Selecting Partial Scan Flip-Flops for Circuit Partitioning

Toshinobu Ono

NEC Corporation
Kawasaki, JAPAN

Abstract

This paper presents a new method of selecting scan flip-flops (FFs) in partial scan designs of sequential circuits. Scan FFs are chosen so that the whole circuit can be partitioned into many small subcircuits which can be dealt with separately by a test pattern generator. This permits easy automatic test pattern generation for arbitrarily large sequential circuits. Algorithms of selecting scan FFs to allow such partitioning and of scheduling tests for subcircuits are given. Experimental results show that the proposed method makes it possible to generate test patterns for extra large sequential circuits which previous approaches cannot deal with.

1 Introduction

Partial scan has been considered as one of the most realistic solutions to testing of large sequential circuits. Partial scan has possibility to give a circuit enough testability by means of small amount of extra hardware.

Several partial scan methodologies [1]–[9] have been proposed to obtain higher testability with smaller number of scan flip-flops (FFs). Among them, methods based on structural analysis [2, 3, 4, 6, 7, 8, 9] are showing relatively good empirical results with some simple heuristics. Most of them try to break cycles in the original circuits, most typically to break all the cycles except self loops. However, even after all the loops except self loops are cut, it still takes much time to generate test vectors for very large circuits, such as ones with hundreds of FFs.

This paper proposes a new partial scan method for arbitrarily large circuits. It makes circuits have enough testability regardless of their scale. With partial scan, circuits are partitioned into many small subcircuits which can be easily handled by the sequential test pattern generators available today. Since the size of each subcircuit can be set reasonably small, test pattern generation can be well in control no matter how large the whole circuit is.

Its another advantage is that the length of test patterns can be reduced. Vectors for two or more subcircuits can be applied simultaneously if they do not share any primary input. This allows test vector compaction even for sequential circuits.

In the following sections, first a new scheme of circuit partitioning for test pattern generation is given. Next, a method of partial scan FF selection is presented to allow partitioning of a circuit into small subcircuits. Then, some experimental results are shown, followed by conclusion and future work.

2 Partitioning for Test Generation

A partitioning method for sequential circuit is proposed. A sequential circuit can be partitioned into subcircuits so that a test pattern generator can handle each of them separately. In general, difficulty of test pattern generation may possibly increase exponentially according to the circuit size. Therefore, significant reduction in test pattern generation time can be achieved by generating test patterns for partitioned circuits.

2.1 Conditions for Partitioning

It is necessary to guarantee that test vectors generated separately for each subcircuit can be applied to the whole circuit. In order to allow valid but independent test pattern generation for each subcircuit, partitioning must be done under the following conditions.

- Every primary input (PI)/primary output (PO) of the subcircuits must be a PI/PO of the whole circuit.
- All the gates and PIs reachable to a PO in the whole circuit must be included in the subcircuit that has the PO.

Subcircuits thus obtained may not be necessarily disjoint. Subcircuits may share gates or PIs. Such overlapping allows more partitionings. However, POs should not be shared by different partitions.

Overlapping of subcircuits may cause overhead in test pattern generation because overlapped areas are processed more than once. However, overhead can be suppressed minimally by avoiding test generation for faults which have been detected in other subcircuits. This will be discussed later.

2.2 Partitioning Example

Figure 1 shows an example of partitioning. In this case, one sequential circuit, which has six FFs, can be partitioned into three subcircuits with some overlap. Test vectors generated for partitions A and C can be applied simultaneously because the two circuits do not share any PIs.

To illustrate the concept of the partitioning more easily, a graph description of circuits is introduced. A sequential circuit is represented as a graph called dependency graph. A dependency graph is defined as follows:

Definition 1 (Dependency Graph) *A dependency graph is a directed graph with nodes corresponding to either the FFs or the POs of the circuit. There is an arc from node A to node B if and only if there is a combinational path from FF A to FF/PO B in the circuit.*

Figure 1: Example of partitioning

Partitioning can be considered in dependency graphs instead of gate level circuits. Partitioning of the dependency graph can be translated to that of the gate level circuit without ambiguity. In the following sections, dependency graphs are used to represent sequential circuits and partitioning is considered in dependency graphs.

3 Scan FF Selection for Partitioning

3.1 Objective and Modeling

Generally, each partition may not be small enough for test pattern generation. In such a case, partial scan is used to reduce the size of large partitions. To scan a FF means to delete the out-going arcs of the corresponding node in the dependency graph. This elimination of arcs sometimes allows partitioning that is not possible in the original graph.

Figure 2 illustrates how scan FFs enables further partitioning. (A) shows the dependency graph of a circuit, where six nodes correspond to FFs (denoted by rectangles) and one node corresponds to a PO (denoted by a triangle). This circuit can be partitioned into two subcircuits as shown in (B) by scanning the shaded FF.

Figure 2: Scanning a FF for partitioning

The objective of scanning FFs is to reduce the size of each subcircuit to a certain limitation. The size of a circuit is defined as the number of the FFs in it because it is assumed that the number of FFs is one of the most important factors that determine the difficulty in testing the circuit.

To formulate the problem, the weight of each node in a dependency graph and the weight of the graph itself are defined.

Definition 2 (Weight of Node) *The weight $w(n)$ of node n is defined as the number of the nodes, including itself, from which the node is reachable through directed arcs in the dependency graph.*

Definition 3 (Weight of Graph) *The weight $W(G)$ of dependency graph G is defined as follows:*

$$W(G) = \max_{n \in G} w(n)$$

Definition 4 (Scanned Graph) *Given dependency graph G and a set of nodes N, the scanned graph $S(G,N)$ is defined as the graph that is obtained by removing all the out-going arcs of the nodes in N from G.*

Graph G can be partitioned into subgraphs having no more than $W(G)$ nodes, and the subcircuits corresponding to the subgraphs have $W(G) - 1$ FFs or fewer. $S(G,N)$ is the graph that is obtained by scanning the FFs corresponding to the nodes in N.

To reduce the overhead for partial scan, the number of the scan FFs should be minimized. With the above definitions, the FF selection problem is formulated as follows:

> *Given a dependency graph G and a size limit s, find the smallest set of nodes N such that $W(S(G,N)) \leq s + 1$.*

Figure 3 shows the weight of the nodes and the graphs for the same circuit as figure 2. The numbers above nodes are the weight of the node. The original circuit (A) can be partitioned into subcircuits with as many as 6 FFs because $W(G) = 7$. After the nodes in $N = \{e\}$ are scanned, the result circuit (B) can be partitioned into subcircuits with 3 or fewer FFs because $W(S(G,N)) = 4$.

Figure 3: Weight of nodes and graphs

3.2 A Heuristic Algorithm

Since it requires exponential time to find the optimal solution, a scan FF selecting procedure using heuristics has been developed.

The purpose of scanning FFs is to reduce the weight of overweighted nodes. However, it is difficult to estimate the reduction in the weight of other nodes obtained by scanning a FF, especially for large and complicated graphs. Therefore, the structure of the graph is first simplified for easy analysis of weight reduction by scanning some FFs. Then, more FFs are chosen to be scanned to reduce the size of partitions. Finally, unnecessary scan FFs are restored to normal FFs.

The procedure consists of the following three phases.

1. Graph Simplification — Scan FFs to cut cycles
2. Partition Size Reduction — Scan FFs to break partitions larger than the size limit
3. Normal FF Restoring — Restore scanned FFs to normal ones under the size limitation

3.2.1 Phase 1: Graph Simplification

Scan FFs are chosen to cut cycles in the circuit. This makes the graph acyclic, that is, no pair of nodes depends each other. Since the weight of nodes is calculated according to the dependencies between nodes, it is much easier to analyze the weight reduction produced by scanning a FF for acyclic graphs than for ones with cycles.

Although several FF selecting methods have been proposed to remove cycles in the circuit [3, 4, 6, 7, 8, 9], simple heuristics are used in the current implementation.

All the strongly connected components (SCCs) are found in the dependency graph. Then, a node is chosen and removed from each of the SCCs that consist of two or more nodes. Selection continues until every SCC contains only one FF. As the result, all the cycles are eliminated except self loops and an acyclic dependency graph is obtained.

In selecting a node from a SCC, the node which has the largest number of arcs is chosen. The heuristics are simple, but produce solutions close enough to the optimal [6]. The optimal solution is not necessary here because the final target is not to break cycles.

3.2.2 Phase 2: Partition Size Reduction

In this phase, FFs are chosen and scanned in order to reduce the size of partitions larger then the given limit.

Other heuristics are introduced here. A cost function is used and the node having the highest cost is scanned one by one until the weight of all nodes becomes less than the limit. The cost of each node should reflect potential of weight reduction of other nodes in case it is scanned. If a node is scanned, its predecessor nodes lose weight. However, the predecessor nodes do not necessarily lose weight equal to the weight of the scanned node because there may be other paths from its ancestor nodes to its predecessor nodes. Therefore, not only the weight of nodes but also the graph structure should be considered. Before defining the final cost function, another function is introduced.

Definition 5 (Flow) *The flow $F(n)$ on node n is defined as follows:*

$$F(n) = \begin{cases} \displaystyle\sum_{m \in P(n)} \frac{F(m)}{S(m)} & \textit{(if } n \textit{ is a PO)} \\ \displaystyle\sum_{m \in P(n)} \frac{F(m)}{S(m)} + 1 & \textit{(if } n \textit{ is a FF)} \end{cases}$$

where $P(n)$ is the set of parent nodes of n and $S(m)$ is the number of child nodes of node m.

The flow $F(n)$ counts the probabilistic number of nodes reachable to n with the fanins and fanouts taken into account. Every node that corresponds to a FF has 1 as the original cost, and the flow on each node is divided and distributed equally to its fanout nodes. Since there is no cycle in the graph, calculation of $F(n)$ for every node n can be done easily. With this flow function, the cost function is now defined.

Definition 6 (Cost) *The cost $C(n)$ of node n is defined as the sum of the reduction in the flow on the nodes that have higher weight than the limit s if n is scanned.*

A node has higher cost if greater reduction in the weight of overweighted nodes is expected. The cost $C(n)$ of node n can be calculated as follows:

$$C(n) = \begin{cases} F(n) \cdot R(n) & (\text{if } W(n) \le s) \\ F(n) \cdot R(n) + 1 & (\text{if } W(n) = s + 1) \\ 0 & (\text{if } W(n) > s + 1) \end{cases}$$

where $R(n)$ is the number of nodes which are reachable from n and whose weight is higher than the given size limit s.

To reduce the weight of all overweighted nodes to less than the limit, at least one node must be scanned among the nodes with weight of $s + 1$ or less. Therefore, only the nodes with weight of $s + 1$ or less are considered in selection, that is, the cost of overweighted nodes is defined as 0. If a node has weight equal to $s + 1$, 1 is added to its cost because scanning it reduces not only the flow on its predecessor nodes but also its own flow.

In this phase, the cost $C(n)$ for every node is calculated. Then, the node with the highest cost is chosen as a scan FF, and the costs are calculated again for the new graph. Selection is repeated until no node has weight higher than the given partition size limit.

3.2.3 Phase 3: Normal FF Restoring

This is the final phase where some of the scanned FFs are restored to original normal FFs. Because of the non-backtracking selection in the previous phases, some scan FFs may not be necessary for the final objective, that is, to keep the size of partitions within the limit.

Each scan FF is checked if it needs to be scanned or not. Check is done by temporarily restoring the scan FF to a normal FF and calculating the weight of all nodes again. If some nodes become to have weight over the limit, the scan FF is left untouched. Otherwise, it is replaced with a normal FF. Although results depend on the order of the scan FFs being checked, its effects are not considered for the present.

Restoring some FFs may make cycles in the circuits again. Then, another option is provided here in order to keep the circuit acyclic. With this option selected, a scan FF which, if restored, will make a cycle is kept scanned. Since all the cycles are broken in the first phase, the final circuit has no cycle in this option.

This cycle breaking option is practically very useful. It means combination of the two methods based on cycle breaking and partitioning. Because the cycle breaking method can improve testability of each subcircuit, subcircuits may be reasonably large while guaranteeing enough testability. This results in reduction in the total number of necessary scan FFs.

3.3 FF Selection Example

FF selection in the second phase is demonstrated using an example. Figure 4 shows the flow and the cost of the nodes of graphs for the circuit used in the previous sections. Suppose that the size limit of partitions is 3. The numbers written above and below nodes denote the flow and the cost of the node, respectively. In the original graph (A), node e is selected and scanned because it has the highest cost 4.5. Comparing the graphs (A) and (B), the total flow reduction on the overweighted nodes is 4.5 because node e lost 1 and node g lost 3.5. The cost of node e, which was 4.5, exactly represents this flow reduction.

Figure 4: Flow and cost of nodes

4 Test Pattern Generation

Due to overlapping of subcircuits, test pattern generation may be performed more than once on some faults. If a fault in an overlapped area is not detected in a subcircuit, it must be tried in other subcircuits again. However, the faults which are previously detected in some subcircuit should be eliminated from the fault list for the subcircuits handled later.

Not only aborted faults but also ones declared redundant must be tried again in other subcircuits because they may be detectable on other POs in other subcircuits.

As for the order in which test pattern generation is performed, easier subcircuits should be handled earlier. Therefore, test pattern generation tasks are implemented in the increasing order on the number of FFs in each subcircuit.

5 Test Pattern Compaction

Test vectors for the entire circuit are built from the test vectors generated for the subcircuits. Because test vectors for each subcircuit have values required only on part of the PIs of the whole circuit, some vectors for different subcircuits can be applied simultaneously. Such test vector compaction reduces the number of test patterns for the entire circuit.

In general, a test sequence for a subcircuit generated by a test generation program cannot be broken into two or more separate sequences because most vectors in the sequence are generated depending on the previous vectors.

If testing of each subcircuit is considered as a job, and test length as time needed to do the job, test pattern compaction is regarded as a kind of scheduling problem. It is formulated as follows:

> *Given a set of jobs, time necessary to have each job done, and a set of job pairs that cannot be done simultaneously, schedule jobs so that the total time to finish all the jobs is minimal.*

The following heuristics are used to get a solution in reasonable time.

1. Begin a job as soon as it can be begun.
2. If two or more jobs can be begun, begin the job having the largest number of jobs that cannot be done with it.
3. If two or more jobs can be begun and have the same number of jobs that cannot be done with it, begin the job that requires the longest time.

6 Experimental Results

Experiments have been performed on some of the ISCAS89 sequential benchmark circuits [10]. Since the proposed method particularly targets very large circuits, only the circuits having more than 100 FFs are tried.

The number of partial scan FFs required for partitioning is examined first. The size limit of partitions is changed from 10 FFs per partition to 30. As a comparison, OPUS [4] is used to select FFs to break all cycles except self loops. Combination of the partitioning method and the cycle breaking method is also examined. The results are shown in Table 1.

Next, test patterns are generated for the partitioned circuits by HITEC [11]. Test pattern generation is performed both on the circuits with all the cycles broken by OPUS and the ones partitioned by the proposed method into subcircuits with no more than 30 FFs and no cycles. The programs run on SUN SPARC Station 10. The results are shown in Table 2. FC and FE denote the fault coverage and the fault efficiency respectively. Some numbers are not available because there is not enough memory for the test pattern generation program to process these circuits.

Although the cycle breaking method needs fewer scan FFs, it takes much longer time to generate test vectors, which detect only a small number of faults for some circuits. For some large circuits, it even cannot generate patterns. In contrast, with scan FFs selected by the proposed method, the test generator can produce much higher fault coverage for all the circuits. Since every subcircuit has only 30 FFs or fewer, it never has the problem of exhausting the memory space. For some circuits, the fault coverage may not be enough even with the proposed method. However, higher coverage can be obtained by changing the aborting conditions, which is 2 seconds per fault in this experiment.

The test vector compaction is examined next. Tests for the partitioned circuits are scheduled by the proposed method. Table 3 shows the results. In the table, the original length means the sum of the test pattern length of all the subcircuits. With the compaction, about 20 percent reduction is obtained at most. However, no or very

circuit	#FFs	cycle breaking		partitioning						partitioning + cycle breaking	
				≤10FFs/part		≤20FFs/part		≤30FFs/part		≤30FFs/part	
		#scan	%scan	#scan	%scan	#scan	%scan	#scan	%scan	#scan	%scan
s5378	179	30	16.8	95	53.1	69	38.6	57	31.8	58	32.4
s9234.1	211	55	26.1	130	61.6	106	50.2	86	40.8	106	50.2
s13207.1	638	58	9.1	267	41.9	210	32.9	174	27.3	217	34.0
s15850.1	534	91	17.0	289	54.1	233	43.6	203	38.0	216	40.5
s35932	1728	306	17.7	605	35.0	469	27.1	422	24.4	469	27.1
s38417	1636	380	23.2	913	55.8	788	48.2	687	42.0	750	45.8
s38584.1	1426	313	21.9	719	50.4	567	39.8	426	29.9	509	35.7

Table 1: FF Selection Results

little compaction can be done for some circuits. In these circuits, all or most subcircuits share some PIs.

circuit	cycle breaking			partitioning + cycle breaking		
	FC(%)	FE(%)	time(s)	FC(%)	FE(%)	time(s)
s5378	93.5	96.2	696	96.6	100.0	169
s9234.1	72.8	75.1	4350	88.5	94.1	1573
s13207.1	5.1	9.1	17836	55.9	60.8	7564
s15850.1	17.5	26.3	15378	81.2	89.4	3553
s35932	89.8	99.7	8264	88.4	100.0	253
s38417	N/A	N/A	N/A	53.5	54.9	6024
s38584.1	N/A	N/A	N/A	49.6	54.6	11756

Table 2: Test Generation Results

circuit	#partitions	original length	compacted length	reduction (%)
s5378	42	2871	2826	1.6
s9234.1	34	4027	3259	19.1
s13207.1	121	56037	48828	12.9
s15850.1	134	27775	24787	10.8
s35932	391	16778	16778	0.0
s38417	164	124468	159393	21.9
s38584.1	270	212847	212218	0.3

Table 3: Pattern Compaction Results

7 Conclusion and Future Work

A new approach to selecting scan FFs for partial scan design has been proposed. It selects FFs to be scanned so that the circuit can be partitioned into many small subcircuits. These subcircuits can be handled separately by any sequential test pattern generation program. This partitioning allows easy test pattern generation for arbitrarily large circuits.

An algorithm has been developed to select FFs effectively in reasonable time. Experimental results on large benchmark circuits have been given to show an advantage of the proposed method over the previous methods, especially for extra large circuits.

Future work includes enhancement of the FF selection algorithms to reduce the number of scan FFs, investigation of the efficiency in combination with other partial scan methods, and more evaluation with many other circuits.

Acknowledgements

The author would like to thank Prof. Janak Patel of University of Illinois for his support with the sequential test pattern generator HITEC and the partial scan tool OPUS.

References

[1] E. Trischler, “Incomplete Scan Path with an Automatic Test Generation Methodology”, *International Test Conf.*, pp.153-162, 1980.

[2] R. Gupta, R. Gupta and M. A. Breuer, “BALLAST: A Methodology for Partial Scan Design”, *International Symp. on Fault-Tolerant Computing*, pp.118-125, June 1989.

[3] K.-T. Cheng and V. D. Agrawal, “A Partial Scan Method for Sequential Circuits with Feedback”, *IEEE Trans. Computers*, Vol.39, No.4, pp.544-548, April 1990.

[4] V. Chickermane and J. H. Patel, “An Optimization Based Approach to the Partial Scan Design Problem ”, *International Test Conf.*, pp.377-386, September 1990.

[5] K. S. Kim and C. R. Kime, “Partial Scan by Use of Empirical Testability”, *International Conf. on Computer-Aided Design*, pp.314-317, November 1990.

[6] D. S. Lee and S. M. Reddy, “On Determining Scan Flip-Flops in Partial-Scan”, *International Conf. on Computer-Aided Design*, pp.322-325, November 1990.

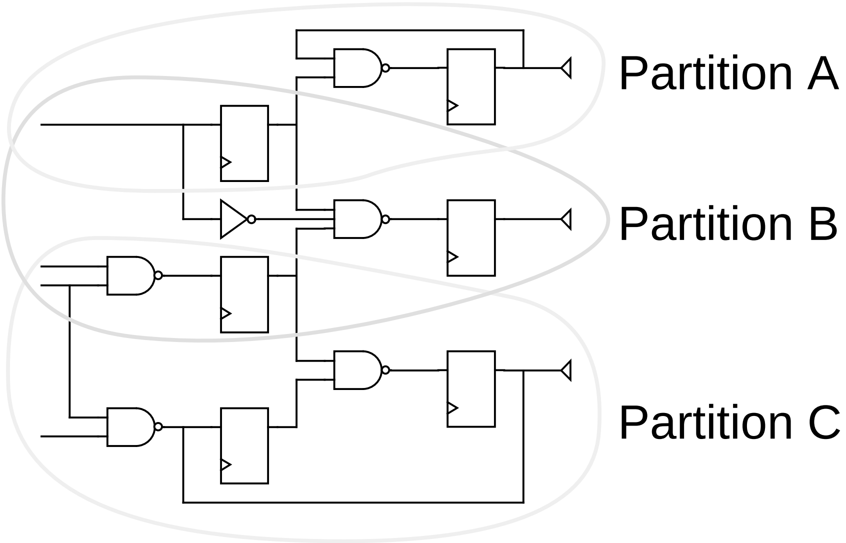
[7] S. Park and S. B. Akers, “A Graph Theoretic Approach to Partial Scan Design by K-Cycle Elimination”, *International Test Conf.*, pp.303-311, September 1992.

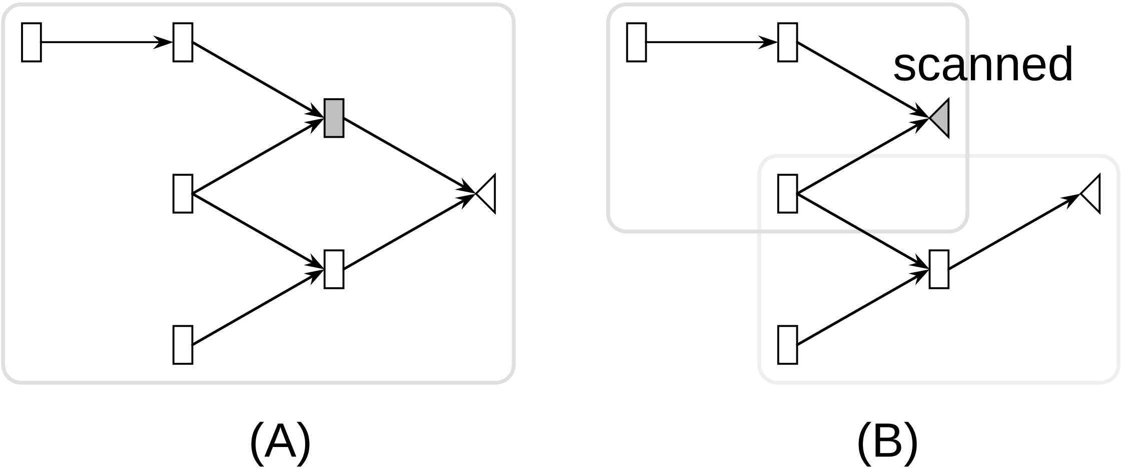
[8] P. Ashar and S. Malik, “Implicit Computation of Minimum-Cost Feedback-Vertex Sets for Partial Scan and Other Applications”, *Design Automation Conf.*, pp.77-80, June 1994.

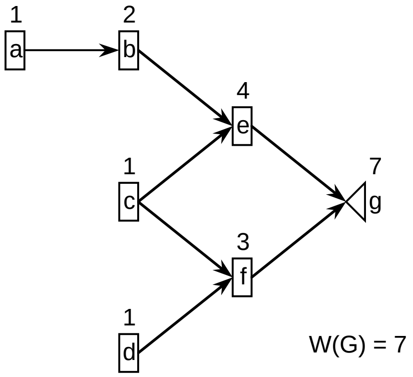
[9] S. T. Chakradhar, A. Balakrishnan and V. D. Agrawal, “An Exact Algorithm for Selection Partial Scan Flip-Flops”, *Design Automation Conf.*, pp.81-86, June 1994.

[10] F. Brglez, D. Bryan and K. Kozminski, “Combinational Profiles of Sequential Benchmark Circuits”, *International Symp. on Circuits and Systems*, pp.1929-1934, May 1989.

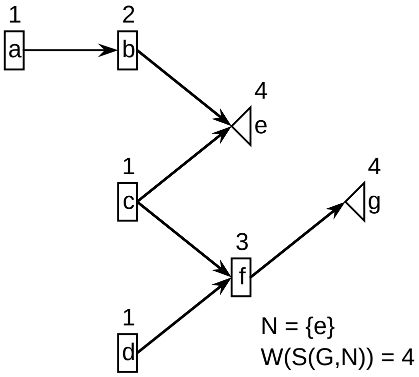
[11] T. Niermann and J. H. Patel, “HITEC: A Test Generation Package for Sequential Circuits”, *European Design Automation Conf.*, pp.214-218, March 1991.


Partition A
Partition B
Partition C

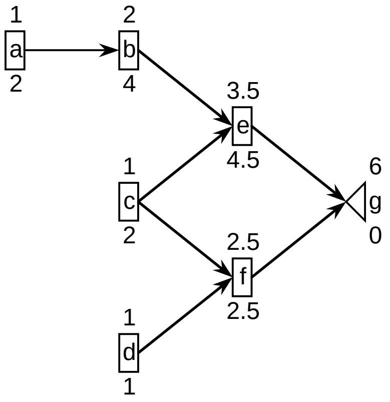

scanned
(A)
(B)



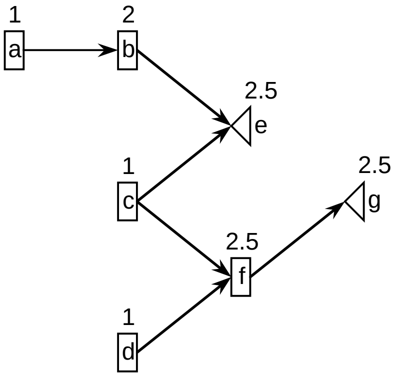
Graph G
(A)



Scanned Graph S(G,N)
(B)



(A)



(B)