimum total scan-path length.

C. Phase 3: Determination of connection-order of FFs in each scan-path

The problem of determining the connection-order of FFs is formulated as a Traveling Salesmen Problem (TSP). The TSP can be classified into two types of problems: Symmetric TSP (STSP) and Asymmetric TSP (ATSP). An STSP is a TSP in which the length (cost) of an edge is independent of its direction, i.e., distance from node $A$ to node $B$ is always equal to that from node $B$ to node $A$. In an ATSP, the length (cost) of an edge depends on its direction, i.e., distance from node $A$ to node $B$ is not always equal to that from node $B$ to node $A$.

The problem of determining the connection-order of FFs can be formulated as either an STSP or an ATSP, depending on the way the wire length between two FFs is estimated. If we define the location of an FF as the center of the FF’s input and output pins and estimate the wire length by the Manhattan distance between the locations of FFs (Fig.6(a)), distance between two FFs is independent of signal direction, and therefore the problem is formulated as an STSP. (We refer to this estimation
formula as the symmetric formula hereafter.)

On the other hand, if we consider the locations of FF pins and estimate the wire length by the Manhattan distance from the output pin of one FF to the input pin of another FF (Fig. 6(b)), distance between two FFs would depend on signal direction. In this case, the problem should be formulated as an ATSP. (We refer to this estimation formula as the asymmetric formula from here on.)

Adopting STSP heuristics to determine the FF connection-order has the advantage of fast operation, while adopting ATSP heuristics has the merit of accurate estimation of wire length. To make the most of both methods, we propose a method for Phase 3 which combines STSP heuristics with an ATSP heuristic.

In this method, the problem is first formulated as an STSP, and an initial connection-order is constructed using one of the methods for solving STSPs called the Greedy Method [6]. Then, the method improves the initial connection-order through use of the 3-opt Method [7], which is a fast and effective improvement heuristic for STSPs. Finally, the problem is formulated as an ATSP in order to estimate distance between FFs more precisely, and an algorithm called the Asymmetric 3-opt Method, which we propose as a variant of the 3-opt Method, further improves the connection-order.

Fig. 7 shows the flow of our method for Phase 3.

The Greedy Method [6] is a constructive method which is used to construct an initial solution in the proposed method for Phase 3. It finds the nearest pair of FFs and creates an edge between them (Fig.8), unless the edge makes an unacceptable solution (e.g. three edges from one FF). Greedy Method iterates these operations until a scan-path is completed.

The 3-opt Method [7] removes three edges from the current solution and then generates three edges so that the resulting solution satisfies the scan-path conditions (Fig. 9). The 3-opt Method iterates this improvement until no further improvement is obtained. Generally, this method efficiently achieves a near-optimal solution. Note that in the Greedy Method and 3-opt Method, the distance between two FFs is measured by the symmetric formula.

The Asymmetric 3-opt Method improves the solution by the same operation as that of the 3-opt Method, except that the distance between two FFs is measured by the asymmetric formula and that none of the edges may change their directions in the updating (Fig.10).

D. Phase 4: Inter-path exchange of FFs

This phase improves scan-path length by exchanging FFs among scan-paths. We propose two methods for Phase 4. The first method, which we call the Exchange-Permutation Method (EPM), iterates by exchanging FFs and applying the Asymmetric 3-opt Method, and the other is based on Tabu Search [3][4].
D.1 Exchange-Permutation Method

This method first iterates exchanges of any two FFs in different scan-paths until scan-path length cannot be improved any further by such exchanges. (We refer to this phase as the exchange phase below.) If the exchange phase does not exchange any FFs, EPM is terminated (Termination Condition 1). Otherwise, the method permutes the connection-order of FFs in each scan-path by using the Asymmetric 3-opt Method so as to improve the length of each scan-path. (We refer to this phase as the permutation phase below.) If the permutation phase does not change connection-order of any scan-path at all, EPM is terminated (Termination Condition 2). Otherwise, after the permutation phase, the exchange phase is executed again. Iteration of these two phases, exchange and permutation phases, continues until one of the above Termination Conditions is satisfied. The exchange phase shown in Fig.11 can be described as follows:

The exchange phase repeatedly exchanges pairs of FFs included in different scan-paths, which improves the solution. An exchanged FF is inserted into the destination scan-path at the position where the increase in the scan-path length caused by the insertion is minimized (Fig.11). This exchange is iterated until no further improvement can be obtained.

D.2 Tabu-Search-based Method

The second method for Phase 4 is based on the conventional Tabu Search algorithm [3][4]; we call it the Tabu-

Search-based Method (TSM). TSM iterates by exchanging two selected FFs a specified number of times, and selects the best solution during the iterations.

TSM first holds all scan-paths as the current-scan-path set \( S \), and also as the best-scan-path set \( S^* \). Next it finds a pair of FFs in \( S \) which gives the maximum gain and exchanges the pair, where gain is defined as a decrease in the total scan-path length by the exchange. Note that even if the best gain is less than 0, (which means the scan-path length increases), the exchange is executed anyway. On the occasion of exchange, one FF of the selected pair is inserted into the destination scan-path at the position where the other FF is removed (Fig.12). After an FF is removed from a scan-path, the FF is prohibited from returning to the scan-path which included the FF during iterations of the user-specified number \( n_{it} \) of times.

After an exchange, the best solution \( S^* \) is replaced by the current solution \( S \) if the total scan-path length of \( S \) is shorter than that of \( S^* \).

TSM iterates such exchanges until the number of iterations reaches the user-specified number \( n_{max} \) and after every \( n_{pc} \) (which is also specified by the user) iterations, it optimizes the connection-order of FFs in each scan-path by the Asymmetric 3-opt Method.

The best solution found during the iterations is always preserved in \( S^* \). When the number of iterations reaches \( n_{max} \), TSM is terminated and the best solution \( S^* \) is selected as the resultant solution.

IV. Experimental Results

To evaluate our algorithms, we implemented them on a 179MIPS UNIX machine in C language. In evaluating the algorithms, we use actual design data with a single scan-path, to which are added nine SIPs and nine SOPs. (Thus giving the data ten scan-paths, and accordingly scan-test time is reduced by approximately 90%). The locations of the added SIPs and SOPs on the layout plane are determined randomly. Note that, in actual designs which employ multiple-scan design methodology, the numbers
and locations of SIPs and SOPs are specified by designers, and our algorithm accepts any numbers and locations of SIPs and SOPs.

Table I shows details of the data used in the experiments. In this table, #FF is the number of FFs, and Initial SPL is the single scan-path length [mm] before scan-chain optimization. The scan-path length is equal to the total wire length on the scan-path estimated by the asymmetric formula (described in section 3.3). Single SPL in Table I means the single scan-path length [mm] after optimizing the connection-order of the initial scan-path (by the proposed method for Phase 3).

As we presented two methods in Phase 4, two algorithms A and B are implemented and evaluated. These algorithms use the same methods from Phase 1 through Phase 3, and Algorithm A employs EPM in Phase 4, while Algorithm B employs TSM in Phase 4.

When using TSM, we specified \( n_{FF} = 10 \), \( n_{PC} = 10 \), \( n_{max} = 1000 \).

<table>
<thead>
<tr>
<th>data</th>
<th>#FF</th>
<th>Initial SPL [mm]</th>
<th>Single SPL [mm]</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSC1</td>
<td>7437</td>
<td>1685.1</td>
<td>889.7</td>
</tr>
<tr>
<td>MSC2</td>
<td>8120</td>
<td>1806.9</td>
<td>888.7</td>
</tr>
<tr>
<td>MSC3</td>
<td>9063</td>
<td>2185.6</td>
<td>1009.6</td>
</tr>
</tbody>
</table>

Table II shows the experimental results. In this table, TSPL means total scan-path length [mm] and Time means computing time [sec].

In this experiment, the initial scan-path length is greatly reduced by 44-52% (47% on the average). Even if compared with single scan-chain optimization, the proposed algorithms result in only a 5-8% (7% on the average) increase in scan-path length in spite of the disadvantage of having many extra SIPs and SOPs. (Generally, the more scan-paths there are, the more severe the constraints of the problem become, which leads to an increase in the total scan-path length.)

Comparing Algorithms A and B, Algorithm A yields slightly better results and takes much less computing time.

V. Conclusions

In this paper, we have proposed a four-phase algorithm framework for the scan-chain optimization problem in multiple-scan design methodology. The proposed method first determines pairs of scan-in and scan-out pins and each pair is associated with a scan-path (Phase 1). Then, flip-flops are assigned to the scan-paths evenly (Phase 2), and connection-order of flip-flops in each scan-path is optimized (Phase 3). Finally improvement by exchanging flip-flops among scan-paths is executed (Phase 4).

We also proposed algorithms which were implemented on a UNIX machine and evaluated by experiments using actual design data which included more than 7,000 flip-flops. In the experiments, scan-test time was reduced by approximately 90% and the total scan-path length was 44-52% (47% on the average) shorter than the initial scan-path length and only 5-8% (7% on the average) longer than the single optimized scan-path length.

As possible future improvement, adopting a more powerful TSP heuristic (e.g. Lin and Kernighan’s algorithm [7]) in Phase 3 might further improve the scan-path length, although the computing time would increase. Also, determining both the pairs of scan-in and scan-out pins and the assignment of flip-flops to scan-paths simultaneously, i.e., merging Phases 1 and 2 of the proposed algorithm framework could be an effective approach to obtaining better solutions.

References


