Layout Driven Selecting and Chaining of Partial Scan Flip-Flops*

Chau-Shen Chen Kuang-Hui Lin TingTing Hwang
Department of Computer Science, Tsing Hua University
Hsin-Chu, Taiwan 30043 R.O.C.

Abstract – In an era of sub-micron technology, routing is becoming a dominant factor in area, timing, and power consumption. In this paper, we will study the problem of selecting and chaining of scan flip-flops with the objective of achieving minimum routing area overhead. Most of previous work on partial scan has put emphasis on selecting as few scan flip-flops as possible to break all cycles in S-graph. However, the flip-flops that break more cycles are often the ones that have more fanins and fanouts. The area adjacent to these nodes is often congestive in layout. Such selections will cause layout congestion and increase in number of tracks to chain the scan flip-flops. We propose a matching-based algorithm to perform simultaneously the selecting and chaining of scan flip-flop taking layout information into account. Experimental results show that our approach outperforms the traditional one in final layout area.

1 Introduction

Scan design methodology as design for testability (DFT) has become of a popular technique for sequential circuits [1]. It adds test mode control signals to circuit, connects flip-flops to form a shift register in test mode, and makes primary input/output of test shift register controllable and observable, thus providing controllability and observability of internal state variable for testing.

Three types of scan method have been proposed: full scan, random access, and partial scan. Among them, the partial scan which leads to low overhead is the most popular of scan methodology. In order to maintain both low complexity of test generation and low overhead, scan flip-flops must be selected carefully in partial scan design. Cheng and Agrawal [2] have proposed a cycle-breaking technique to select scan flip-flops by which test generation complexity of the resultant circuits grows linearly with the sequential depth and the overhead is about 25% of full scan. Later, the cycle-breaking algorithm was improved by [3] and [4]. Taking timing into consideration, Jou and Cheng [5] have proposed a timing-driven partial scan system which aims at reducing area overhead and performance degradation caused by added test logic.

Most of previous work [1, 2, 3, 4, 5, 6, 7] on partial scan has put emphasis on selecting scan flip-flops at logic level. The objective is to select the minimum number of flip-flops to break all cycles. However, in an era of sub-micron technology, routing is becoming a dominant factor in area, timing, and power consumption. The selecting and chaining of scan flip-flops should take routing into consideration. However, in many cases, owing to not considering the routing information, the flip-flops which are in congested area may be selected by traditional method. To chain these flip-flops often cause the increase in routing track.

We use the following example to illustrate such case. Figure 1(a) shows the S-graph of a given circuit, where a vertex represents a flip-flop of the circuit and an edge $e_{i,j}$ exists if there is a combinational path from flip-flop i to flip-flop j. Figure 1(b) shows the result of placement and routing before scan flip-flops are selected. By traditional cycle-breaking approach, selection of flip-flops $\{B, G\}$, or $\{B, F\}$, or $\{B, E\}$ as scan flip-flops will produce the minimum number of flip-flop overhead and hence the best solution. Since the routing channel close to flip-flops B is congestive, to chain flip-flop B and flip-flop G (or B and F, or B and E) needs additional routing track. That is, the above-mentioned three best solutions will cause extra routing track overhead. However, if we take into account the layout information in scan flip-flops selection phase, the selecting of flip-flops C, D, and G will break all cycles in Figure 1(a) and chaining of them will not cause additional routing overhead. Moreover, the sequence of chaining of scan flip-flops should also be taken into account during the selection of flip-flops because different chaining sequence will cause different routing requirement. We take Figure 1 as example again. Given that the routing channel between the flip-flops D and G is congestive, although the Manhattan distance of flip-flop D and G is minimum among C, D, and G, we should avoid chaining flip-flop D and G so that no additional routing

*Supported in part by a grant from the National Science Council of R.O.C. under contract no. NSC-83-0404-E-007-021

track is needed. Therefore, the only feasible chaining sequence without increasing routing overhead is D–C–G in which D and C are connected through channel 1 and C and G through channel 2.

To achieve minimum routing area, we present a new approach to the selecting and chaining of scan flip-flops taking the layout information into account. Our approach incorporates the selecting step and the chaining step into one. To estimate the congestivity of routing channel, initial placement will be performed before scan flip-flops are selected. Iteratively, some flip-flops are selected and chained according to the congestivity of the present layout. And after each selecting and chaining of flip-flops the layout information will be updated accordingly.

The remaining of this paper is organized as follows. In Section 2, we introduce the system flow. In Section 3, we present the selecting-chaining scan flip-flop algorithm. Experimental results are presented in Section 4. Finally, in Section 5, we draw a concluding remark.

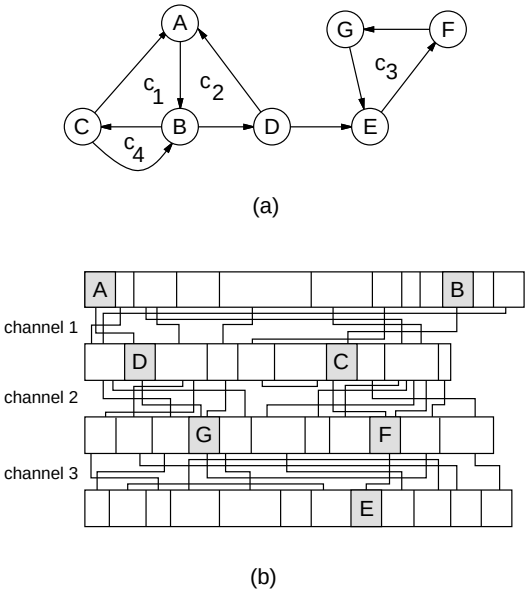

Figure 1: (a) S-graph of a given circuit, (b) Initial placement and routing before scan flip-flops are selected.

2 Two Design Flows

2.1 Selecting and Chaining of Flip-Flops without Layout Information

One traditional approach would decide the scan flip-flops before layout synthesis as shown in Figure 2(a). First, it selects scan flip-flops of a given circuit. Then, it replaces a selected flip-flop by a scan flip-flop. Next, it chains the scan flip-flops. Finally, it performs the place and route.

In selecting of flip-flop phase, since no layout information is provided, the objective is usually to select the minimum number of flip-flops to break all cycles. In chaining of flip-flop phase, the topology of scan flip-flops can be used in deciding the order, e.g.,the Depth First Search (DFS) order of the circuit. As was expected intuitively and will be demonstrated experimentally in Section 4, such approach proves not to be so effective because layout information is not taken into account.

2.2 Selecting and Chaining of Flip-Flops with Layout Information

Our system takes into account the layout information during the selecting and chaining scan flip-flops. Figure 2(b) depicts the design flow, *l.select*. It performs an initial placement and routing first. After placement and routing phase, iteratively, it selects some flip-flops as scan flip-flops and chain them. Then, after each selecting and chaining of flip-flops, the layout information will be updated accordingly. Such procedure will continue till all cycles in the circuit are broken and all scan flip-flops are chained. The input to our system is a sequential boolean network. The output from our system is a circuit with partial scan. TimberWolf tool [9] will be the placement and routing tool.

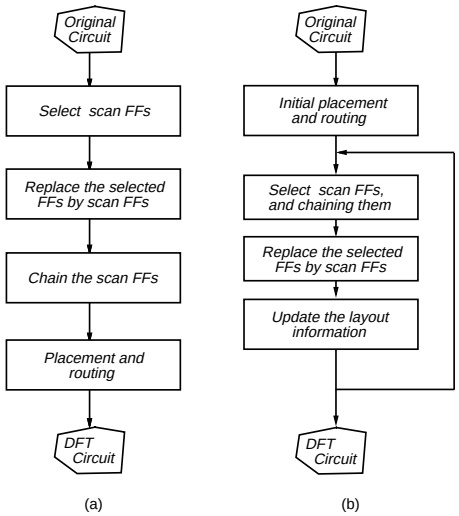

Figure 2: (a)One traditional partial scan design flow, (b)Layout driven selection and chaining of partial scan design flow.

3 Layout Driven Algorithm of Selecting and Chaining Scan Flip-Flops

To reduce test generation complexity, the scan flip-flops should be selected to break all cycles. To incur minimum area overhead, flip-flops should be selected so that minimum number of flip-flops can be selected and chaining of them can lead to minimum routing overhead. To this end, one objective function of selecting scan flip-flops will be to break all cycles with as few scan flip-flops selected as possible. The other objective function is to select flip-flops so that the chaining of flip-flops will not

incur extra routing track. With these two objective functions, we propose a two-phase algorithm taking layout information into consideration. The first phase is a cycle-relation-construction one that is used for selecting as few scan flip-flops as possible. The second phase is a selecting and chaining flip-flops phase that takes into account layout information by maximum-weighted matching.

3.1 A Matching-Based Selecting and Chaining of Flip-Flops

Selecting minimum number of flip-flops to break all cycles in S-graph is the feedback vertex set problem which is NP-complete [12]. To tackle this NP-complete problem efficiently, we will propose a heuristic based on the notion of selecting a flip-flop by which maximum cycles can be broken. Thus the first phase of our algorithm is a cycle-relation-construction one. For each flip-flop, we compute the cycles which will be broken when the flip-flop is selected. These cycles form the *cycle list* of the flip-flop in this phase. A depth-first-search (DFS) traversal is used in which a stack is utilized to record the traversal sequence for each primary input. When a new flip-flop is visited, the flip-flop is pushed into stack. When a flip-flop in the stack is visited again, a cycle is found. Flip-flops are popped out from the stack when the backtrack is performed. The cycle information will be recorded for all flip-flops in the cycles. By this method, for each flip-flop, we can record all cycles passed by the flip-flop.

In our approach, flip-flops are selected and chained incrementally in each iteration so that the updated layout information can be used in guiding the selection in the next iteration. In each iteration, to ensure that the cycles broken by the selected flip-flops do not overlap (the selected flip-flops break different sets of cycles), we partition the flip-flops into several Connected Component (CC). Let CL_{v_i} (Cycle List) denote the cycles broken by flip-flop v_i when the flip-flop v_i is selected as scan flip-flop. We define formally Connected Component, CC_1, CC_2, $\cdots$, CC_m, as a partition of flip-flops to satisfy

$$v_i \in CC_k \text{ and } v_j \in CC_k \text{ iff } CL_{v_i} \cap CL_{v_j} \neq \emptyset \quad \text{for } k \in \{1, 2, \cdots, \mathrm{m}\}.$$

where v_i and v_j are flip-flops in the circuit. The CC list is constructed as follows. Initially, CC list is empty. We pick flip-flops one by one from the flip-flop list to check if the flip-flop should be included in a Connected Component. Let the flip-flop v_i be checked to see if it be included in Connected Component CC_j. If the cycle list of flip-flop v_i and that of any flip-flop in CC_j contain a common cycle, flip-flop v_i should be included in CC_j. However, if there is no CC containing a flip-flop whose cycle list contain a common cycle as the checked flip-flop, a new CC should be created. The checked flip-flop is included as the first flip-flop in the newly created CC.

We take the S-graph shown in Figure 1(a) as example. First, we compute the cycle lists of the flip-flops. The set of cycles broken by each flip-flop are $CL_{v_A} = \{ C1, C2 \}$, $CL_{v_B} = \{ C1, C2, C4 \}$, $CL_{v_C} = \{ C1, C4 \}$, $CL_{v_D} = \{ C2 \}$, $CL_{v_E} = CL_F = CL_G = \{ C3 \}$. To construct the CC list, we pick up flip-flops one by one for processing. Initially, CC_1 is created to have only one element v_A. We then process v_B. Since both two cycles, $C1$ and $C2$, are on the cycle lists of v_B and v_A, v_B is included into CC_1. Similarly, v_C and v_D are included to CC_1 sequentially. Since the cycle list of v_E does not contain common cycle on the cycle lists of v_A, v_B, and v_C, we create a new Connected Component CC_2 and include v_E to CC_2. Finally, all flip-flops are partitioned into two Connected Components, $CC_1 = \{ v_A, v_B, v_C, v_D \}$, $CC_2 = \{ v_E, v_F, v_G \}$.

After all flip-flops are partitioned into several Connected Components, we would select a flip-flop from a Connected Component and chain the selected flip-flops. In the case of one Connected Component, we select the flip-flop which breaks the maximum number of cycles as the scan flip-flop. Thus, the Connected Component will be spilt into many Connected Components. To incur minimum layout overhead derived from the larger cell area of a scan flip-flop, we would select a minimum number of flip-flops as scan flip-flops. In that case, we would select the flip-flop in each Connected Component which breaks maximum number of cycles. However, the selected flip-flops need to be chained. The node which breaks more cycles is often the one which has more fanins and fanouts. The area adjacent to node is often congestive. To consider the routing overhead, the flip-flop in each Connected Component which breaks the maximum number of cycles may not be the best choice.

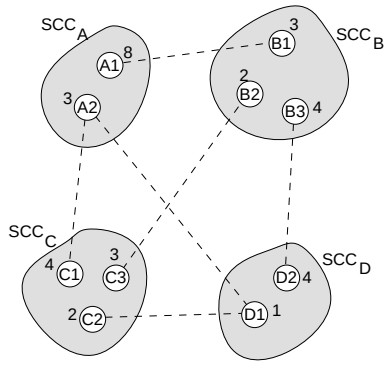

Figure 3: Matching graph considering the layout information.

We take Connected Components, CC_A and CC_B, in Figure 3 to illustrate this case. There are two flip-flops, A_1 and A_2, in CC_A and three flip-flops, B_1, B_2, and B_3, in CC_B. The numbers of cycles broken by flip-flops A_1, A_2, B_1, B_2, and B_3 are 8, 3, 3, 2, and 4, respectively. If we are to select one flip-flop from CC_A and one from CC_B and chain them, selecting A_1 and B_3 among the six combinations can break a maximum number of cycles in S-graph. However, if the density of the columns passed by the route connecting flip-flop A_1 and flip-flop B_3 is equal to the channel density, chaining the flip-flop A_1

and B_3 need increase one track. In terms of routing area incurred this selection is not the best.

Therefore, in selecting and chaining flip-flops from two Connected Components, we consider both the number of cycles broken and the congestivity of the route connecting the two flip-flops. We construct a weighted complete graph, where each vertex represents a Connected Component and where weights are defined on edges, for each pair of Connected Component, to reflect both the number of cycles broken and the routing congestivity. (The computation of weight will be discussed in Section 3.2.) For each pair of Connected Components, there are many selecting and chaining possibilities. The possibility with the best gain is selected as the weight on the edge. For example, Figure 3 is a constructed complete graph of four Connected Component, CC_A, CC_B, CC_C, and CC_D. For CC_A and CC_B, among the six possible solutions, A_1–B_1, A_1–B_2, A_1–B_3, A_2–B_1, A_2–B_2, and A_2–B_3, A_1–B_1 will break maximum number of cycles without incurring extra routing overhead and its gain is defined on the edge of CC_A and CC_B.

After the complete graph is constructed, we will select one flip-flop from each Connected Component and chain them all. If we select one flip-flop from one Connected Component greedily, the resultant solution may not be satisfactory. For example, since selecting A_1 in CC_A and B_1 in CC_B will break the most flip-flops, A_1 and B_1 will be the first selection by a greedy method. Together, they will break 11 cycles. To break different set of cycles, the next solution will be C_2 in CC_C and D_1 in CC_D. Then the total number of cycles broken will be 14. However, if A_2, B_3, C_1, and D_2 are selected, the number of cycles broken is 15 in total. Therefore, to break more cycles from a global point of view, we propose to use a matching algorithm on the graph to select flip-flops.

Iteratively, the matching algorithm is performed on the complete graph of Connected Components. For each pair of component for matching, the two flip-flops corresponding to the weight on the edge are selected and chained. We take the example in Figure 3 again. Suppose CC_A is matched to CC_C, and CC_B is matched to CC_D by the matching algorithm. Then A_2 and C_1 are selected as scan flip-flops and chained together. Similarly, B_3 and D_2 are selected and chained.

After each iteration of matching, the computed graph of CC must be reconstructed. First, after some flip-flops are selected as scan flip-flops, many cycles in source S-graph are broken and these cycles can be deleted from each flip-flop. Hence CC may be broken into many new CC's and weight recomputed among the newly constructed CC's. Second, after two flip-flops are selected as scan flip-flops and chained, only the ending vertices of a chain sequence can be connected to those of the other chain. Let a supervertex to represent a chain of vertices. For two supervertices, there are four possible connections between them. The maximum gain of all possible connections between two supervertices will be assigned as the weight on the edge of these two supervertices. Figure 4 shows an example of reassigning weight on the edge between two supervertices. In Figure 4(a), vertices $1, 2, 3$ and vertices $4, 5$ are combined to form two supervertices, $v_{\{1,2,3\}}$ and $v_{\{4,5\}}$, respectively. Only the ending vertices v_1 and v_3 of supervertex $v_{\{1,2,3\}}$ can be connected to the ending vertices v_4 and v_5 of supervertex $v_{\{4,5\}}$. Four gains g_1, g_2, g_3 and g_4 resultant from the four connections, v_1–v_4, v_1–v_5, v_3–v_4, v_3–v_5 are computed. Figure 4(b) shows that the weight on the edge of the two supervertices is the maximum value of g_1, g_2, g_3 and g_4.

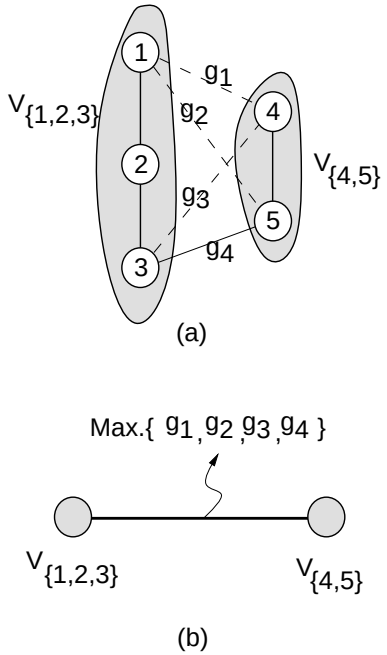

Figure 4: (a) Possible connections between two supervertices (b)Maximum gain assigned as the weight on the edge of two supervertices.

The overall *l.select* algorithm is shown in Figure 5 and Figure 6.

3.2 Defining Weight on the Edge

The weight on the edge can reflect the area overhead, the timing cost, and power consumption when the circuit operates in the test mode. Since the circuit usually operates at slow clock rate in test mode, timing issue is not so critical. But the area caused by the test logic will affect the overhead, and power consumption which is proportional to the capacitance load. Therefore, we will consider the minimization of area in this section.

The first consideration is the number of cycles broken by each flip-flop. As flip-flop v_x is selected as scan flip-flop, we define $W_1(v_x)$ as the number of cycles that are broken.

$W_1(v_x)$ = *the number of cycle broken when* v_x *is selected as scan flip flop*,

The function, W_1, will result in the selecting of flip-flops which break maximum number of cycles. As we

Algorithm *l.–select*(G)
Input:G is a S-graph;
Begin
construct *Cycle List* by DFS traversal;
construct *Connected Component;*
while $|CC| = 1$
select one flip-flop which breaks the maximum cycles as scan flip-flop;
re-construct the *Connected Component;*
endwhile
Construct the complete graph $G = (V, E, W)$,
Matching$(G = (V, E, W))$;
End

Figure 5: The *l.select* algorithm.

Algorithm *Matching*$(G = (V, E, W))$
Input:G is a complete graph with weight on each edge;
Begin
while ($(|V| > 1)$ **then**
Perform maximum weighted matching on G;
for each pair of matched vertices i and j **do**
Select flip-flop from verteies i and j;
Chain flip-flop i an j;
endfor
Reconstruct the complete graph;
endwhile
End

Figure 6: The matching-based algorithm.

pointed out in Figure 1, the flip-flops which break maximum number of cycles are usually in a congested area. To chain these flip-flops may incur extra routing track. To consider the routing cost, we should take layout information into account. Since an initial placement and routing are performed before scan flip-flops are selected, layout information can be used in guiding the selecting of flip-flops.

Figure 7 illustrates this situation. There are two channels among three rows of cells, where the channel densities of channel 1, 2 are 5, 4, respectively. To measure the routing overhead, Manhattan distance between flip-flops can be used. However, Manhattan distance does reflect distance of the route but not congestivity. Suppose that flip-flops A, B, and C are to be chained. If the Manhattan distance is used, the flip-flops will be chained as A–B–C. In this case, the channel density of channel 1 has to be increased to 6 to accommodate the connection of A and B because the maximum column density between flip-flops A and B is equal to the channel density. That is, an extra track is required for channel 1. However, if the flip-flops are chained as B–C–A, the Manhattan distance is larger but no extra routing track is required for both channel 1 and channel 2 to complete these connections. Therefore, to save the routing overhead, flip-flops should be chained on the basis of congestivity of the layout plane. The column density of the initial placement and routing will be used in estimating the congestivity of the layout plane.

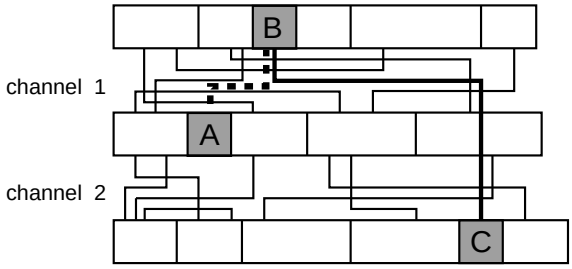

Figure 7: Flip-flops chained using placement and routing informatio n.

We assume that feedthrough and only one channel will be used in connecting two flip-flops. Since the flip-flops may not be located at adjacent rows, more than one channels can be considered to connect the two cells. For channel selecting, the following heuristics are used. If two cells are in the adjacent rows, the channel between these two cells is considered. If two cells are in the same row, the up channel and the lower channels are considered. If the cells are apart from each other by more than one channel, the channels between the two cells are considered. Therefore, for each pair of scan flip-flops, v_i and v_j, and a channel c under consideration for routing, we define the following weight as,

$$W_2(v_x, v_y, c) = W_1(v_x) + W_1(v_y) + \alpha \times MIN_{k \in D}\{channel\_density_c - column\_density_k\}$$

where D are the columns between v_x and v_y, $column\_density_k$ is the column density of column k in channel c and α is a constant. The first two terms are the number of cycles broken by flip-flop v_x and v_y. The third term denotes the number of tracks which can be used without increasing channel density. Finally, the weight defined on the edge for vertices, v_x, v_y, is the least congested channel from channels under consideration for routing. It is defined as,

$$W_3(v_x, v_y) = MAX_{c \in C}\{W_2(v_x, v_y, c)\}$$

where C are the channels selected using the channel selecting heuristic described above.

Finally, the weight on edge connecting of CC_i and CC_j is the maximum $W_3(v_x, v_y)$, for each pair $v_x \in CC_i$ and $v_y \in CC_j$.

$$W(CC_i, CC_j) = MAX_{x,y}\{W_3(v_x, v_y)\},$$
for each $v_x \in CC_i$ and $v_y \in CC_j$, $i \neq j$.

4 Experimental Result

The algorithms described in previous sections have been implemented as *l.select* in C and executed on a Sun workstation. Several experiments have been conducted to investigate their effectiveness. A subset of benchmark circuits from ISCAS is selected for that purpose. The circuits whose selected scan flip-flop is less than 20 are excluded in the selection because the interconnection required to route the flip-flops is too insignificant as compared with the area for the whole circuit. We compare the result of *l.select* as indicated and that of the traditional design flow. For the latter, the experiment begins with selecting the scan flip-flops using the method by [4]. The scan flip-flops are then chained using topology information of the S-graph. Finally, placement and routing are performed.

TimberWolf [9] is used as the placement and routing tool. The output of TimberWolf includes the number of rows in layout, the length of each row, and the total number of tracks. We will use the library [10] in which the cell height is 58 μm, and the routing pitch is 8 μm. Therefore, the area needed will be estimated by the following formula

$$Area = [58 \times (the\ no.\ of\ rows) + 8 \times (the\ total\ no.\ of\ tracks)] \times the\ length\ of\ the\ longest\ row.$$

The first table is to compare the number of routing tracks by using two design flows. Table 1 shows the comparisons. The first column is the names of circuits. Columns label $\#\ FF$ and *initial track* are the number of total flip-flops and the number of tracks required without considering the scan design. Columns label $\#\ SFF$ and *track* give the number of selected scan flip-flops and the routing tracks of a scan design, respectively. It is clear, as the table shows, from all the examples tested that, *l.select* selects more scan flip-flops but requires less number of routing tracks.

Table 1: The comparisons of routing track overhead.

Circuit	#FF	initial track	tradition		l.select	
			#SFF	track	#SFF	track
s1423	74	106	21	115	29	113
s5378	179	272	30	352	31	296
s9234	228	418	53	501	60	445
s13207	669	529	64	627	70	565
s15850	597	591	95	690	104	601

To understand the total area overhead incurred by two design flows, we compare them in final layout area. Table 2 shows the comparisons. Column *init. area* lists the layout area without partial scan. Column label *area* and *overhead* give the layout area and percentage of overhead to the initial area (without partial scan), respectively. The table shows clearly that *l.select* outperforms traditional design flow from the listed examples and that the larger the design, the less amount overhead will be incurred by *l.select* than by the traditional design flow.

5 Conclusions

In an era of sub-micron technology, routing is becoming a dominant factor in area, timing, and power consumption. In this paper, we have studied the problem of selecting and chaining of scan flip-flops with the objective of achieving minimum area overhead. A new design flow taking layout information into account is proposed. A matching based algorithm is used to select and chain scan flip-flops in one phase. Experimental results show that our algorithm is very effective in comparison with the traditional design flow in reducing layout area overhead.

Table 2: The comparisons of layout overhead.

Circuit	init. area	tradition		l.select	
		area	overhead	area	overhead
s1423	426	462	8.5%	448	5.1%
s5378	1751	2055	17.4%	1857	6.0%
s9234	3443	3926	14.0%	3657	6.2%
s13207	5433	6207	14.3%	5725	5.4%
s15850	6292	7166	13.9%	6596	4.8%

References

[1] V. D. Agrawal, K.-T. Cheng, D. D. Johnson, and T. Lin, "Designing Circuits with Partial Scan," *IEEE Design & Test of Computer*, pp. 8-15, April 1988.

[2] K.-T. Cheng, and V. D. Agrawal, "A Partial Scan Method for Sequential Circuits with Feedback," *IEEE Trans. on Computers*, pp. 544-548, April 1990.

[3] D. H. Lee, and S. M. Reddy, "On Determining Scan Flip-Flops in Partial-Scan," *Proc. ICCAD-90*, pp. 322-325, May 1991.

[4] A. Bhawmik, C. J. Lin, K.-T. Cheng, and V. D. Agrawal, "Pascant : A Partial Scan and Test Generation System," *Proc. Custom Integrated Circuits Conf.*, May 1991.

[5] J.-Y. Jou, and K.-T. Cheng, "Timing-Driven Partial Scan," *Proc. Int'l Conf. on Computer-Aided Design*, pp. 404-407, Nov. 1991.

[6] D. Kagaris and S. Tragondas, "Partial Scan with Retiming," *Proc. Design Automation Conf.*, pp. 249-254, June, 1993.

[7] P. S. Parikh and M. Abramovici, "A Cost-Based Approach to Partial Scan," *Proc. of Design Automation Conf.*, pp. 255-259, June, 1993.

[8] M. Garey and D. Johnson, "Computers and Intractability : A Guide to the Theory of NP-Completeness," *W. H. Freeman And Co.*, 1979.

[9] "TimberWolf : Mixed Macro / Standard Cell Floorplaning, Placement and Routing Package," *Yale University*, Sep. 1991.

[10] MCNC 2.0 standard cell data book, Mar., 1993.

[11] C. H. Papadiminitriou and K. Steiglitz, "Combinatorial Optimization : Algorithms and Complexity," *Prentice-Hall Publishing Co., Inc.*, N. J. 1982.

[12] R. R. Karp, "Reducibility Among Combinatorial Problems," R.E. Miller and J.W. Thatcher,(eds.), *Complexity of Computer Computations, Plenum Press, New York*, pp. 85 – 103, 1972.