On Minimization of Peak Power for Scan Circuit during Test

Jaynarayan T. Tudu*, Erik Larsson†, Virendra Singh*, and Vishwani D. Agrawal‡
* *Indian Institute of Science, Bangalore 560012, India*
Email: jayttudu@csa.iisc.ernet.in, viren@serc.iisc.ernet.in
† *Linköping University, SE-581 83 Linköping, Sweden*
Email: erila@ida.liu.se
‡ *Auburn University, Auburn, AL 36849, USA*
Email: vagrawal@eng.auburn.edu.

Abstract

Scan circuit generally causes excessive switching activity compared to normal circuit operation. The higher switching activity in turn causes higher peak power supply current which results into supply voltage droop and eventually yield loss. This paper proposes an efficient methodology for test vector re-ordering to achieve minimum peak power supported by the given test vector set. The proposed methodology also minimizes average power under the minimum peak power constraint. A methodology to further reduce the peak power, below the minimum supported peak power, by inclusion of minimum additional vectors is also discussed. The paper defines the lower bound on peak power for a given test set. The results on several benchmarks shows that it can reduce peak power by up to 27%.[1]

1. Introduction

Power consumption during testing of scan circuit is an important issue to address for today's very complex sequential circuits. It becomes especially important when chips are designed with small feature size and higher frequency; hence, at-speed test becomes necessary. Excessive average power results into burn out of chip whereas excessive peak power results into power droop problem which can falsely classify a good chip as a faulty chip. Average power can be reduced by reducing clock frequency. Reduction of peak power during test becomes very important for two reasons: 1. Higher peak power causes yield loss due to power droop and cross talk, 2. If the scan circuit is a module in an SoC then multiple module can be scheduled together to minimize test time. Our methodology shows that the test vector order under minimum achievable peak power, determined by our approach, would also smooth out the power profile. The smooth power profile can facilitate us to use box model [11] for test scheduling in an SoC. Otherwise, a more complex cycle accurate model [12], [13] must be used for SoC test scheduling [14].

The problem of test power reduction is an active area of research for quite sometime. Most of the solutions [3], [8], [11] are aimed at average power reduction. However, they have also achieved small reduction in peak power as a by-product. Test vector reordering [8] is used to achieve average power reduction. Bonhomme et al. [10] used scan chain reordering technique to minimize the total number of transitions to reduce average power. The scan chain reordering leads to extra interconnect area and power dissipation. In [4] and [9], logic is added to hold the output of the scan cells at a constant value during scan shifting thereby reducing power dissipation. This approach greatly reduces average power, and will avoid peak power problems during scan shifting but these approaches lead to area overhead. Moreover, this approach degrades circuit performance because it adds extra logic in the functional paths. Another set of solutions which have been proposed in the literature are the low activity pattern generation using ATPG. Corno et al. [5] proposed a test pattern generation technique which modifies test sequence for sequential non-scan testing for reducing peak power. The approaches proposed by Shankarlingam and Touba [6] and Wen et al. [7] assign don't care bits of the deterministic test cubes used during test in such a way that it can reduce the peak power. Badereddine et al. [2] proposed a solution based on a power-aware assignment of don't care bits in deterministic test patterns. This solution proposes that X values of generated pattern is filled before doing the test pattern minimization, thereby the algorithm may put limits on the test pattern compaction and it may increase test time. This paper proposes methodology to minimize peak power by test vector ordering. We also provide the bound on the achievable minimum peak power for a given test set.

The rest of the paper is organized as follows. Section 2 presents the problem formulation where we formulated three problems and also describe the lower bound on peak power. Section 3 describes algorithms for all three problems. Section 4 presents the experimental results. The paper concludes Section 5.

1. This research is partly supported by The Swedish Foundation for International Cooperation in Research and Higher Education (STINT) through Institutional Grant for Younger Researchers for collaboration with Indian Institute of Science (IISc), Bangalore, India.

1530-1877/09 $25.00 © 2009 IEEE
DOI 10.1109/ETS.2009.36

IEEE computer society

Authorized licensed use limited to: Linkoping Universitetsbibliotek. Downloaded on November 8, 2009 at 07:18 from IEEE Xplore. Restrictions apply.

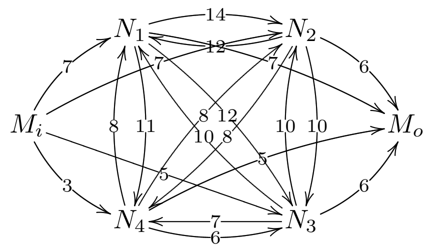

Figure 1. A weighted digraph for patterns in Example 1

2. Problem Formulation

Problem Statement: Determination of the minimum achievable peak power during test for a given test set and find out a test vector sequence which can support the minimum achievable peak power.

The proposed work formulates a graph theoretic problem by mapping test vector sequence to a complete weighted directed graph. The complete weighted directed graph D_c can be constructed in the following way: Each test pattern T_i corresponds to the node N_i of the digraph. There would be a directed edge, E_{ij}, from node N_i to node N_j if pattern T_i and T_j can be applied consecutively. The weight of the edge E_{ij}, EW_{ij}, is the maximum of number of transitions occurs in scan chain per clock cycle for complete scan operation including load/unload, and launch and capture cycles while applying test pattern T_i followed by T_j. Since any pattern can follow any other pattern, the constructed graph would be a complete graph. Example 1 helps understanding the graph construction. We use Example 1 as a running example in this paper.

Example 1: Let length of scan chain be 15 and test set size be 5. Test patterns and responses are listed below:

T_1: 111111101010111	R_1: 111010101111000
T_2: 111111111010101	R_2: 101010111111111
T_3: 111111111110101	R_3: 101011111111001
T_4: 111111111111101	R_4: 110111111110110

Figure 1 shows the weighted diagraph for this test set. We introduce a dummy node M_i to scan in the first vector assuming scan chain is initially in reset state and M_o to scan out the last response.

We have formulated three problems to obtain the minimum peak power for a given test vector set and the order of test vectors. The first problem gives test vector sequence without time penalty and the other two problems give the test vector sequences with marginal increase in time. At the end we'll define the lower bound on the minimum achievable peak power.

2.1. Formulation of Problem-1

Definition 1. ***path-weight*** *is defined as the maximum of weight of each edge in the path in a digraph.*

Problem Statement: Given a complete weighted directed graph D_c, find out a Hamiltonian path which has minimum *path-weight*.

Consider three different Hamiltonian paths, Path1, Path2 and Path3 for the digraph shown in Figure 1:

$$M_i \xrightarrow{7} N_1 \xrightarrow{12} N_3 \xrightarrow{7} N_4 \xrightarrow{8} N_2 \xrightarrow{6} M_o$$
$$M_i \xrightarrow{7} N_2 \xrightarrow{8} N_4 \xrightarrow{6} N_3 \xrightarrow{10} N_1 \xrightarrow{7} M_o$$
$$M_i \xrightarrow{5} N_3 \xrightarrow{10} N_1 \xrightarrow{14} N_2 \xrightarrow{8} N_4 \xrightarrow{5} M_o$$

Let PW_1, PW_2 and PW_3 be the corresponding *path-weights*. Then, we have $PW_1 = \max\{(M_i, N_1), (N_1, N_3), (N_3, N_4), (N_4, N_2), (N_2, M_o)\} = 12$. Similarly, $PW_2 = 10$ and $PW_3 = 14$. Thus,

Minimum Peak Power = $\min(PW_1, PW_2, PW_3) = 10$

To obtain a solution, we have transformed the problem to an unweighted directed graph problem. The transformation of a weighted digraph for a given threshold peak power value, P_{th} to an unweighted digraph is as follows:

- Every node in weighted graph has a corresponding node in unweighted graph.
- Remove all the edges whose weight is greater than P_{th}.
- Replace all other edges $EW_{ij} \leq P_{th}$, with unweighted edges.

The problem can be restated as: For a given complete weighted directed graph D_c, find out the minimum peak power P^1_{th}, such that its corresponding unweighted graph, D_u has at least one Hamiltonian path.

The above problem formulation obtains minimum possible peak power P^1_{th}. There may be more than one possible Hamiltonian paths in the unweighted graph. Although minimization of peak power minimizes average power, we can further optimize for given peak power P^1_{th} by choosing a path that gives minimum average power. Note that we will visit every node only once and will not increase the test time.

2.2. Formulation of Problem-2

A directed graph, D, which does not contain any Hamiltonian path may have a possible walk [1] that visits each and every node at least once. This is illustrated by the following scenario shown in Figure 2.

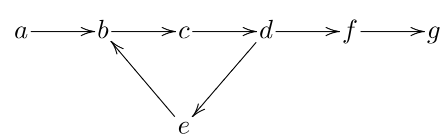

Figure 2. An unweighted digraph

It can be observed from Figure 2 that Hamiltonian path does not exist in the graph but there is a walk from node

Authorized licensed use limited to: Linkoping Universitetsbibliotek. Downloaded on November 8, 2009 at 07:18 from IEEE Xplore. Restrictions apply.

a to node g which revisit node e, b, c and d and visit each node at least once.

Definition 2. *We define* ***walk-weight*** *as the maximum edge-weight in the walk in a digraph.*

Problem Statement: For a given complete weighted digraph D_c, find a walk which visits each and every nodes at least once with minimum *walk-weight*.

The problem can be redefined for unweighted graph as follows: For a given weighted directed graph D, find out the minimum peak power value, P_{th}^2, such that its corresponding unweighted graph has at least one walk.

The possible walk will revisit a few nodes; hence, it will increase the test time. The increase in test time is proportional to the number of revisited nodes.

2.3. Formulation of Problem-3

Problem-1 obtains Hamiltonian path having minimum peak power P_{th}^1 and Problem-2 finds a walk having minimum peak power P_{th}^2(less than P_{th}^1) at the cost of revisiting some nodes. If we further reduce P_{th}(minimum achievable peak power), the corresponding unweighted graph, D_u, will not have a walk to apply complete test set. Further reduction would result into disjoint paths. These paths can be joined to form a single walk by introduction of some extra low activity nodes (currently we consider only all 0's patterns and all 1's patterns whose corresponding nodes are N_{a0} and N_{a1}) to D_u. We refer, this walk with these additional nodes as *extended-walk* and the graph D_u with these additional nodes as *extended graph*. Note that we only scan in these patterns and will not apply so that scan out would have same value. The above stated scenario is illustrated in Figure 3.

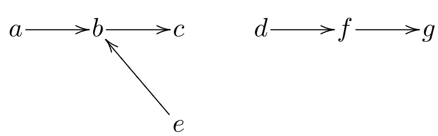

Figure 3. Disconnected unweighted digraph

In Figure 3, the possible disjoint paths could be $a \longrightarrow b \longrightarrow c$, $d \longrightarrow f \longrightarrow g$, and e. After interconnecting these paths the resultant will be as follows: $a \longrightarrow b \longrightarrow c \longrightarrow x \longrightarrow d \longrightarrow f \longrightarrow g \longrightarrow y \longrightarrow e$, where x and y are new nodes. Therefore, the problem can be defined in following way.

Problem statement: For a given complete weighted directed graph D_c, find the minimum peak power value, P_{th}^3 such that its corresponding unweighted *extended graph* has at least one *extended-walk*.

2.4. Lower Bound on Minimum Peak Power

Problem-3 can achieve the minimum possible peak power by inserting additional nodes N_{a0} (all 0's) and N_{a1} (all 1's).

Theorem 1. *The lower bound on minimum achievable peak power for a test set is*

$$\max_{\forall_i}[\max\{\min(EW_{a0i},\ EW_{a1i}), \min(EW_{ia0},\ EW_{ia1})\}] \tag{1}$$

Proof: In order to achieve the minimum peak power the graph should be augmented by additional nodes N_{a0} and N_{a1} to construct an extended graph. In extended graph, $EW_{ij} \geq \max\{\min(EW_{ia0},\ EW_{ia1}), \min(EW_{a0j},\ EW_{a1j})\}$. Therefore, it is guaranteed that visiting node j directly from node i would be expensive or equal to visiting through either node N_{a0} or N_{a1}. In worst case, we may always need to traverse through extended nodes while going from node i to node j. The minimum power to traverse node j from node i would be $\max\{\min(EW_{ia0},\ EW_{ia1}), \min(EW_{a0j},\ EW_{a1j})\}$. Hence, the minimum achievable peak power for a given test set would be the maximum of minimum peak power computed for all nodes, that is $\max_{\forall_i}[\max\{\min(EW_{a0i},\ EW_{a1i}), \min(EW_{ia0},\ EW_{ia1})\}]$.

Lemma 1. *Let there be a test set of n vectors and scan chain length be l. The worst case test time to achieve the lower bound is $(3n-1) \times l + n - 1$.*

Proof: According to Theorem 1, we can always achieve minimum power by traversing through N_{a0} or N_{a1}. Hence, it adds one extra node in the traversal from node i to node j. However, it can be observed that to achieve minimum power, node j can be reached from node N_{a1} or N_{a0}, and N_{a0} or N_{a1} can be reached from node i. Therefore, in the worst case we need to take the following path to reach node j from node i: N_i to N_{a0} to N_{a1} to N_j. We walk through additional $(2n-1)$ nodes in the worst case. Therefore, the worst case time would be: $(3n-1) \times l + n - 1$.

Corrollary 1. *The minimum achievable peak power under maximum test time $(2n-1) \times l + (n-1)$ for a test set is* $\max_{\forall_i}\{\max(EW_{a0i}, EW_{a1i}, EW_{ia0}, EW_{ia1})\}$.

Proof: A walk from node i to node j under power constraint $\max(EW_{a0i}, EW_{a1i}, EW_{ia0}, EW_{ia1})$ can be made by the following traversal: V_i to V_{a0} to V_j, or V_i to V_{a1} to V_j. Hence, the worst case test time under above stated power constraint would be $(2n-1) \times l + n - 1$.

Note that the costs of walks N_{a0} to N_{a1} to N_j, and N_i to N_{a0} to N_{a1} to N_j differ by at the most one transition as $\max\{\min(EW_{a0i}, EW_{a1i}), \min(EW_{ia0}, EW_{ia1}\} - \max(EW_{a0i}, EW_{a1i}, EW_{ia0}, EW_{ia1}) = 1$. Therefore, we can infer from Corollary 1 that we can reduce the test time significantly at the cost of one additional transition in the worst case.

Authorized licensed use limited to: Linkoping Universitetsbibliotek. Downloaded on November 8, 2009 at 07:18 from IEEE Xplore. Restrictions apply.

3. Algorithms

3.1. Algorithm-1 for Problem-1

The following algorithm computes the Hamiltonian path and minimum peak power value P^1_{th}.

```
AlgoMinHam(Input:D_c,Output:HamPath,P^1_th)
Let A be nondecreasing sorted array of edge-weight;
index = SelectInitialIndex;
while(TRUE)
  UpLimit = A(index);
  RemoveEdge(D_c, UpLimit, D_u);
  found = SearchHam(D_u , HamPath) ;
  if( found == TRUE )
  then
      P^1_min = UpLimit;
      return HamPath and P^1_th;
  else
      index = SelectNextIndex;
      CONTINUE ;
EndofAlgo
```

The procedure *RemoveEdge* removes all the edges with edge-weight greater than *UpLimit*. *SearchHam* finds the Hamiltonian path if one exists. The following example illustrates this.

Example 2 : Let D_c be the digraph from Figure 1. The Algorithm-1 will return the following Hamiltonian path:

$M_i \xrightarrow{3} N_4 \xrightarrow{8} N_2 \xrightarrow{10} N_3 \xrightarrow{10} N_1 \xrightarrow{7} M_o$

and minimum peak power value P^1_{th} = 10.

3.2. Algorithm-2 for Problem-2

The following algorithm finds out a minimum-weight *walk* if exist which visits every nodes of given digraph at least once.

```
AlgoMinWalk(Input:D_c, Output:Walk, P^2_min)
Let A be nondecreasing sorted array of edge-weight;
index = SelectInitialIndex;
while(TRUE)
   UpLimit = A(index);
   RemoveEdge(D_c, UpLimit, D_u);
   Success = AlgoConDag(D_u,DAG);
   if(Success == TRUE)
       AlgoConIpath(DAG, Ipath);
       AlgoConWalk(Ipath, Walk);
       P^2_min = UpLimit;
       return Walk and P^2_th;
   else
       index = SelectNextIndex ;
       CONTINUE ;
EndofAlgo.
```

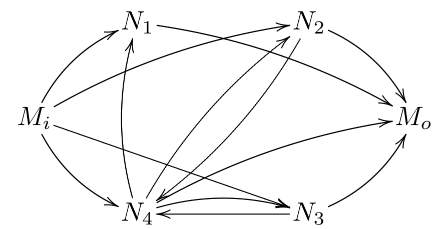

Figure 4. Unweighted digraph with edge-weights $\leq$ 8

The procedure *AlgoConDag* constructs a directed acyclic graph (DAG) from given digraph D_u by forming each cycle into corresponding *supernode* [15] if it can be constructed. *AlgoConIpath* constructs an intermediate path *Ipath* from a given DAG by using the concept of acyclic ordering [1]. Finally, *AlgoConWalk* procedure constructs a walk from a given *Ipath* by unrolling *supernodes* into cycles and connecting the nodes. The complexity of the algorithm would be O($nodes^3$ + $edges^2$), which is polynomial. Following example illustrates this.

Example 3: Let D_c be complete digraph from Figure 1. Then unweighted digraph D_u after RemoveEdge() is shown in Figure 4. After obtaining this digraph, resultant $Walk$ and P^2_{th} are obtained. The $Walk$ is:

$M_i \xrightarrow{7} N_2 \xrightarrow{8} N_4 \xrightarrow{6} N_3 \xrightarrow{7} N_4 \xrightarrow{8} N_1 \xrightarrow{7} M_o$

and P^2_{th} = 8 which is $< P^1_{th}$.

3.3. Algorithm-3 for Problem-3

This algorithm find a minimum-weight, P^3_{th},*walk* in an extended graph.

```
AlgoMinEWalk(Input:D_c Output:EWalk, P^3_th)
Let A be nondecreasing sorted array of edge weight;
index=SelectInitialIndex;
while(TRUE)
     UpLimit = A(index);
     RemoveEdge(D_c, UpLimit, D_u);
     FindDisjointPath(D_u, C);
     JoinPath(C, EWalk);
     TempWalk = EWalk;
     P^3_th =UpLimit;
     if(LowerBound < P^3_th)
            index=SelectNextIndex;
            CONTINUE;
     else
          return TempWalk and P^3_th;
EndofAlgo
```

Procedure *FindDisjointPath* finds the directed path cover. Then procedure *JoinPath* introduces new nodes to interconnect paths. Here we are using all 0's N_{a0} or all 1's N_{a1} nodes to connect. The use node N_{a0} or N_{a1} depends upon the

Authorized licensed use limited to: Linkoping Universitetsbibliotek. Downloaded on November 8, 2009 at 07:18 from IEEE Xplore. Restrictions apply.

Table 1. Results for Algorithm-1

Benchmark	#testpattern	scanlength	Peaktrans GivenOrder	Peaktrans Algo-1	Improvement%	Avgtrans GivenOrder	Avgtrans Algo-1	Improvement%
s9234	105	211	68	56	17.64	27.02	26.37	2.40
s38584	110	1426	444	379	14.63	120.99	119.45	1.24
s15850	95	534	120	96	20.00	37.09	36.42	1.80
s35932	12	1728	106	89	16.03	34.61	33.38	3.55
s38417	68	1636	473	342	27.69	161.35	158.84	1.55
s1196	136	18	17	14	17.64	9.21	8.80	4.45
b04	43	66	49	44	10.20	29.17	28.58	2.03
b07	52	45	34	31	08.82	19.37	18.95	2.17
b10	47	17	16	13	18.75	8.26	7.85	4.96
b14	627	215	163	150	7.97	98.10	98.00	0.11
b19	628	215	171	162	5.26	98.12	97.96	0.17
b21_1	488	215	144	129	10.41	99.34	99.20	0.15

response of the last vector or the stimuli of the first vector. Note that the complexity of the algorithm is polynomial. The following example illustrates the function of Algorithm-3.

Example 4: Let D_c be the digraph from Figure 1. The unweighted digraph D_u after RemoveEdge() is shown in Figure 5.

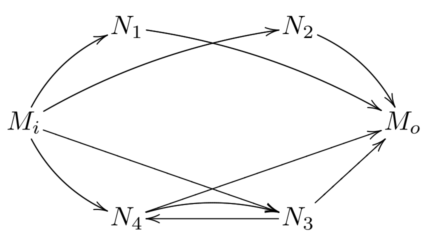

Figure 5. Unweighted digraph having edge weight ≤ 7

In this example the possible disjoint path would be $M_i \xrightarrow{3} N_4 \xrightarrow{6} N_3$, N_1 and $N_2 \xrightarrow{6} M_o$. Then resultant *extended-walk* would be,

$$M_i \xrightarrow{3} N_4 \xrightarrow{6} N_3 \xrightarrow{6} N_{a1} \xrightarrow{6} N_1 \xrightarrow{7} N_{a1} \xrightarrow{6} N_2 \xrightarrow{6} M_o$$

and P_{th}^3 = 7 which is $< P_{th}^2$.

4. Implementation and Experimental Results

All the three Algorithms are implemented in C. For experimental evaluation we used ITC99 and ISCAS89 benchmark circuits. Scan insertion and synthesis are carried out using "DesignCompiler" and test patterns are generated using "TetraMAX". The results are shown in Tables I, II and III, respectively, for Algo-1, 2 and 3. For each benchmark circuit, the peak transition due to given order of test patterns (given by TetraMax, peak transition using proposed algorithms, and percentage gain achieved are shown in tables. In Table I, the average transition due to given order and due to Algo-1 are shown with percentage of improvement. For Algo-1 test time remains unchanged. For Algo-2 and Algo-3 test time is reported in Tables II and III in columns named "testtime overhead".

4.1. Analysis of Experimental Results

The experimental results shown in Table I indicates that Algo-1 can achieve up to 27% reduction in peak power and average reduction in peak power is about 16% in comparison to given order. It is important to note that it also reduces average power up to 4.45%. Also due to the NP-completeness, Algo-1 was aborted for s838, and b22, and Algo-2 is used for these as it is faster algorithm - with polynomial time complexity.

Algo-2 achieved about 8.33% improvement over Algo-1 while incurring test-time overhead of 1.33%. Algo-2 was not able to reduce the peak power of other benchmarks below the limit achieved by Algo-1 because either Algo-1 itself achieved the lower bound or the benchmark did not have any cycle that can make walk different than Hamiltonian path. *However, for the benchmarks which achieved lower bound in Algo-1, Algo-2 was able to achieve the same result without increase in test time* (without revisiting a node) quickly in comparison to Algo-1. For rest of the benchmarks (which are not mentioned in Table II) reported in Table I, Algo-2 also achieved the same results (in terms of peak power reduction and average power reduction without test time overhead) as obtained by Algo-1 and reported in Table I.

For benchmark b19 and b22 Algo-3 has marginal improvement over Algo-1 and Algo-2. It could not further reduce the peak power for the other bench marks which are not reported in Table III, because they already achieved the lower bound by Algo-1 or Algo-2. It is important to note that Algo-2 and Algo-3 are polynomial time complexity algorithms. Therefore, these algorithms can be used for large industrial circuits.

5. Conclusion

We have discussed graph theoretic approaches to minimize peak power during the test, which is a concern for a complex scan based design. A presented approach can

Authorized licensed use limited to: Linkoping Universitetsbibliotek. Downloaded on November 8, 2009 at 07:18 from IEEE Xplore. Restrictions apply.

Table 2. Results for Algorithm-2

Bench mark	#test pattern	scan length	Peaktran GivenOrder	Peaktran Algo-2	Improve ment%	improve% overAlgo-1	testtime overhead	Avgtran GivenOrd	Avgtran Algo-2	Improve ment%
s838	75	32	16	12	25.00	0.00	0.00	4.61	4.59	0.43
s838	75	32	16	11	31.25	08.33[2]	1.33	4.61	4.57	0.80
b19	628	215	171	162	5.26	0.00	0.00	98.12	97.96	0.17
b22	622	215	172	158	6.97	0.00	0.00	98.23	98.07	0.16

Table 3. Results for Algorithm-3

Bench mark	#test pattern	scan length	Peaktran GivenOrder	Peaktran Algo-3	Improve ment%	improve% overAlgo-2	testtime overhead	Avgtran GivenOrd	Avgtran Algo-3	Improve ment%
b19	628	215	171	161	5.84	0.61	0.63	98.12	97.50	0.63
b22	622	215	172	157	8.72	0.63	0.64	98.23	97.61	0.63

achieve minimum peak power without increase in test time. We also present two approaches that can further reduce the peak power at a cost of increase in test time. Finally, we have presented a lower bound on the minimum achievable peak power for a given test set. The results indicate that we can achieve up to 27% peak power reduction without increase in test time. We can also reduce average power by up to 4.96%. The proposed algorithms Algo2 and Algo-3 are polynomial complexity algorithms, hence the methodology is scalable. In the future, Algo-1 can be implemented as an approximation algorithm to reduce the time complexity. Note that the proposed method can be applied with the existing X-fill based approaches because it uses fully specified vectors. Hence, we can achieve more power saving in combination with the existing techniques. It could also be more effective if integrated with ATPG.

6. Acknowledgment

The authors thank to Prof. Michiko Inoue, NAIST, Japan and Prof. Tim Cheng, UCSB, USA for their valuable suggestions. The authors also thanks to Prof. Sunil Chandran, IISc, Bangalore for helping in understanding graph theoretic concepts, and Prof. Seiji Kajihara, KIT, Japan for providing test pattern file for d695.

References

[1] J. Bang-Jensen and G. Gutin, *Digraphs Theory, Algorithms and Applications*, Springer-Verlag, 2006.

[2] N. Badereddine, P. Girard, S. Pravossoudovitch, C. Landrault, A. Virazel, and H.-J. Wunderlich, "Minimizing Peak Power Consumption during Scan Testing: Test Pattern Modification with X Filling Heuristics," *Proc. Int'l Conf. Design and Test of Integrated System in Nanoscale Tech.*, pp. 359–364, 2006.

[3] R. Sankaralingam, R. Oruganti, and N. A Touba, "Static Compaction Technique to Control Scan Vector Power Dissipation," *Proc. of VLSI Test Symposium,* pp. 35-40, 2000.

[4] A. Hertwig and H.-J. Wunderlich, "Low Power Serial Built-In Self-Test," *Proc. European Test Workshop,* pp. 49–53, 1998.

2. when Algo-2 is used as approximation of Algo-1 - as in previous row

[5] F. Corno, M. Rebaudengo, M. Sonza Reorda, and M. Violante, "On Reducing the Peak Power Consumption of Test Sequences," *Proc. European Conf. Circuit Theory and Design,* pp. 247–250, 1999.

[6] R. Sankaralingam and N. A. Touba, "Controlling Peak Power during Scan Testing," *Proc. IEEE VLSI Test Symp.,* pp. 153–159, 2002.

[7] X. Wen, Y. Yamashita, S. Kajihara, L.-T. Wang, K. K. Saluja, and K. Kinoshita, "On Low-Capture-Power Test Generation for Scan Testing," *Proc. IEEE VLSI Test Symposium,* pp. 265–270, 2005.

[8] V. Dabholkar, S. Chakravarty, I. Pomeranz, and S. M. Reddy, "Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application," *IEEE Trans. Comp.-Aided Design,* Vol. 17, No. 12, pp. 1325–1333, Dec. 1998.

[9] S. Gerstendörfer, and H.-J. Wunderlich, "Minimized Power Consumption for Scan-Based BIST," *Proc. Intl. Test Conf.,* pp. 77-84, 1999.

[10] Y. Bonhomme, P. Girard, C. Landrault, and S. Pravossooudovitch, "Power Driven Chaining of Flip-Flops in Scan Architectures," *Proc. Intl. Test Conf.,* pp. 796–802, 2002.

[11] R. M. Chou, K. K. Saluja, and V. D. Agrawal, "Scheduling Test for VLSI Systems under Power Under Power Constraints," *IEEE Trans. VLSI Systems,* Vol. 5, No. 2, pp. 175–185, June 1997.

[12] S. Samii, M. Selkälä, E. Larsson, K. Chakrabarty, and Z. Peng "Cycle-Accurate Test Power Modeling and Its Application to SoC Test Architecture Design and Scheduling," *IEEE Trans. Comp.-Aided Design,* Vol. 27, No. 5, pp. 973–977, May 2008.

[13] S. Samii, E. Larsson, K. Chakrabarty, and Z. Peng "Cycle-Accurate Test Power Modeling and its Application to SoC Test Scheduling," *Proc. Intl. Test Conf.* pp. 1-10, 2006.

[14] J. Pouget, E. Larsson, and Z. Peng "Multiple-constraint driven system-on-chip test time optimization," *Journal Electronic Testing: Theory and Application,* Vol. 21, No. 6, pp. 599–611, 2005.

[15] H. Gabow, Z. Galil and T. Spencer, "Efficient Implementation of Graph Algorithms Using Contraction," *Proc. 25th Annual Symp. Foundation of Computer Science,* pp. 347-357, 1984.

Authorized licensed use limited to: Linkoping Universitetsbiblioteket. Downloaded on November 8, 2009 at 07:18 from IEEE Xplore. Restrictions apply.