DP-fill: A Dynamic Programming approach to X-filling for minimizing peak test power in scan tests

***Abstract*—At-speed testing is crucial to catch small delay defects that occur during the manufacture of high performance digital chips. Launch-Off-Capture (LOC) and Launch-Off-Shift (LOS) are two prevalently used schemes for this purpose. LOS scheme achieves higher fault coverage while consuming lesser test time over LOC scheme, but dissipates higher power during the capture phase of the at-speed test. Excessive IR-drop during capture phase on the power grid causes false delay failures leading to significant yield reduction that is unwarranted. As reported in literature, an intelligent filling of don't care bits (X-filling) in test cubes has yielded significant power reduction. Given that the tests output by automatic test pattern generation (ATPG) tools for big circuits have large number of don't care bits, the X-filling technique is very effective for them. Assuming that the design for testability (DFT) scheme preserves the state of the combinational logic between capture phases of successive patterns, this paper maps the problem of optimal X-filling for peak power minimization during LOS scheme to a variant of *interval coloring problem* and proposes a *dynamic programming (DP)* algorithm for the same along with a theoretical proof for its optimality. To the best of our knowledge, *this is the first ever reported X-filling algorithm that is optimal.* The proposed algorithm when experimented on ITC99 benchmarks produced peak power savings of up to 34% over the best known low power X-filling algorithm for LOS testing. Interestingly, it is observed that the power savings increase with the size of the circuit.**

Keywords: Digital Systems Testing, Peak Test Power, X-filling, Dynamic Programming

I. Introduction

The exponential scaling of metal-oxide-semiconductor (MOS) transistor feature size with successive technology generations has led to an exponential increase in on-chip power densities, causing thermal hot-spots. The power dissipation during the test mode of a chip is often several times higher than the power dissipation during the normal functioning of the chip [1], [2]. Excessive average power leads to thermal stress and excessive peak power causes unacceptable dynamic IR-drop leading to false delay failures, which are important issues motivating low power test.

Excessive IR-drop specific to test mode can lead to delay failures that are not observed during the normal functioning of the chip [3], [4], thereby leading to discarding good chips as faulty, ultimately reducing their final yield. This paper focuses on reduction in peak test power through minimization of peak switching activity. The test cubes for large circuits are typically dominated by don't care (X) bits as shown in column 4 of Table I, making X-filling an effective technique for minimizing peak test power. Motivated by this, this paper proposes an analytical solution for solving the problem of optimal X-filling for minimizing peak test power. The main contributions of this paper are as follows:

- Given a test cube ordering, mapping the problem of optimal X-filling for minimizing peak test power to a variant of *interval coloring problem* which we refer to as *bottleneck coloring problem*; and,
- Proposing a polynomial time algorithm for the *bottleneck coloring problem*.

The formal definition for the problem of X-filling for test peak power minimization is shown in section IV. Our mapping of the optimal X-filling problem to the *bottleneck coloring problem* is explained in section V. Following this, the proposed algorithm for optimal X-filling along with its proof of correctness is shown in section V. The experimental setup used in this paper to implement the proposed algorithm, the results thus obtained by applying it, and comparing the same with the best known techniques proposed in the literature are shown in section VII. Section VIII concludes the paper.

TABLE I
X % : Average % of X-bits in test cubes. PIs and FFs stand for primary inputs and flip-flops respectively.

Benchmark	$\#(PIs + FFs)$	# Gates	X %
b01	5	57	7.1
b02	4	31	5
b03	29	103	70.4
b04	77	615	64.4
b05	35	608	36.8
b06	5	60	12.5
b07	50	431	58.6
b08	30	196	60.4
b10	28	217	58.7
b11	38	574	64.1
b12	126	1.6K	76.9
b13	53	596	65.4
b14	275	5.4K	77.9
b15	485	8.7K	87.8
b17	1452	27.99K	89.9
b18	3357	75.8K	86.9
b19	6666	146.5K	89.8
b20	522	9.4K	75.3
b21	522	9.4K	73.2
b22	767	13.4K	74.1

II. Related Work

There have been several techniques proposed in the past for minimizing peak test power. These techniques can be broadly categorized into circuit level [1], [5]–[7], gate level [9]–[12] and system level [13], [15]–[17], [20] techniques. Circuit level techniques include supply gating [6], scan flip-flop redesign [1], [5] and supply voltage scaling [7], [8]. Gate level techniques include clock gating [10], [15], scan cell output gating [12], and low power scan chain synthesis [1], [5], [8]–[10]. System level techniques include low power test pattern generation [16], power aware test scheduling [17], test pattern ordering [13], [14], [20] and X-filling [19], [21], [22]. All of these X-filling techniques for Launch-On-Shift (LOS) scheme [19], [21], [22] are heuristics without a performance guarantee. This paper proposes a theoretical framework to arrive at an optimal X-filling for minimizing peak test power. The following sections motivate this theoretical framework, the underlying design for testability (DFT) scheme necessary to apply the proposed technique, its proof of optimality and the results obtained after applying the same on benchmarks. To the best of our knowledge, *this is the first ever reported X-filling algorithm that is optimal.*

III. MOTIVATION

In scan based test, the input test pattern is serially shifted in, while serially shifting out the response for previous test pattern. An important assumption that is made about the role of the DFT architecture is that it preserves the state in which the combinational logic settles down after launching the current test pattern and before capturing the response, until the launching of next test pattern. The scan architecture can be made to satisfy this property with minimal area and physical design overhead [18]. If the combinational state preservation property can be ensured, the combinational part of the circuit behaves in such a way as if the test patterns are applied one after the other. Once this property is satisfied, since the combinational logic is undisturbed during scan-shift and capture phases, *as far as application of test patterns is concerned, the sequential circuit behaves like a combinational circuit. Thus, test pattern ordering technique that was proposed earlier for reducing test power in combinational circuits [13], [20] becomes equally effective for sequential circuits.* Having understood this, the next step is to compute a test pattern ordering that achieves the same.

Once the test pattern ordering is computed, the next step is to minimize the peak toggles at the inputs (primary inputs and the scan cell outputs) through filling of the X-bits in test patterns (cubes) with binary values. The expectation is that reducing the input toggles leads to lower power dissipation inside the circuit, as shown previously in [20].

The most recent and effective X-filling algorithm for peak power minimization during LOS scheme is X-Stat [22]. The X-Stat algorithm follows a two phase approach. In the first phase, it uses adjacent X-fill technique to convert don't care (X-bit) stretches 0XX...X1 and 1XX...X0 into smaller X-bit stretches 0X1 and 1X0 respectively as shown in *Phase 1* column of Fig 1. In the second phase, it replaces X-bits by either 0 or 1 in order to minimize peak toggles as shown in *Phase 2* column of Fig 1. This figure shows that the global minimum peak toggles is 2 (shown under the *Optimum-Fill* column), while the minimum peak toggles achieved by XStat technique is 3, making it sub-optimal. Because of greedy approach used in *Phase 1* of XStat technique, it does not achieve the global optimal-fill for peak toggle reduction. Motivated by this, we choose a *Dynamic Programming* paradigm which takes global picture into consideration and optimally fill the X-bits with binary values to achieve the best reduction in peak toggles.

Test Vector Sequence					X-Stat										Optimum-Fill				
					Phase 1					Phase 2									
V1	V2	V3	V4	V5	V1	V2	V3	V4	V5	V1	V2	V3	V4	V5	V1	V2	V3	V4	V5
0	X	X	1	X	0	0	X	1	1	0	0	0	1	1	0	1	1	1	1
1	X	X	X	0	1	1	1	X	0	1	1	1	0	0	1	1	1	1	0
0	X	X	1	X	0	0	X	1	1	0	0	1	1	1	0	0	1	1	1
1	X	X	X	0	1	1	1	X	0	1	1	1	0	0	1	1	1	1	0
1	X	1	X	0	1	1	1	X	0	1	1	1	1	0	1	1	1	0	0
X	0	X	X	1	0	0	0	X	1	0	0	0	0	1	0	0	1	1	1
1	0	X	X	1	1	0	0	X	1	1	0	0	0	1	1	0	0	1	1
										1	1	3	3		2	2	2	2	

Fig. 1. X-Stat [22] vs Optimum-Fill

IV. PEAK TOGGLE MINIMIZATION PROBLEM

Objective: Given a sequence of test cubes $T_1, T_2, ...T_n$ each of length m, replace each don't care in test cubes by either 0 or 1 such that $max\{hd(T_1, T_2), hd(T_2, T_3), ...hd(T_{n-1}, T_n)\}$ is $minimized$, where $hd(T_i, T_{i+1})$ is the Hamming distance between test cubes T_i, T_{i+1} after replacing don't cares by either 0 or 1.

This problem can be formulated as a variant of interval coloring problem, which we call *Bottleneck Coloring Problem*. Next, we explain and define *Bottleneck Coloring Problem* and how peak power minimization is an instance of this problem. Since our objective is to minimize the peak toggles we are naming this problem as *Bottleneck Coloring Problem*.

V. BOTTLENECK COLORING PROBLEM (BCP)

A. Problem Explanation in Terms of Hotel Room Booking

Suppose a hotel received several guest requests for accommodation each of which has a *start-date* and *end-date* of a time period, and asking the hotel to provide accommodation for exactly one day which falls in the given period. The aim of the hotel is to assign rooms to all guest requests such that number of guests staying in the hotel on any given day is minimized, which is a variant of the *interval coloring problem* [23].

B. Mathematical Definition of Problem

- Let *S* = $(s_1, e_1), (s_2, e_2) \ldots (s_k, e_k)$ be a sequence of intervals such that s_i and e_i are integers corresponding to starting and ending times of interval i respectively, for all $1 \leq i \leq k$.
- Let $max\_color = max(e_1,\ e_2,\ e_3, \ldots e_k)$.
- Let $\{\ c_1,\ c_2,\ c_3 \ldots c_{max\_color}\ \}$ be a set of colors.
- For each interval (s_i, e_i) assign a color c_j such that $s_i \leq j \leq e_i$.
- Let $h_1,\ h_2,\ h_3 \ldots h_{max\_color}$ be a sequence of integers such that h_j be the number of intervals which are assigned color c_j.
- Our objective is to assign colors to intervals such that $max(h_1,\ h_2 \ldots h_{max\_color})$ is minimized.

Here, Each interval corresponds to a accommodation request in the subsection V-A. Each color corresponds to a day. Assigning color c_j to the interval $(s_i,\ e_i)$ is same as allocation of hotel room on j^{th} day to this request. Note that h_j denotes the number of guests who are assigned room on j^{th} day.

C. Mapping of Input Peak Toggle Minimization Problem to the BCP

- Let $T_1, T_2, ...T_n$ be a sequence of test cubes each of length m
- Construct a $m \times n$ matrix A such that i^{th} column of A is equal to the test cube T_i.

- **for** $i = 1 \rightarrow m$ **do**

```
/* Preprocessing of 0XX..X0,1XX..X1
   stretches                          */
```

If $\{\ i^{th}$ row contain a sub-sequence 0XX...X0$\}$ then replace every don't care in this sub-sequence by zero since there exists an optimal solution in which all of these don't cares are replaced by zeros irrespective of how other don't cares are replaced.

If $\{\ i^{th}$ row contain a sub-sequence 1XX...X1$\}$ then replace every don't care in this sub-sequence by one since there exists an optimal solution in which all of these don't cares are replaced by ones irrespective of how other don't cares are replaced.

end

- Let S=ϕ

- **for** $i = 1 \rightarrow m$ **do**

/* Creating intervals for 0XX..X1,1XX..X0 stretches */

If there exist $k < l$ such that $A_{i,k}$ = **0**, $A_{i,l}$ = **1** and $A_{i,k+1}...A_{i,l-1}$ are don't cares then append an interval $(k, l-1)$ to sequence of intervals S.

Comment 1 : Note that there exists an optimal solution to *Peak Toggle Minimization Problem* such that $A_{i,k}$ = **0**, $A_{i,k+1}$ = **0**,..., $A_{i,j}$ = **0**, $A_{i,j+1}$ = **1**, $A_{i,j+2}$ = **1**,..., $A_{i,l}$ = **1**, where $k \leq j < l$, irrespective of how other don't cares are replaced. There is only one toggle between j^{th} and $j + 1^{th}$ test vectors in this sub-sequence. The color assigned to this newly added interval in the solution of *BCP* captures the location of this toggle in this sub-sequence.

If there exist $k < l$ such that $A_{i,k}$ = **1**, $A_{i,l}$ = **0** and $A_{i,k+1}...A_{i,l-1}$ are don't cares then append an interval $(k, l-1)$ to sequence of intervals S.

Comment 2 : Note that there exists an optimal solution to *Peak Toggle Minimization Problem* such that $A_{i,k}$ = **1**, $A_{i,k+1}$ = **1**,..., $A_{i,j}$ = **1**, $A_{i,j+1}$ = **0**, $A_{i,j+2}$ = **0**,..., $A_{i,l}$ = **0**, where $k \leq j < l$, irrespective of how other don't cares are replaced. There is only one toggle between j^{th} and $j + 1^{th}$ test vectors in this sub-sequence. The color assigned to this newly added interval in the solution of *BCP* captures the location of this toggle in this sub-sequence.

end

Each row of the matrix represents an input pin to the circuit (corresponds to a guest in BCP) and each column represents a test cube (corresponds to a *day* in the BCP formulation as per section V-A). A toggle in i^{th} row, from j^{th} position to $(j+1)^{th}$ position corresponds to a hotel room allocation for i^{th} customer on j^{th} day. The BCP ensures that the number of allocations on any given day is minimized, which in the current context translates to minimization of number of peak toggles on any given test cycle (launch-capture duration).

D. Constructing Optimal solution for Peak Toggle Minimization Problem from Optimal solution for Bottleneck Coloring Problem

- Suppose color c_j is assigned to interval (s_i, e_i) in the given optimal solution for *Bottleneck Coloring Problem* .
- Look at the row in matrix A correspond to interval (s_i, e_i), make all bits from column s_i to column j same as bit value at column s_i and make all bits from column $j+1$ to column e_i+1 same as bit value at column $e_i + 1$

VI. Algorithm

A. Dynamic Programming Approach to compute Lower-Bound (LB) for Bottleneck Coloring Problem

Algorithm 1 gives the lower bound on the number of intervals which are assigned the same color. This algorithm can be implemented such that running time is $O(k^2)$, where k is the number of intervals.

B. Greedy Approach to Bottleneck Coloring Problem

Algorithm 2 assigns colors to intervals such that for each interval (s_i,e_i) it assigns i a color c_j, where $s_i \leq j \leq e_i$, and maximum number of intervals which are assigned the same color is at most the lower bound value computed in Algorithm 1. We call this algorithm as Optimal X-filling Algorithm (DP-Fill). Running time of this algorithm is O ($k\ log\ k$), where k is the number of intervals

Algorithm 1: Computing Lower-Bound

Input: S= (s_1, e_1),(s_2, e_2) ... (s_k, e_k) be a sequence of intervals

Output: *Lower-Bound Value.*

1 Let $T_{i,j}$, where $i \leq j$, denotes number of intervals whose starting time is $\geq i$ and ending time is $\leq j$;

2 If $i > j$ then let $T_{i,j}$ = 0 else $T_{i,j}$ can be expressed recursively as follows : $T_{i,j}$ = $T_{i,j-1}$ + $T_{i+1,j}$ - $T_{i+1,j-1}$ + Number of intervals whose staring time is equal to i and ending time is equal to j.

/* Note that $T_{i+1,j-1}$ is subtracted since the set of intervals whose starting time is at least $i+1$ and ending time is at most $j-1$ are counted in both $T_{i,j-1}$, $T_{i+1,j}$. */

3 Let $max\_color = max(e_1, e_2 \dots e_k)$

4 $Lowerbound\ LB = max\{\lceil \frac{T_{i,j}}{j-i+1} \rceil | 1 \leq i \leq j \leq max\_color\}$

/* If we take any interval whose starting time is at least i and ending time at most j then we should assign a color c_k to this interval such that $i <= k <= j$. This means there exists a color c_k such that at least $\lceil \frac{T_{i,j}}{j-i+1} \rceil$ intervals are assigned color c_k, where $i <= k <= j$ */

Result: $return\ LB$

Algorithm 2: Assigning color to intervals

Input: S= (s_1, e_1),(s_2, e_2) ... (s_k, e_k) be a sequence of intervals, LB - lower-bound

Output: Intervals with assigned colors

1 Sort the intervals in S based on starting time.

2 Let *H be a min heap*. Each node of this heap can store information of an interval (starting time and ending time). Nodes of this heap are ordered by ending times of intervals i.e ending time of interval stored in a node is less than or equal to ending times of intervals stored in that node's children.

3 **for** $i = 1 \rightarrow n$ **do**

4 Insert into heap H all intervals whose starting time is equal to i.

/* if we take any interval in H starting time is at most i. */

5 Remove top l elements from heap and assign color c_i, where $l = min(current\ heap\ size, LB)$;

/* The reason for picking top elements and assigning colors c_i is we want to assign colors to intervals which are ending soon. We prove in section *Proof of correctness* that ending times of all these removed intervals are at least i. */

6 **end**

C. Proof of correctness

In the following paragraph we will prove that at the end of i^{th} iteration of Algorithm 2 ending times of all intervals contained in *min heap* are greater than i. This means each interval (s_i,e_i) it assigned a color c_j such that $s_i \leq j \leq e_i$.

Suppose at the end of some iteration i *min heap* contains an interval whose ending time is greater than i. Let i be such that it's value is minimum. Let $j < i$ such that number of intervals which are assigned color in j^{th} iteration is less than *lower bound*. Let j be such that it's value is maximum. If there is no such a j then let $j = 0$. Let $j < k < i$ such that in the k^{th} iteration the above algorithm assigned color to an interval whose ending time is more than i. Let k be such that it's value is maximum. If there is no such k then let $k = j$. Ending times and starting times of all intervals which are assigned color from $k + 1^{th}$ iteration to i^{th} iteration are less than or equal i and greater than k respectively. Note that the number of intervals which are assigned colors from $k + 1^{th}$ iteration to i^{th} is equal to $lowerbound*(i-k)$ and *min heap* contains an interval whose ending time is equal to i and starting time is greater than k. This implies number of intervals whose starting time is greater than k and ending time is less than or equal to i is more than $lowerbound * (i - k)$, which is a contradiction.

D. Test Vector Ordering Algorithm

For a given vector ordering, Algorithm 2 gives the optimum value of peak input toggles. Note that if the length of don't care stretches in the rows of matrix A (which is defined in section V-C) is high, then the optimum value of peak input toggles is small. To achieve such a large don't cares stretches in the rows of matrix A we propose the following test vector ordering algorithm, we call this ordering as *interleaved* test vector ordering (I-Ordering). Experimentally we observed that the number of times the *while loop* in Algorithm 3 gets executed is $O(log(n))$, where n is number of test vectors. This experimental observation is shown in Figures 2(a) and 2(b).

Fig 2(c) analyzes the don't care stretch statistics in the test cubes of `b19` circuit, for different test vector orderings. One can observe that I-Ordering increases the sizes of don't care stretches, which are finally exploited by the proposed X-filling Algorithm 2 to achieve the best possible peak toggle savings. The next section explains the experimental setup used to implement the described algorithms and compare the peak toggle savings obtained using the proposed algorithm to that of a commercial tool as well as techniques proposed in the prior literature.

VII. Experimental Setup and Results

We have considered the ITC'99 benchmark suite to validate our algorithms. Synthesis, test generation and place-and-route (PAR) phases of different benchmark circuits are performed using $DesignCompiler^{TM}$, $TetraMax^{TM}$ and $SoCEncounter^{TM}$ tools respectively, using a $45nm$ standard library. After PAR phase, the interconnect capacitances are extracted to compute actual power values. Tables II and III show comparison of peak input toggles for various X-filling methods w.r.t to test vector orderings given by the $TetraMax^{TM}$ tool and the XStat method [22] respectively. Each row in these tables corresponds to a benchmark circuit. The shaded cell in each row corresponds to best X-filling method among all X-filling methods for the given ordering. We can observe that the proposed DP-fill method consistently performed better than all the other X-filling methods, under both the test vector orderings. *This is because, under a given ordering, DP-fill is an optimal algorithm for minimizing peak input toggles*. Next we will evaluate the impact

Algorithm 3: Computing Test vector Ordering

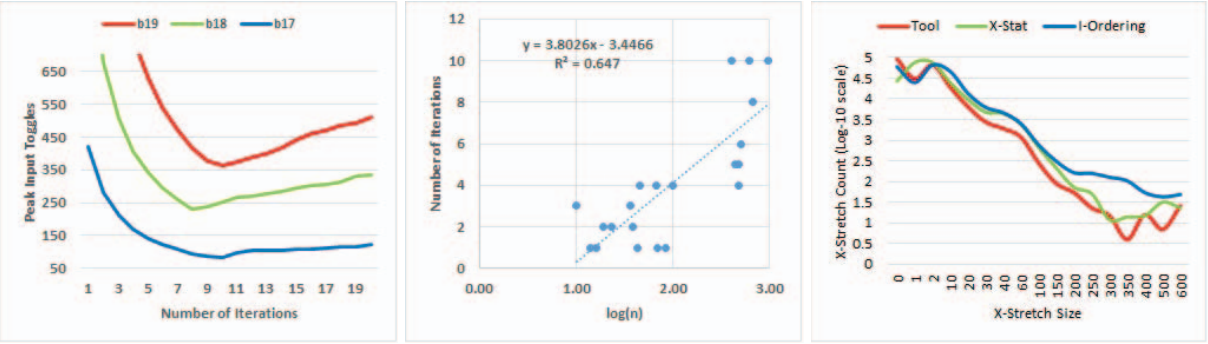
Input: T = T_1, T_2,.. T_n be the set of input test vectors.
Output: S = Sequence of input test vectors.

Let $T^{'}$= T_1^1, $T_2^{'}$,.. $T_n^{'}$ be an ordering of input test vectors such that the number of don't cares in $T_i^{'} \leq$ the number of don't cares in $T_{i+1}^{'}$, where $1 \leq i < n$.
Let $current\_optimal\_value = \infty$
Let $k = 0$
Let $exit\_flag = false$
while $exit\_flag = false$ **do**
  Let $k = k + 1$ /* Interleaving size */
  Let $S = \emptyset$
  for $i = 1 \rightarrow \lfloor \frac{n}{k+1} \rfloor$ **do**
    /* pick i^{th} vector from $T^{'}$ and append to S */
    $S = S,T_i^{'}$
    /* pick $n-(i-1)*k$ th vector to $n-(i-1)*k-k+1$ th vector from $T^{'}$ and append to S */
    $S = S,T_{n-(i-1)*k}^{'}, T_{n-(i-1)*k-1}^{'}, ..T_{n-(i-1)*k-k+1}^{'}$
  end
  Select all the vectors in $T^{'}$ which are not in S and add them to S, there can be at most k such vectors.
  Let $temp\_optimal\_value$ be the optimal bottleneck value computed on sequence S using Algorithm 2
  if $temp\_optimal\_value < current\_optimal\_value$ **then**
    $current\_optimal\_value = temp\_optimal\_value$;
  else
    $exit\_flag = true$;
  end
end

Result: $return\ S$

TABLE II
Peak Input Toggles : Tool-Ordering with different fillings. Ckt stands for circuit.

Ckt	MT-fill	R-fill	0-fill	1-fill	B-fill	DP-fill
b01	4	4	4	4	4	4
b02	4	4	4	4	4	4
b03	15	21	17	16	14	14
b04	41	50	47	45	39	39
b05	20	23	19	20	17	17
b06	4	4	5	4	4	4
b07	31	30	34	27	23	23
b08	20	20	20	18	14	12
b09	18	20	22	18	18	18
b10	12	19	17	15	10	10
b11	22	27	29	21	20	20
b12	63	76	62	89	59	58
b13	31	34	38	30	30	29
b14	181	180	194	159	157	156
b15	305	334	344	298	292	282
b17	916	923	943	880	871	841
b18	2134	2167	2251	2114	2066	2009
b19	3926	4099	4201	3955	3819	3753
b20	309	314	315	305	302	299
b21	317	307	315	305	276	260
b22	489	494	507	471	472	466



(a) Number of Iterations vs Peak Input Toggles

(b) Optimum Number of Iterations vs log(n)

(c) Don't care stretch statistics for `b19` circuit (Tool vs X-Stat [22] vs I-Ordering)

Fig. 2. Algorithm Iterations

TABLE III
PEAK INPUT TOGGLES : XSTAT-ORDERING [22] WITH DIFFERENT FILLINGS. CKT STANDS FOR CIRCUIT.

Ckt	MT-fill	R-fill	0-fill	1-fill	B-fill	DP-fill
b01	3	4	4	3	3	3
b02	4	4	4	4	4	4
b03	15	19	18	15	8	7
b04	45	52	47	43	25	24
b05	21	24	21	23	15	14
b06	5	4	5	5	5	4
b07	27	33	38	25	15	14
b08	16	20	18	15	8	7
b09	20	19	17	16	14	14
b10	14	20	16	14	10	7
b11	18	26	22	20	10	9
b12	60	76	99	68	31	31
b13	37	32	28	23	17	17
b14	181	164	208	152	79	79
b15	308	277	314	198	144	144
b17	912	774	953	680	421	421
b18	2130	1752	2200	1569	1011	1008
b19	3926	3457	4340	3168	1877	1877
b20	314	291	352	297	152	152
b21	288	290	346	237	130	130
b22	483	419	475	440	237	234

TABLE IV
PEAK INPUT TOGGLES : I-ORDERING WITH DIFFERENT FILLINGS. CKT STANDS FOR CIRCUIT.

Ckt	MT-fill	R-fill	0-fill	1-fill	B-fill	DP-fill
b01	3	4	4	3	3	3
b02	3	3	3	3	3	3
b03	12	19	15	15	8	6
b04	41	45	43	39	23	15
b05	20	22	21	23	15	14
b06	4	4	4	4	4	4
b07	24	31	38	23	15	11
b08	16	18	16	14	8	6
b09	14	18	16	16	11	11
b10	10	18	14	13	9	7
b11	15	25	22	18	10	9
b12	59	72	99	65	30	15
b13	28	31	28	23	15	10
b14	168	158	208	148	77	40
b15	296	267	314	193	141	33
b17	882	770	953	676	419	85
b18	2030	1741	2200	1550	980	232
b19	3862	3436	4340	3167	1871	364
b20	301	285	352	284	143	65
b21	280	286	333	237	129	67
b22	451	409	475	425	210	91

of the proposed test vector ordering technique (I-ordering) under different X-filling schemes explained previously including DP-fill. Table IV shows the results for the same. It can be seen that DP-fill method consistently performed better than all the other X-filling methods under the proposed I-ordering scheme. Additionally, it can be observed from Tables II, III and IV that the combination of *I-ordering + DP-fill* is most effective in reducing peak toggles, especially for the larger circuits. Next, we will compare *I-ordering + DP-fill* with other existing technique in the literature.

Table V shows the *peak input toggles* comparison between the proposed technique and best known existing techniques. Column 1 shows the minimum peak input toggles obtained among all aforementioned X-filling methods, under test vector ordering given by the $TetraMax^{TM}$ tool. Columns 2, 3 and 4 show minimum peak input toggles obtained using the techniques proposed in [20], [21] and [22] respectively. Columns 6-9 of this table show the percentage improvement of proposed *I-ordering + DP-fill* technique over these best known low power techniques for the LOS scheme. It is evident that the proposed technique outperforms all the existing techniques for most of the benchmark circuits and the percentage improvement consistently increases with increase in circuit size.

Tables II, III and IV correspond to different X-fillings for a given ordering. So, in all cases DP-fill gave the optimal solution of lowest peak input toggles for **all the benchmarks**. On the other hand, the orderings employed by the techniques proposed in [20], [21] or [22] need not necessarily be same as I-ordering. Thus, unlike earlier comparisons made in tables II, III and IV, we cannot give a performance guarantee for *I-ordering + DP-fill* over techniques proposed in [20], [21] or [22]. However, it is interesting to note from Table V that the proposed *I-ordering + DP-fill* technique actually outperforms all the these techniques for **most of the benchmarks** and the percentage improvement increases with increase in circuit

size. This is because I-ordering as well as DP-fill are both designed for reducing peak toggles when test sets are dominated by don't cares and practically the test sets of most of these circuits are dominated by don't cares as shown earlier in Table I.

TABLE V
PEAK INPUT TOGGLES : COMPARISON OF PROPOSED *I-Ordering + DP-fill* METHOD OVER EXISTING *Ordering + X-filling* METHODS

	Peak Input Toggles					*%Improvement over*			
Ckt	Tool	ISA [20]	Adj-fill [21]	XStat [22]	Proposed	Tool	ISA [20]	Adj-fill [21]	XStat [22]
b01	4	2	4	3	3	25	-50	25	0
b02	4	1	3	4	3	25	-200	0	25
b03	14	8	6	8	6	57.1	25	0	25
b04	39	31	29	25	15	61.5	51.6	48.3	40
b05	17	12	19	15	14	17.6	-16.7	26.3	6.7
b06	4	2	4	4	4	0	-100	0	0
b07	23	18	17	15	11	52.2	38.9	35.3	26.7
b08	14	10	9	8	6	57.1	40	33.3	25
b09	18	11	17	14	11	38.9	0	35.3	21.4
b10	10	9	9	10	7	30	22.2	22.2	30
b11	20	12	18	10	9	55	25	50	10
b12	59	46	77	31	15	74.6	67.4	80.5	51.6
b13	30	20	26	17	10	66.7	50	61.5	41.2
b14	157	89	69	79	40	74.5	55.1	42	49.4
b15	292	172	149	144	33	88.7	80.8	77.9	77.1
b17	871	573	438	421	85	90.2	85.2	80.6	79.8
b18	2066	1384	1065	1011	232	88.8	83.2	78.2	77.1
b19	3819	2609	2100	1877	364	90.5	86	82.7	80.6
b20	302	214	198	152	65	78.5	69.6	67.2	57.2
b21	276	181	182	130	67	75.7	63	63.2	48.5
b22	471	324	232	237	91	80.7	71.9	60.8	61.6

Table VI shows the comparisons of actual peak power dissipation during test, between the proposed technique and the existing techniques. It can be observed that similar to peak toggles savings, the proposed technique performs better than all the existing techniques in peak power savings for most of the benchmarks and percentage improvement increases with increase in circuit size. This can be attributed to well known fact that there is a good correlation between input toggles and circuit toggles, as explained in [20]. Additionally, we can observe that the magnitude of improvement in tables V and VI is not same. The difference is due to the fact that the relation between input toggles and circuit toggles is not perfectly linear and while computing actual power dissipation of the circuit, we need to take interconnect capacitances into account. However, our proposed technique outperforms all the existing techniques considerably, in both peak input toggles as well as actual peak circuit power.

TABLE VI
PEAK CIRCUIT POWER (IN μW): COMPARISON OF PROPOSED *I-Ordering + DP-fill* METHOD OVER EXISTING *Ordering + X-filling* METHODS

	Peak Circuit Power					*%Improvement over*			
Ckt	Tool	ISA [20]	Adj-fill [21]	XStat [22]	Proposed	Tool	ISA [20]	Adj-fill [21]	XStat [22]
b01	3.8	2.3	3.3	3.1	3.1	18.8	-33.1	6.1	0
b02	2.4	1.5	2.8	2.6	2.6	-6.2	-68.3	7.3	0
b03	5.6	4	4.6	3.9	4.2	25	-5.5	9.2	-5.6
b04	17.2	17.1	15.8	16.9	14.8	14	13.9	6.6	12.7
b05	15.6	13.6	16.4	14.6	14.9	4.4	-9.8	9	-2
b06	4.4	2.6	4.4	4.3	4.4	0.9	-67.2	-0.1	-1.7
b07	15.7	14.8	13.1	14.6	13.3	15.7	10.6	-1.5	8.9
b08	7.8	6.8	8.1	7.7	6.3	18.5	6.8	21.5	18.1
b09	9.8	8.4	10.7	8.9	7.4	24.7	12.1	30.8	17.2
b10	9.3	8.8	9	8.7	8.2	11.6	6.5	9.2	6.3
b11	16.4	15.4	15.2	14.6	13.9	15.2	9.6	8.9	4.8
b12	56.5	49.4	58.4	39.3	36.4	35.5	26.3	37.6	7.2
b13	18	13.7	15.1	14.7	10.9	39.4	20.1	27.6	25.3
b14	99.3	101.7	99	86.5	85.4	14	16.1	13.8	1.3
b15	197.1	171	155.3	140.4	122	38.1	28.7	21.4	13.1
b17	1085.5	847.1	665.5	641.7	431.6	60.2	49.1	35.1	32.7
b18	3350.7	2405.3	2012.2	1761	1192	64.4	50.4	40.8	32.3
b19	7621.6	6708.3	5885	4135	2699.4	64.6	59.8	54.1	34.7
b20	252.8	243	214.8	202.6	195.3	22.7	19.6	9.1	3.6
b21	248.4	226.1	223.8	183.2	166.4	33	26.4	25.6	9.2
b22	395.6	372.8	328.9	304.8	277.1	30	25.7	15.8	9.1

VIII. CONCLUSIONS

We address the problem of excessive peak capture power that leads to false delay failures. Since the test cubes are dominated by X-bits and there is a good correlation of input toggles to circuit toggles, X-filling is very effective for reducing peak capture power. We map the problem of optimal X-filling to a variant of *interval coloring problem*, so as to minimize peak input toggles of the circuit. This algorithm leads to significant reductions in peak capture power dissipated inside the circuit. To the best of our knowledge, *this is the first ever reported X-filling algorithm that is optimal*.

REFERENCES

[1] S. Gerstendorfer et. al, "Minimized Power Consumption for Scan-based BIST", International Test Conference, IEEE, 1999, pp. 77-84.

[2] P. Girard, "Survey of low-power testing of VLSI circuits", IEEE Design and Test of Computers, 2002, vol.19, no.3, pp. 80-90.

[3] J. Saxena et. al, "A case study of IR-drop in structured at-speed testing", International Test Conference, IEEE, 2003, pp. 1098-1104.

[4] P. Pant et. al, "Lessons from at-speed scan deployment on an Intel Itanium microprocessor", IEEE ITC, 2010, pp.1-8.

[5] N. Parimi and Sun Xiaoling, "Toggle-masking for test-per-scan VLSI circuits", IEEE DFT, 2004, pp. 332-338.

[6] S. Bhunia et. al, "Low-power scan design using first-level supply gating", IEEE TVLSI, Vol.13, No.3, 2005, pp. 384-395.

[7] V. R. Devanathan et. al, "PMScan : A power-managed scan for simultaneous reduction of dynamic and leakage power during scan test", International Test Conference, IEEE, 2007, pp. 1-9.

[8] S. Potluri et. al, "LPScan: An algorithm for supply scaling and switching activity minimization during test", International Conference on Computer Design, IEEE, 2013, pp. 463-466.

[9] P. Girard et. al, "Circuit partitioning for low power BIST design with minimized peak power consumption", IEEE ATS 1999, pp. 89-94.

[10] K. J. Lee et. al, "Peak-power reduction for multiple-scan circuits during test application", Asian Test Symposium, IEEE, 2000, pp. 453-458.

[11] S. Almukhaizim and O. Sinanoglu, "Peak Power Reduction Through Dynamic Partitioning of Scan Chains", International Test Conference, IEEE, 2008, pp.1-10.

[12] X. Lin and J. Rajski, "Test Power Reduction by Blocking Scan Cell Outputs", Asian Test Symposium, IEEE, 2008, pp. 329-336.

[13] V. Dabholkar et. al,"Techniques for Minimizing Power Dissipation in Scan and Combinational Circuits During Test Application", IEEE TCAD, Vol. 17, No. 2, 1998, pp. 1325-1333.

[14] A. Satya Trinadh et. al, "An Efficient Heuristic for Peak Capture-Power Minimization during Scan-based Test", ASP Journal of Low Power Electronics, Vol. 9, No. 2, 2013.

[15] R. Sankaralingam, N. A. Touba, "Controlling peak power during scan testing" VLSI Test Symposium, IEEE, 2002, pp. 153-159.

[16] V. R. Devanathan et. al, "Glitch-Aware Pattern Generation and Optimization Framework for Power-Safe Scan Test", VLSI Test Symposium, IEEE , 2007, pp.167-172.

[17] C. Yao et. al, "Power and thermal constrained test scheduling under deep submicron technologies", IEEE TCAD, Vol. 30, No. 2, 2011, pp. 317-322.

[18] S. Bhunia et. al, "First level hold: a novel low-overhead delay fault testing technique", IEEE DFT 2004, pp. 314-315.

[19] V. R. Devanathan et. al, "A stochastic pattern generation and optimization framework for variation-tolerant, power-safe scan test", International Test Conference, IEEE, 2007, pp.1-10.

[20] P. Girard et.al, "Reducing power consumption during test application by test vector ordering", IEEE ISCAS, 1998, pp. 296-299.

[21] F. Wu et. al, "Power reduction through X-filling of transition fault test vectors for LOS testing", IEEE DTIS, 2011, pp.1-6.

[22] A. Satya Trinadh et. al, "XStat: Statistical X-Filling Algorithm for Peak Capture Power Reduction in Scan Tests", ASP Journal of Low Power Electronics, Vol. 10, No. 1, 2014.

[23] D. B. West, "Introduction to Graph Theory", Second Edition, Prentice Hall, 2000.