Bridging Fault Test Method with Adaptive Power Management Awareness

Saqib Khursheed, Urban Ingelsson, *Student Member, IEEE,* Paul Rosinger, Bashir M. Al-Hashimi, *Senior Member, IEEE,* and Peter Harrod, *Senior Member, IEEE*

***Abstract*—A key design constraint of circuits used in hand-held devices is the power consumption, mainly due to battery life limitations. Adaptive power management (APM) techniques aim to increase the battery life of such devices by adjusting the supply voltage and operating frequency, and thus the power consumption, according to the workload. Testing for resistive bridging defects in APM-enabled designs raises a number of challenges due to their complex analog behavior. Testing at more than one supply voltage setting can be employed to improve defect coverage in such systems, however, switching between several supply voltage settings has a detrimental impact on the overall cost of test. This paper proposes a multi-Vdd automatic test generation method which delivers 100% resistive bridging defect coverage and also a way of reducing the number of supply voltage settings required during test through test point insertion. The proposed techniques have been experimentally validated using a number of benchmark circuits.**

***Index Terms*—Adaptive Power Management, Resistive Bridging Faults, Test Generation, Test Points**

I. Introduction

ENERGY-EFFICIENT design is becoming more important with technology scaling and with high performance requirements, especially for portable, battery-driven appliances. Several adaptive power management methods have been employed in a wide range of consumer electronics to optimize their power consumption. A popular adaptive power management technique is scaling the supply voltage and operating frequency according to the processing load [1], as implemented in several state-of-the-art processors [2] and [3]. Typically, a design with adaptive power management has a set of discrete supply voltage/frequency settings it can switch between depending on the current workload and power saving mode. Manufacturing test needs to ensure that such a design operates correctly over the entire set of supply voltage/frequency settings, while keeping the overall cost of test low.

Resistive bridges represent a major class of defects for deep submicron CMOS and have received increased attention. Research has investigated modeling [4], [5], [6], [7], [8], [9], [10], simulation [6], [9] and test generation [6], [11], [12], [13], [14] for resistive bridging faults (RBF). It has been shown in [8] that the fault coverage of a test set targeting resistive bridging faults can vary with the supply voltage used during test. This means that, depending on the operating Vdd setting, a given RBF may or may not affect the correct operation of the design. Consequently, to ensure high fault coverage for a design that needs to operate at a number of different Vdds, it may be necessary to perform testing at more than one Vdd to detect faults which manifest themselves only at particular Vdds. The first aim of this paper is to propose an automatic test generation method targeting multiple Vdd settings. The second aim of this work is to demonstrate a method of reducing the number of Vdd settings required during test without affecting the defect coverage.

The paper is organised as follows: Section II gives background information and summarizes the prior work on RBF testing. The motivation of multi-Vdd testing is discussed in Section III. In Section IV-A we present a deterministic test generation method targeting RBF at multiple Vdd settings and report the experimental validation results. A method to reduce the number of Vdd settings required during test by using test points is presented in Section V. Concluding remarks are given in Section VI.

II. Background and prior work

The main difficulty in RBF test generation arises from the fact that the bridging resistance is a continuous parameter which is not known in advance. A recent approach based on interval algebra [8], [9] allowed treating the whole continuum of bridge resistance values R_{sh} from 0Ω to ∞ by handling a

Manuscript received September 6, 2007; revised November 9, 2007. The authors would like to acknowledge EPSRC(UK) for funding this work under grant no. EP/DO57663/1. A preliminary version of this work has been accepted at ATS'07 (http://eprints.ecs.soton.ac.uk/14203). The work presented in this paper brings the following extensions: improved test generation algorithm including a post processing test set reduction step and real bridge list extraction from placed and routed designs, improved test point selection algorithm including algorithms for control and observation point minimization. The experimental validation also includes a larger number of benchmark designs.

S. Khursheed, U. Ingelsson, P. Rosinger and B. M. Al-Hashimi are with the School of Electronics and Computer Science, University of Southampton, SO17 1BJ, UK. (Telephone: +44-23-8059-3716, Fax: +44-23-8059-2901, Email: {ssk06r, bui05r, pmr, bmah}@ecs.soton.ac.uk)

P. Harrod is with ARM Limited, Cambridge, UK. (Email: peter.harrod@arm.com)

0000–0000/00$00.00 © 2007 IEEE

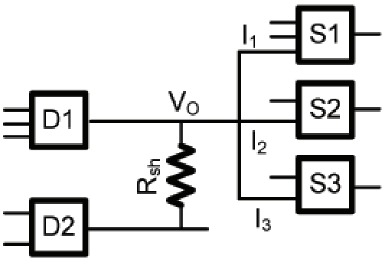

Fig. 1. Bridging fault example

finite number of discrete intervals. The key observation which enables this method is that a resistive bridge changes the voltages on the bridged lines from 0V (logic-0) or Vdd (logic-1) to some intermediate values, which will be different for different R_{sh} values. The logic behavior of the physical defect can be expressed in terms of the logic values perceived by the gate inputs driven by the bridged nets based on their specific input threshold voltage.

A typical bridging fault scenario is illustrated in Fig. 1. D1 and D2 are the gates driving the bridged nets, while S1, S2 and S3 are successor gates, i.e. gates having inputs driven by one of the bridged nets. The resistive bridge affects the logic behavior only when the two bridged nets are driven at opposite logic values. For example, let us consider the case when the output of D1 is driven high and the output of D2 is driven low. The dependence of the voltage level on the output of D1 (V_O) on the equivalent resistance of the physical bridge is shown in Fig. 2. The deviation of V_O from the ideal voltage level (Vdd) is highest for small values of R_{sh} and decreases for larger values of R_{sh}. To translate this analog behavior into the digital domain, the input threshold voltage levels V_{th1}, V_{th2} and V_{th3} of the successor gates S1, S2 and S3 have been added to the V_O plot. For each value of the bridging resistance R_{sh}, the logic values read by inputs I_1, I_2 and I_3 can be determined by comparing V_O with the input threshold voltage of the corresponding input. These values are shown in the second part of Fig. 2. Crosses are used to mark the faulty logic values and ticks to mark the correct ones. It can be seen that, for bridges with $R_{sh} > R_3$, the logic behavior at the fault site is fault-free (all inputs read the correct value), while for bridges with R_{sh} between 0 and R_3, one or more of the successor inputs are reading a faulty logic value. The R_{sh} value corresponding to R_3 is normally referred to as “critical resistance” as it represents the crossing point between faulty and correct logic behavior. Methods for determining the critical resistance have been presented in several publications [6], [9].

A number of bridging resistance intervals can be identified based on the corresponding logic behavior. For example, bridges with $R_{sh} \in [0, R_1]$ exhibit the same faulty behavior in the digital domain (all successor inputs read the faulty logic value), similarly, for bridges with $R_{sh} \in [R_1, R_2]$, successor gates S2 and S3 read the faulty value, while S1 reads the correct value, and finally, for bridges with $R_{sh} \in [R_2, R_3]$ only S3 reads a faulty value while the other two successor gates read the correct logic value. Consequently, each interval $[R_i, R_{i+1}]$ corresponds to a distinct logic behavior occurring at the bridging fault site. The logic behavior at the fault site can be captured using a data structure which will be further referred to as logic state configuration (LSC). An LSC consists of the logic values at the inputs of the driving gates and the logic values detected by the inputs of the successor gates. LSCs capturing faulty logic behavior can be looked at as logic fault models. An LSC is said to be non-redundant, if there is at least one test pattern which can justify the net values specified by the LSC and also make the faulty behavior observable at the primary outputs. An LSC for which no such test pattern exists is referred to as a redundant LSC. This means that redundant LSCs cannot occur during the functional operation of the circuit, and consequently only non-redundant LSCs have to be targeted during test generation in order to ensure correct operation of the circuit. The union of the resistance intervals corresponding to non-redundant LSCs forms the Global Analogue Detectability Interval (G-ADI) [9]. Basically, G-ADI represents the entire range of detectable physical defects. Given a test set TS, the Covered Analogue Detectability Interval (C-ADI) represents the range of physical defects detected by TS. The C-ADI for a bridging defect is the union of one or more disjoint resistance intervals, the union of intervals corresponding to non-redundant LSCs [4], [8], [9], [13]. Throughout this paper, the quality of a test set is estimated by measuring how much of the G-ADI has been covered by the C-ADI. When the C-ADI of test set TS is identical to the G-ADI of fault f, TS is said to achieve full fault coverage for f.

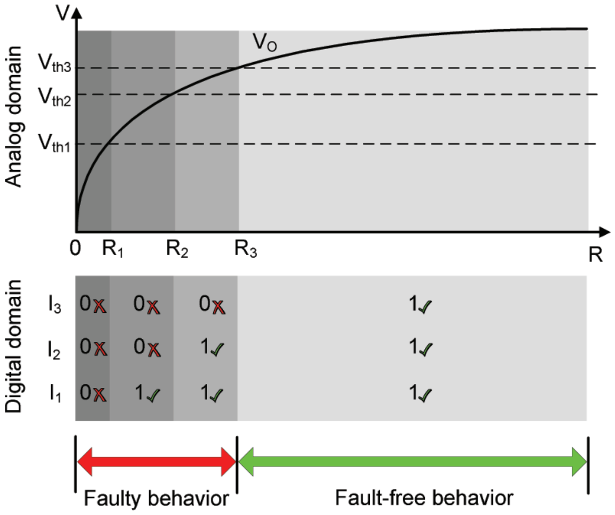

Fig. 2. Bridging fault behavior

Several test generation methods for resistive bridging faults have been proposed [6], [11], [14] and more recently [12], [13].

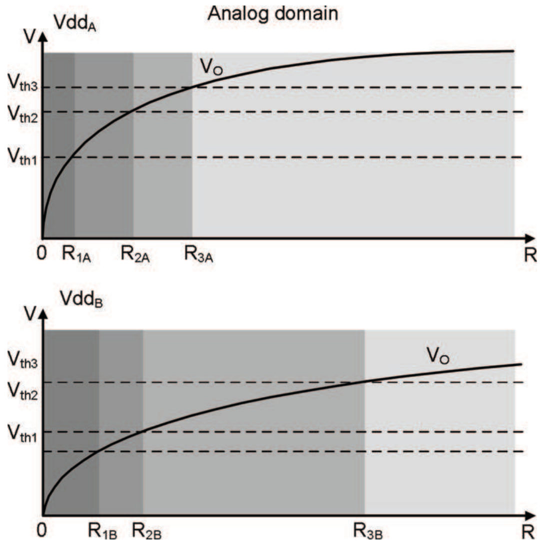

Fig. 3. Effect of supply voltage on bridging fault behavior: Analog domain

The method presented in [11] is to guarantee the application of all possible values at the bridge site without detailed electrical analysis. In [12], the effect of a bridge on a node with fanout is modeled as a multiple line stuck-at fault. The study in [6], identifies only the largest resistance interval and determines the corresponding test pattern. In contrast to [6], the sectioning approach from [14] considers all the sections (resistance intervals) $[R_i, R_{i+1}]$. For each section, the corresponding LSC (and associated faulty logical behavior) is identified. This avoids the need for dealing with the resistance intervals and improves the test quality compared with [6], but the number of considered faults grows.

In [13], the authors combined the advantages of the interval based [6] and the sectioning approach [14] into a more efficient test generation procedure by targeting the section with the highest boundaries first. Interval based fault simulation is then used to identify all other sections covered by the test pattern. It should be noted that all test generation methods described above [6], [11], [12], [13], [14] are intended for a fixed supply voltage setting, i.e. all tests are applied at the same supply voltage. In the next section we explain why it is sometimes necessary to use more than one Vdd setting during test to ensure full bridging defect coverage for adaptive power management enabled designs.

III. Motivation of Multi-Vdd testing

This section provides an analysis of the effect of varying supply voltage on bridging fault behavior, which provides the starting point for the work presented in this paper. Figures 3, 4 and 5 show the relation between the voltage on the output of gate D1 (Fig. 1) and the bridging resistance for two different supply voltages Vdd_A and Vdd_B. The diagrams in Fig. 4 show how the analog behavior at the fault site translates

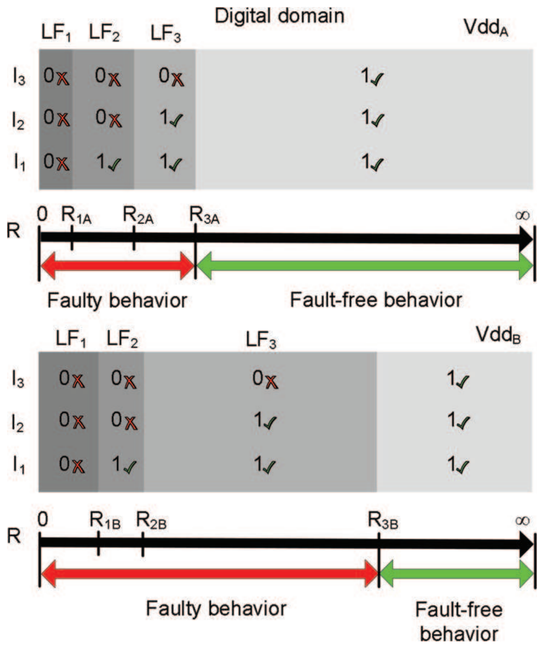

Fig. 4. Effect of supply voltage on bridging fault behavior: Digital domain

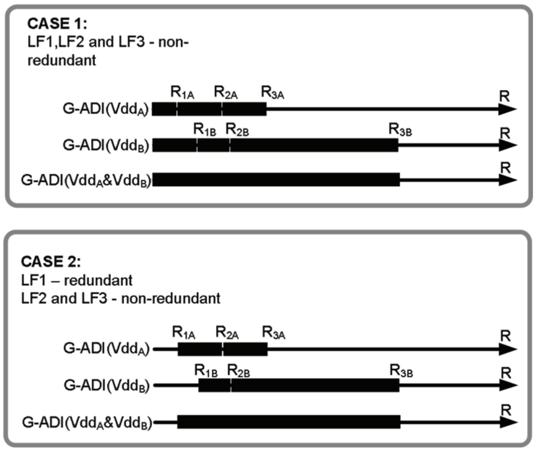
CASE 1:
LF1,LF2 and LF3 - non-redundant
G-ADI(Vdd_A) R1A R2A R3A R
G-ADI(Vdd_B) R1B R2B R3B R
G-ADI(Vdd_A&Vdd_B) R
CASE 2:
LF1 – redundant
LF2 and LF3 - non-redundant
G-ADI(Vdd_A) R1A R2A R3A R
G-ADI(Vdd_B) R1B R2B R3B R
G-ADI(Vdd_A&Vdd_B) R

Fig. 5. Effect of supply voltage on bridging fault behavior: Observable bridging resistance ranges

into the digital domain. In this example, three distinct logic faults LF1, LF2 and LF3 could be identified for each Vdd setting. However, because the voltage level on the output of D1 does not scale linearly with the input threshold voltages of S1, S2 and S3 when changing the supply voltage (this has been validated through SPICE simulations), the resistance intervals corresponding to LF1, LF2 and LF3 differ from one supply voltage setting to another. This means that a test pattern targeting a particular logic fault will detect different ranges of physical defects when applied at different supply voltage settings. For example, at Vdd_A, a test pattern targeting LF3 will detect bridges with $R_{sh} \in [R_{2A}, R_{3A}]$, while at Vdd_B it will detect a much wider range of physical bridges ($R_{sh} \in [R_{2B}, R_{3B}]$). Analysing this from a different perspective, a bridge with $R_{sh} = R_{3B}$ will cause a logic fault at Vdd_B but not at Vdd_A. To demonstrate the need for using multiple Vdd settings during test we use the following two scenarios. In Case 1 (Fig. 5) all three logic faults LF1, LF2 and LF3 are non-redundant. Fig. 5 shows the ranges of bridging resistance corresponding to faulty logic behavior for the two Vdd settings (basically the G-ADI sets corresponding to the two Vdd settings). Previous work on test generation for bridging faults [13] has used the concept of G-ADI assuming a fixed Vdd scenario. We have extended the concept of G-ADI to capture the dependence of the bridging fault behavior on the supply voltage by defining the multi-Vdd G-ADI as the union of Vdd specific G-ADIs for a given design.

$G\text{-}ADI = \bigcup G\text{-}ADI(Vdd_i)$

The overall G-ADI consists of the union of the two Vdd specific G-ADI sets. It can be seen that $G\text{-}ADI(Vdd_A)$ represents about 45% of the overall G-ADI while $G\text{-}ADI(Vdd_B)$ fully covers the overall G-ADI. This means that a test set detecting LF1, LF2 and LF3 will achieve full bridging defect coverage when applied at Vdd_B. In Case 2 from Fig. 5, only LF2 and LF3 are non-redundant, which means that there is no test pattern which can detect LF1. In this case, $G\text{-}ADI(Vdd_A)$ represents about 30% of the overall G-ADI while $G\text{-}ADI(Vdd_B)$ represents about 90% of the overall G-ADI. This means that full bridging fault coverage cannot be achieved using a single Vdd setting.

From the previous analysis it can be concluded that to achieve full G-ADI coverage in a variable Vdd system, it may be necessary to apply tests at several Vdd settings. Instead of repeating the same test at all Vdd settings, which would lead to long testing times and consequently would increase the manufacturing cost, it would be desirable to be able to determine for each Vdd settings only the test patterns which effectively contribute to the overall defect coverage. A methodology for achieving this is presented in Section IV-A.

Although in this work we are considering equal probabilities for defects with different resistance values, the real life occurrence distribution of bridge resistance may make some resistance values unlikely to be found on a fabricated circuit. The impact of a real life distribution on multi-Vdd testing of RBFs is addressed in Fig. 6. It shows the distribution of defects which cannot be detected at 0.8V Vdd (which would be a preferred Vdd for a 1.2V process according to [4], [8]). The distribution in Fig. 6 is based on seven of the medium and large size ISCAS benchmarks. The random spread of these defects across the resistance range suggests that to ensure high defect coverage it will be necessary to test at more than one Vdd, even if the defect occurrence distribution corresponding to a particular manufacturing process is concentrated around a certain resistance range.

Switching between different Vdd settings during test is not a trivial task, and therefore a large number of Vdd settings required during test can have a detrimental effect on the overall cost of test. Consequently it would be desirable to keep the number of Vdd settings required during test to a minimum. By analysing the scenario described in Case 2 (Fig. 5), it can be seen that full bridging defect coverage could be achieved using a single Vdd setting (Vdd_B), if the logic fault corresponding to the resistance interval $[R_{1A}, R_{1B}]$, LF1 in this case, would become detectable, i.e. non-redundant. Based on this observation, we propose a test point insertion methodology which aims to increase the defect coverage at specific Vdd settings by exposing resistance intervals corresponding to redundant logic faults, which consequently reduces the total number of Vdd settings during test. The proposed test point insertion methodology is presented in Section V.

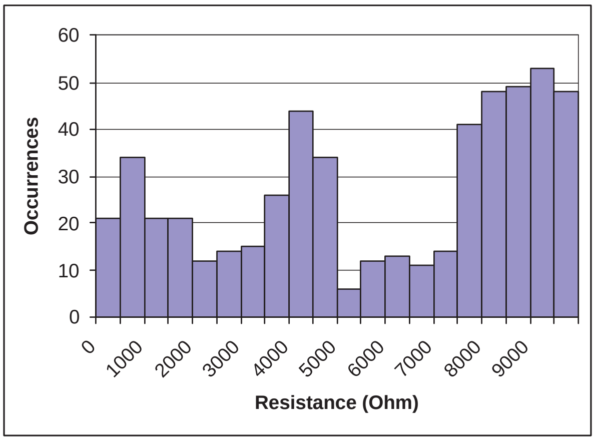

Fig. 6. The distribution of resistance values that cannot be detected at 0.8V Vdd

IV. Proposed multi-Vdd test generation

The proposed multi-Vdd test generation flow is shown in Fig. 7. The multi-Vdd test generation flow starts from the placed and routed design to generate a realistic list of bridges and computes a number of Vdd-specific test sets which provide 100% bridging defect coverage.

A. Methodology

The proposed test generation flow starts by generating the bridge list by coupling capacitance extraction on the placed and routed design, where each pair of nets that are capacitively coupled are considered a likely bridge. The multi-Vdd test generation procedure, detailed in Fig. 8, generates Vdd-specific test sets for the given bridge list and synthesised netlist. An optional post processing step, explained later in this section, can be performed to further reduce the overall number of test patterns.

The test generation algorithm starts by identifying for each bridge the resistance intervals corresponding to faulty logic behaviour (at least one of the inputs fed by the bridged nets

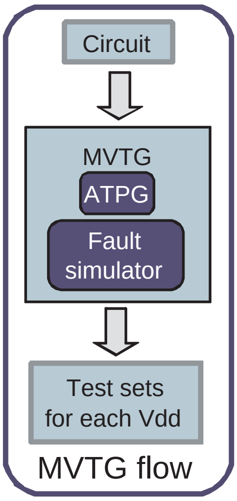

Fig. 7. Flow for the test generation algorithm

```
Input: netlist, Vdd settings $V = \{v\}$, Bridges $B = \{b\}$
Output: Multi-Vdd tests with 100% defect coverage
 1: for all Bridge $b \in B$ do
 2:   for all $v \in V$ do
 3:     compute all $(lsc, ri)$ for $(b, v)$
        // $ri$ abbreviates Resistance Interval
 4:     $LFS(b) := LFS(b) \bigcup \{(lsc, v, ri)\}$
 5:   end for
 6: end for
 7: while $(B \neq \emptyset)$ do
 8:   Get a bridge $b$ from $B$
 9:   for all $lfs \in LFS(b)$ do
10:     if ($lfs$ not covered) then
11:       ATPG$(b, LSC(lfs))$
12:       if (ATPG found a test pattern $tp$) then
13:         $TP(lfs) := tp$
14:         mark $lfs$ as covered in $c\_LFS(b)$
15:       end if
16:     end if
17:   end for
18:   $TS\_LFS :=$ select_LFSs $(c\_LFS(b))$
19:   $B := B \backslash \{b\}$
20:   for all LFS $\overline{lfs} \in TS\_LFS$ do
21:     $v := V(\overline{lfs})$
22:     $tp := TP(\overline{lfs})$
23:     $TestSet(v) := TestSet(v) \bigcup \{tp\}$
24:     fault simulate on $\overline{b} \in B$ and mark all detected LFS
        as covered in $c\_LFS(\overline{b})$
25:   end for
26: end while
```

Fig. 8. Multi-Vdd Test Generation (MVTG)

reads an incorrect logic value) for each Vdd setting (lines 1-6). This is achieved by determining the voltage levels on the bridged nets through SPICE simulations and comparing them with the threshold voltages of the inputs fed by them. One simulation is done for each possible input assignment to the gates that drive the bridged nets. To speed-up the process, the SPICE simulation results are stored into a database, such that for subsequent bridges with the same type of driving gates, the time-consuming SPICE simulations are replaced with fast database look-ups. The logic behaviour corresponding to each resistance interval is captured using logic state configuration (LSC) data structures, as explained in Sections II and III. An LSC holds the Boolean values at the inputs of the gates driving the bridge and the Boolean values seen by the inputs fed by the bridged nets. The tuple consisting of an LSC, its corresponding Vdd setting v and resistance interval ri will be further referred to as a Logic Fault Set (LFS), with the following semantic: a test pattern detecting the faulty behaviour described by lsc covers the resistance interval ri when applied at Vdd setting v.

The algorithm continues by computing for every bridge b the test patterns for resistance intervals that are not yet covered (lines 9-17). All generated test patterns are included in a set of candidate test-patterns for b by adding the corresponding LFS to $c\_LFS(b)$. The resulting set of test patterns comprises all possible ways of covering the detectable resistance intervals corresponding to the bridge b. This test set is minimised using an integer linear programming formulation of a minimum set covering problem (select_LFSs on line 18). The result TS_LFS is the minimum set of test patterns that covers all detectable bridge resistance (the scope of G-ADI). The selected test set (TS_LFS) is fault simulated using all remaining bridges and the resistance intervals corresponding to detected LFSs are marked as covered, so they are not targeted anymore in subsequent iterations (lines 20-25). For fault simulation we have used a method similar to the one employed in [9].

An optional post-processing step can be employed to further reduce the size of the multi-Vdd test set size obtained as described earlier. Initially, each test pattern is fault-simulated using the entire bridge list to compute its individual defect coverage. The test set is then sorted in descending order of the test pattern defect coverage. The test patterns in the ordered test set are fault simulated again, this time dropping resistance intervals as soon as they are detected. If for a particular test pattern (tp) no resistance interval was dropped, tp is removed from the test set as it does not contribute to the overall defect coverage since all resistance intervals targeted by tp had been already detected by the previous test patterns in the ordered test set.

B. Experimental results

The proposed multi-Vdd test generation method has been implemented as a tool chain (Fig. 7) consisting of an automatic test generator and a fault simulator and has been validated experimentally using a number of ISCAS 85 and ISCAS 89

TABLE I
MULTI-VDD TEST GENERATION RESULTS

		Vdd 0.8V	Vdd 1.0V	Vdd 1.2V				
Design	RBF #	#tp TS 1	#tp TS 2	#tp TS 3	Sum #tp	CPU time	ATPG %	Sim/RBF
c1355	80	39			39	21	71	557
c1908	98	57			57	13	58	296
c2670	104	67			67	27	41	269
c3540	363	184	6	1	191	340	62	568
c7552	577	281		1	282	1049	66	552
s838	34	26	2		28	6	54	243
s1488	435	144	2		146	193	10	265
s5378	305	214			214	308	32	914
s9234	223	132	2		134	130	43	1068
s13207	358	192	5	1	198	2454	53	1291
s15850	943	324	4	5	333	11835	42	417
s35932	1170	547	50	63	660	11233	53	263

TABLE II
MULTI-VDD TEST-SETS DEFECT COVERAGE

	TS 1	TS 1&2	TS 1&3	TS 1&2&3
Design	defect coverage	defect coverage	defect coverage	defect coverage
c1355	100.0			
c1908	100.0			
c2670	100.0			
c3540	99.20	99.96		100.0
c7552	99.95		100.0	
s838	95.04	100.0		
s1488	99.98	100.0		
s5378	100.0			
s9234	99.87	100.0		
s13207	99.57	99.92		100.0
s15850	99.84	99.94		100.0
s35932	93.95	98.48		100.0

benchmark circuits. The sequential circuits were treated as combinational by assuming full scan-chains and only non-feedback bridges have been targeted.

The benchmark circuits were synthesised using an ST cell library of 0.12μm. Three Vdd settings were used during the experiment, $0.8V$, $1.0V$ and $1.2V$. In Table I and Table III we show the test-set sizes generated and the CPU time for the algorithm in Fig. 8 and the optional test set post-processing step respectively. Table I shows the results of running the Multi-Vdd Test Generation program (Fig. 8). The left-most column shows the benchmark circuits, where an initial "c" means that the circuit is combinational and an "s" means that the circuit is sequential. The second column from the left shows for each design, how many non-feedback bridges have been identified from the circuit layout. The "extractRC" tool from Cadence was used to get the pairs of nets that are capacitively coupled. These pairs of nets are the most likely bridge locations. Feedback bridges were identified and removed. The next three main columns, marked with "Vdd $0.8V$", "Vdd $1.0V$" and "Vdd $1.2V$", show the test pattern count in the corresponding test-set - TS 1, TS 2 and TS 3 respectively. The sixth column shows the total number of test-patterns necessary to achieve 100% defect coverage. The column that is marked with "CPU time" shows the CPU time required to achieve these results, in seconds. Our implementation uses a SAT solver based ATPG engine [15]. The second column from the right shows the fraction of time spent inside the ATPG engine (line 11 of Fig. 8). We believe this fraction can be significantly reduced if access to a more efficient commercial ATPG engine is available. The right-most column gives the average number of simulations per bridge that would have been required if we did not have a pre-compiled database of bridge simulation data.

In Table II we have evaluated the test-sets that are defined in Table I. Column two until five show the incremental defect coverage of applying the test-sets in order. This defect coverage is defined as follows: $DC = \sum_B$ detected resistance$/ \sum_B$ detectable resistance, where $\sum_B$ signifies the sum over all the RBFs. Column two is the defect coverage of only applying TS 1 (for 0.8V Vdd). Column three is the defect coverage achieved by applying TS 1 at 0.8V and TS 2 at 1.0V. In the same way, column four is the defect coverage of TS 1 and TS 3 (where TS 3 is applied at 1.2V). The last column shows the defect coverage achieved by applying all test-sets at their respective Vdd settings.

As can be seen from Table I, for some circuits, 100% defect coverage can be achieved using a single Vdd during test. However, for other circuits, such as s35932, achieving full defect coverage requires testing at more than one Vdd setting.

Table III shows the results of applying the optional post processing step to the test-sets in Table I. The columns that are marked with # show the final number of test patterns in the test set for the respective Vdd settings. The complimentary %-columns give the relative reduction in test patterns in the respective test sets. So for circuit s15850, the outcome of the post-processing was 235 test patterns for Vdd 0.8V, which is 27% less than before the post-processing (see Table I, column 3). In the fifth main column is the relative reduction in the total number of test patterns. The last column shows the relative difference in CPU time. So again, for circuit s15850, it took 4.58 times longer to include the post-processing compared to column 7 of Table I. Table III demonstrates that it is possible to achieve up to 27% reduction in test set size at the expense of increased CPU time.

An additional experiment was made using Synopsys TetraMAX and Multi-Vdd Test Generation (Fig. 8) as a combined test generation flow. First, a test-set targeting bridging faults is generated with TetraMAX, using the same bridge list as in the

TABLE III
TEST-SET SIZE REDUCTIONS USING THE POST-PROCESSING STEP ON THE TEST SETS FROM TABLE I

	0.8V		1.0V		1.2V		Tot.	CPU
Design								time
	#	%	#	%	#	%	%	%
c1355	32	18					18	95
c1908	47	18					18	238
c2670	57	15					15	393
c3540	151	18	6	0	1	0	17	419
c7552	229	19			1	0	18	528
s838	22	15	2	0			14	67
s1488	121	16	2	0			16	608
s5378	175	18					18	585
s9234	109	17	2	0			17	390
s13207	158	18	3	40	1	0	18	342
s15850	235	27	4	0	3	40	27	458
s35932	459	16	43	14	51	19	16	900

TABLE IV
RESULTS OF USING TETRAMAX AND MVTG AS A COMBINED TEST GENERATION FLOW

	TMAX		MVTG top-up				
	0.8V		0.8V	1.0V	1.2V	Tot.	CPU
Design	DC	#tp	#tp	#tp	#tp	#tp	time
c1355	83	33	32			65	18
c1908	98	42	27			69	11
c2670	90	27	50			77	36
c3540	96	72	126	6	1	205	239
c7552	95	44	198		1	243	789
s838	88	17	17	2		36	2
s1488	96	82	82	2		166	123
s5378	95	60	123			183	214
s9234	89	48	92	2		142	105
s13207	95	60	89	5	1	155	1625
s15850	98	56	144	4	5	209	1954
s35932	96	33	89	36	66	224	11511

experiment of Table I. Then the TetraMAX test-set was fault simulated at Vdd 0.8V (since higher resistive bridging fault coverage is achieved at a lower Vdd). The defect coverage achieved and the number of test patterns in the TetraMAX test-set is given in the second main column of Table IV. Subsequently, MVTG is used on the bridges that were not fully covered by the TetraMAX test-set, to supply the remaining defect coverage up to 100%. The sizes of the test sets generated by the MVTG top-up run are given in the third column for each Vdd setting. In the fourth column of Table IV, marked "Tot." we show the total test pattern count. The last column is the CPU time (in seconds) for simulating the TetraMAX test-set and running MVTG to top-up the test-set.

Please note in Table IV, that Synopsys TetraMAX generates test-sets that may yield defect coverage as low as 83% at Vdd = 0.8V. TetraMAX uses a heuristic that combines the victim-aggressor bridge fault model with a scheme to drive one of the bridged nodes with maximum strength and the other node with minimum strength. This way, the likelihood of detecting a bridge defect is maximised, even though TetraMAX does not take the defect resistance value into account.

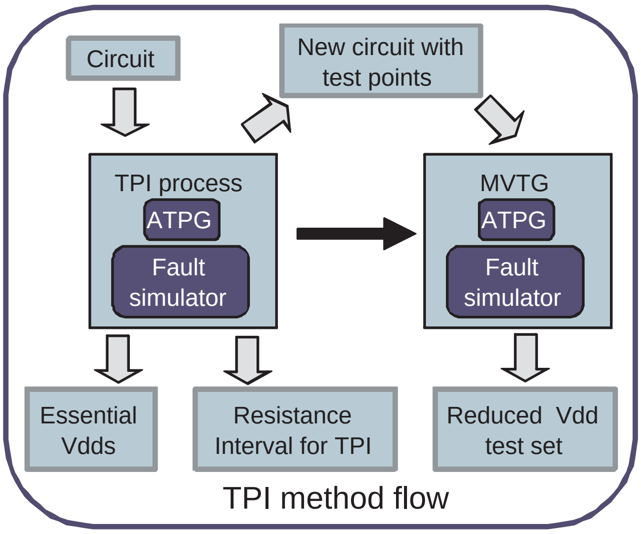

Fig. 9. Flows for the proposed TPI scheme

V. REDUCING THE NUMBER OF TEST VDDS

The flow for reducing the number of Vdd settings during test is shown in Fig. 9, it consists of two phases: test point identification and insertion, during which phase the method identifies the Vdd settings which can be eliminated during test, and test generation on the modified circuit.

A. Methodology

As shown in Section IV-A, different test sets need to be applied at several Vdd settings to ensure 100% defect coverage for multi-Vdd designs. Switching between supply voltage settings during test is not a trivial task and increases the cost of test, mainly due to the switching time overheads. Consequently, it is desirable to keep the number of the test voltage settings to a minimum. Previously, test point insertion (TPI) has been used for increasing the defect coverage [16] and test compaction [17]. In this section, we show how TPI can be used to reduce the number of different Vdd settings required during test without affecting the defect coverage. The proposed flow is shown in Fig. 9. For this purpose, we introduce the concept of ”essential” test Vdd. A test Vdd is said to be essential if there is at least one bridge for which the highest resistance value causing faulty behavior can be detected at this Vdd. This means that any of the resistance intervals targeted at non-essential test Vdds by the test generation algorithm presented in the previous section can be detected at one of

```
1: Compute set of Essential Test Vdds ($V_{ess}$)
2: Compute set of NRINEV
3: **for all** $NRINEV$ **do**
4:    LSC Selection(NRINEV, $V_{ess}$)
5:    Determine a preliminary set of test points at the defect
      site boundary for detecting the selected LSCs
6: **end for**
7: Minimize set of observation points
8: Control Point Minimization ()
9: Generate Essential Vdd Test Sets for the modified netlist
10: **return** $(netlist, Test\,Sets)$
```

Fig. 10. Test Point Insertion

the essential test Vdds, subject to suitable controllability and observability at the bridge site. Test point insertion can be used to provide the controllability and observability required at the bridge site. Fig. 10 outlines the method for achieving this goal. The key steps of this method are further detailed in Fig. 12, 13, 14 and 15.

The algorithm starts by computing the set of essential test Vdds for the given voltage settings and bridge list. To achieve this, for each bridge B, the algorithm determines the highest detectable bridge resistance value across all available Vdd settings and marks the Vdd setting corresponding to the highest resistance value as essential Vdd. In line 2, the algorithm determines for each bridge the set of resistance intervals which cause faulty behaviour at a non-essential Vdd, but are fully or partially undetectable at any of the essential Vdds due to lack of suitable controllability or observability. These resistance intervals are referred to as Non Redundant Interval at Non-Essential Voltage (NRINEV). Next, in lines 3 to 6, for each NRINEV, the algorithm determines a set of test points needed to make the resistance interval detectable at an essential Vdd setting. For this purpose, a set of LSC which fully cover the NRINEV interval is identified. Since in most cases, more than one set of LSCs can be used to cover the same NRINEV, the algorithm selects the LSC set which is likely to require the least number of test points to become detectable. The LSC selection algorithm used for this purpose is detailed in the following section. Once all NRINEV intervals have been covered, in lines 7 and 8 an attempt is made to reduce the number of required test points by identifying test points which can be shared among two or more selected LSCs. The algorithm then inserts the resulting set of test points into the original netlist and invokes MVTG (Fig. 8) to generate the test sets corresponding to the set of essential Vdds.

1) LSC selection: LSC selection aims to determine a set of LSC covering a given NRINEV which is likely to require the least number of test points. The algorithm, illustrated in Fig. 12, uses signal probabilities to quantify the effort required to control the logic values required by a LSC on the corresponding nets. In our experiments, signal probabilities were determined by simulating 5000 pseudorandom patterns, however other analytical methods for estimating signal probability could be used for this purpose just as well. The algorithm continues by identifying all LSCs which expose resistance intervals fully or partially overlapping with the target NRINEV interval. A probabilistic estimate of the controllability and observability (PECO) is computed for each candidate LSC (steps 3 to 5) as follows:

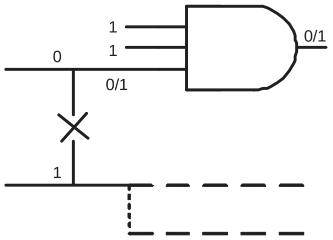

Fig. 11. Observability calculation

$$PECO(LSC) = C(LSC) \cdot O(LSC) \tag{1}$$

where C(LSC) is a probabilistic measure of the LSC controllability and O(LSC) is a probabilistic measure of the observability of the defect at the outputs of the gates fed by the bridge.

$$C(LSC) = \prod_{i=1}^{n}(Prob(i)) \tag{2}$$

where n is the cumulated number of inputs of the two gates driving the bridged nets and *Prob(i)* is the probability of logic value required by the LSC on input i.

$$O(LSC) = \sum_{i=1}^{m}(f(X)) \tag{3}$$

where m is the number of gates fed by the bridged nets which propagate the faulty value to their outputs and $f(X)$ is the probability that the fault effect is propagated through gate X, computed as follows:

$$f(X) = \frac{\sum_{j=1}^{k}\prod_{i=1}^{l} SP_{i,j}}{2^l} \tag{4}$$

where k is the number of input combinations which propagate the fault effect to the output of successor gate X, l is the number of inputs of gate X which are not fed by the bridge, and $SP_{i,j}$ is the probability of having the value corresponding to input combination j on input i. For example, for a 3-input AND gate fed by the bridge (as shown in Fig. 11) there is one input configuration which will propagate the fault (0/1) to its output out of the 4 possible combinations on the two inputs which are not fed by the bridge. Assuming the “1” probabilities of the inputs which are not driven by the bridge to be 0.4 and 0.7 respectively, the probability of this gate propagating

```
Input: NRINEV interval
    Essential Vdd settings $V_{ess}$
Output: Set of LSCs covering NRINEV with minimum number of required test points
    Compute signal probabilities on all nets
    Generate a list of LSC candidates sets which cover partially or completely the NRINEV at $V_{ess}$
    for all LSC candidates do
        Compute PECO(LSC)
    end for
    Determine the set of LSC covering NRINEV with maximum overall PECO
    return LSC selection
```

Fig. 12. LSC Selection

the fault to its output is $\frac{(0.4*0.7)}{4} = 0.07$. In this way O(LSC) provides a probabilistic estimate to help compare various LSCs and favour the one which is likely to require lesser number of observation points.

PECO(LSC) is then used as weight in a set covering linear programming formulation to determine the LSC set covering NRINEV which is likely to require the fewest number of test points.

At this point, the selected LSCs can be made controllable and observable by inserting appropriate test points at the defect site boundary.

2) Preliminary test point insertion at the defect site boundary: The method proposed for determining the preliminary set of test points at the defect site boundary for a given LSC is shown in Fig. 13. The algorithm starts by checking whether the driving gates' input assignments required by the LSC can be satisfied. If the required input assignments can be satisfied, it means there is at least one test pattern which activates the fault. Otherwise the algorithm attempts to determine a set of control points necessary for activating the fault (lines 2-15). This is achieved by using incremental bit-flipping on the driving gates' input assignments until a satisfiable combination is found. The input nets corresponding to the bit-flips in the LSC represent control point candidates and are added to the Exclusive Control Point Candidate list (ECL). At this point (step 17), the algorithm attempts to generate a test pattern which detects LSC and returns on successful generation of a test pattern. If a test pattern detecting the LSC could not be found, it means that although the fault can be activated, it is not observable at the primary outputs. At this point, the following two scenarios are possible: the faulty behavior can be observed at the output of at least one of the successor gates, or, the faulty behavior does not propagate through any of the successor gates. In order to differentiate between these two issues, a stimulus is generated for fault activation. This stimulus is applied to the circuit and all the successor gates are checked to see if the faulty behavior is observable at the output of any of these gates. If the fault is observable at the output of these gates, then the algorithm structurally traverses the circuit and marks all the nets that observes the faulty behavior as potential observation point candidates (step 22). If the fault effect is not observable at the output of any of the successor gates, the algorithm uses the logic values on all the nets, set by the stimulus generated in step 20 of the algorithm, and identifies the successor gate which observes the faulty value and requires the least number of control points in order to propagate it to its output. The nets corresponding to these control points are then added to ECL. In lines 28 to 34 the algorithm repeats steps 17 to 23 to mark all the nets that observe the faulty values for later observation point minimization, if a test pattern cannot detect the defect even after inserting control points for observability.

3) Test points minimization: The TPI algorithm (Fig. 10) minimizes the number of observation points, after processing all the NRINEV intervals. The optimum set of observation points will be the minimum set cover of the nets marked as observation point candidates in lines 22 and 33 of Fig. 13. This is similar to the method proposed in [18].

The TPI algorithm calls control point minimization algorithm in step 8 of Fig. 10, to reduce the number of control points in the modified circuit. This is achieved by finding pairs of control point candidate nets which can be replaced by a single control point while still achieving the required controllability. The algorithm (shown in Fig. 14) starts by determining the fan-in cone (FIC) sets for each net added to the ECL set in lines 9 and 26 of Fig. 13. FIC(ec) consists of all nets in the fan-in logic cone of ec, starting from the primary inputs. Basically, FIC(ec) contains all nets which may affect the logic value on ec. Next, the algorithm finds the Common Nets (CN) for the FIC of all possible pairs of nets in ECL, i.e., CN(ec_i, ec_j) holds the nets which appear in both FIC(ec_i) and FIC(ec_j). For every set of common nets CN(ec_i, ec_j), the algorithm attempts to determine a list of valid candidates (VC) shown in line 8, where every valid candidate is able to provide the required controllability on (ec_i and ec_j), thus reducing two control points to one. These valid candidates are generated by algorithm shown in Fig. 15 (Find Valid CP Candidates) for every pair of control points in ECL. The algorithm then determines the minimum set of control points as a minimum set cover for all VC sets. The resulting set of control points are then inserted in the netlist.

The algorithm shown in Fig. 15 starts by creating a copy of the netlist without any control points, but with the optimized observation points at their respective locations. For every pair of control point candidates (ec_A and ec_B) the algorithm inserts all control points necessary to detect LSC(A) (using information stored in ECL), with the exception of ec_A and ec_B, where LSC(A) is the LSC corresponding to ec_A. It then tries all the common nets CN(ec_A, ec_B), one-by-one and attempts to generate a stimulus using both types of control

```
Input: LSC Candidate
       Bridge $b$
for all Gates driving the bridge do
  if LSC input assignment not satisfiable then
    CPCount = 1;
    SATISFIED = FALSE;
    while NOT SATISFIED do
      for all LSCIA = LSC input assignment with CP-Count bit-flips do
        if LSCIA is satisfiable then
          SATISFIED = TRUE;
          add nets corresponding to bit-flips in LSCIA to ECL
          BREAK;
        end if
      end for
      CPCount = CPCount + 1
    end while
  end if
end for
if LSC non-redundant then
  return $(success)$
end if
Generate a stimulus to activate the fault
if Fault is observable at the output of the gates fed by the bridge then
  Mark all the nets which observe the fault effect as OP candidates
else
  Use the logic values set by the stimulus at the inputs of the gate
  Identify a gate, from all the gates which see a fault, that require min. no. of CPs to propagate the fault
  add control point candidates to ECL
end if
if LSC non-redundant then
  return $(success)$
end if
Generate a stimulus to activate the fault
if Fault is observable at the output of the gates fed by the bridge then
  Mark all the nets which observe the fault effect as OP candidates
end if
```

Fig. 13. Preliminary test point identification at the defect site boundary

```
for all NRINEV do
  for all ec $\in$ ECL do
    Compute FIC(ec)
  end for
end for
for all pair $(ec_i, ec_j)$ where $ec_i$, $ec_j \in$ ECL do
  CN$(ec_i, ec_j)$ = FIC$(ec_i)$ $\bigcap$ FIC$(ec_j)$
  VC$(ec_i, ec_j)$ = Find Valid CP Candidates (CN$(ec_i, ec_j)$)
end for
Find minimum number of CPs as a minimum set cover on $\{$VC$(ec_i, ec_j)\}$
Insert CPs into netlist
```

Fig. 14. Control Point Minimization

```
Input: $ec_A, ec_B$, CN$(ec_A, ec_B)$, LSC(A), LSC(B)
Create a copy of the original circuit
Insert all the CPs required by LSC(A) with the exception of $ec_A, ec_B$
for all cn $\in$ CN$(ec_A, ec_B)$ do
  for all cptype $\in$ CP-0, CP-1 do
    Insert a control point (cptype) at cn
    if LSC(A) is non-redundant then
      FVC = FVC $\bigcup$ $\{$cn$\}$
    end if
  end for
end for
if FVC $\neq \emptyset$ then
  Insert all the CPs required by LSC(B) with the exception of $ec_A, ec_B$
  for all fvc $\in$ FVC do
    Insert a control point of type cptype(fvc)
    if LSC(B) is non-redundant then
      VC = VC $\bigcup$ fvc
    end if
  end for
end if
return VC
```

Fig. 15. Find Valid CP Candidates

points CP-1 and CP-0. For all candidates that detect LSC(A) a tuple consisting of the net, fanout and CP-type is placed in *First Valid Candidates, FVC*. The algorithm then moves to LSC(B) and repeats the above procedure but this time it uses the members of *FVC* instead of common nets' members. It then adds all those members of *FVC* which are able to detect LSC(B) to *Valid Candidates, VC* list and returns the list to the calling Algorithm (Fig. 14).

B. Experimental Results

The TPI algorithm (Fig. 10) has been validated using a similar experimental set up as discussed in Section IV-B. The only difference is that instead of extracting the bridge list using

TABLE V
RESULTS OF TEST POINT INSERTION ALGORITHM

Design	Total Bridges	Vdd(s) bf TPI	Vdd(s) af TPI	CP(s)	OP(s)
c1355	6,566	0.8v, 1.2v	0.8v	6	0
c1908	7,986	All*	0.8v, 1.2v	2	1
c2670	10,000	All	0.8v, 1.2v	19	0
c3540	10,000	All	0.8v, 1.0v	6	1
c7552	9,998	0.8v, 1.2v	0.8v	0	1
s344	469	All	0.8v	5	0
s382	1,146	All	0.8v, 1.2v	7	2
s386	1,625	All	0.8v, 1.0v	9	1
s838	5,737	All	0.8v, 1.0v	26	11
s5378	9,933	All	0.8v, 1.0v	5	1
s9234	10,000	All	All	0	0
s13207	10,000	All	0.8v, 1.0v	3	0
s15850	10,000	All	0.8v, 1.0v	3	0

*All = 0.8v, 1.0v, 1.2v

a layout tool, in this case, an exhaustive bridge list is generated by considering all possible pairs of nets in the netlist, up to a maximum of 10,000 pairs. This is done to increase the total number of bridges, and therefore create more challenging test cases for all the circuits. The test point insertion flow on the layout extracted bridge list required only a very small number of test points, only 3 out of 12 circuits required test points. The experimental data is available at [19] to enable comparison with this work. The total number of bridges for each circuit is shown in the second column of Table V.

The total number of bridges considered for each circuit, along with the number of test Vdd(s) used for detecting all the defects both before and after inserting test points in the circuit, are shown in Table V. As it can be seen, by using the proposed test point insertion method, the number of test Vdds are reduced from three to one, or three to two for almost all circuits without affecting the defect coverage. It is only for s9234 where the number of Vdd settings required during test could not be reduced, this is because it has bridges with highest critical resistance at all three test voltages, i.e., they are all essential. The number of control and observation points added in each circuit are shown next, in Table V. It should be noted that total number of test points (including OPs and CPs) are ten or less for almost all the circuits, and that it is only in the cases of c2670 and s838 that more test points are used.

VI. CONCLUSION

Low power consumption and low cost manufacturing test are key constraints in today's competitive microelectronics industry. This paper has demonstrated that the employment of adaptive power management presents a number of challenges that need to be addressed to achieve high test quality at low cost. The paper has addressed these challenges through a multi-Vdd test generation method which delivers full bridging fault coverage across multiple Vdd settings and a test point insertion method which can be employed to reduce the number of Vdd settings required during test without affecting the test quality. Although only non-feedback bridges have been considered in this work, we believe the same concepts are equally applicable for feedback-bridging testing and we are planning to address this in our future work.

REFERENCES

[1] M. T. Schmitz, B. M. Al-Hashimi, and P. Eles, *System-level design techniques for energy-efficient embedded systems*. Kluwer Academic Publishers, 2004.

[2] S. M. Martin, K. Flautner, T. Mudge, and D. Blaauw, "Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, San Jose, CA, USA, Nov. 2002.

[3] "Intel: Pxa270 processor datasheet," Nov. 2007. [Online]. Available: http://www.phytec.com/pdf/datasheets/PXA270_DS.pdf

[4] M. Renovell, P. Huc, and Y. Bertrand, "Bridging fault coverage improvement by power supply control," in *Proceedings of the VLSI Test Symposium (VTS)*, Apr. 1996, pp. 338–343.

[5] Y. Liao and D. M. H. Walker, "Fault coverage analysis for physically-based CMOS bridging faults at different power supply voltages," in *Proceedings of the International Test Conference (ITC)*, Oct. 1996, pp. 767–775.

[6] V. R. Sar-Dessai and D. M. H. Walker, "Resistive bridge fault modeling, simulation and test generation," in *Proceedings of the International Test Conference (ITC)*, Atlantic City, NJ, USA, Sep. 1999, pp. 596–605.

[7] B. Kruseman, S. van den Oetelaar, and J. Ruis, "Comparison of Iddq testing and very-low voltage testing," in *Proceedings of the International Test Conference (ITC)*, Oct. 2002, pp. 964–973.

[8] P. Engelke, I. Polian, M. Renovell, B. Seshadri, and B. Becker, "The pros and cons of very-low-voltage testing: an analysis based on resistive bridging faults," in *Proceedings of the VLSI Test Symposium (VTS)*, Apr. 2004, pp. 171–178.

[9] P. Engelke, I. Polian, M. Renovell, and B. Becker, "Simulating resistive bridging and stuck-at faults," *IEEE Transactions on Computer-Aided Design*, vol. 25, pp. 2181–2192, Oct. 2006.

[10] M. Renovell, F. Azais, and Y. Bertrand, "Detection of defects using fault model oriented sequences," *Journal of Electronic Testing: Theory and Applications*, vol. 14, pp. 13–22, Feb. 1999.

[11] T. Maeda and K. Kinoshita, "Precise test generation for resistive bridging faults of CMOS combinational circuits," in *Proceedings IEEE International Test Conference (ITC)*, Oct. 2000, pp. 510–519.

[12] G. Chen, S. Reddy, I. Pomeranz, J. Rajski, P. Engelke, and B. Becker, "An unified fault model and test generation procedure for interconnect open and bridges," in *Proceedings IEEE European Test Symposium (ETS)*, May 2005, pp. 22–27.

[13] P. Engelke, I. Polian, M. Renovell, and B. Becker, "Automatic test pattern generation for resistive bridging faults," *Journal of Electronic Testing: Theory and Applications*, vol. 22, pp. 61–69, Feb. 2006.

[14] T. Shinogi, T. Kanbayashi, T. Yoshikawa, S. Tsuruoka, and T. Hayashi, "Faulty resistance sectioning technique for resistive bridging fault atpg systems," in *Proceedings of the Asian Test Symposium (ATS)*, Kyoto, Japan, Nov. 2001, pp. 76–81.

[15] "zChaff Boolean Satisfiability problem solver," Mar. 2007. [Online]. Available: http://www.princeton.edu/˜chaff/zchaff.html

[16] N. A. Touba and E. J. McCluskey, "Pseudo-random pattern testing of bridging faults," in *Proceedings IEEE ICCD*, Oct. 1997, pp. 54–60.

[17] M. Geuzebroek, J. T. Van der Linden, and A. J. Van de Goor, "Test point insertion for compact test sets," *Proceedings IEEE International Test Conference (ITC)*, pp. 292–301, Oct. 2000.

[18] N. A. Touba and E. J. McCluskey, "Test point insertion based on path tracing," in *Proceedings IEEE VLSI Test Symposium (VTS)*, Apr. 1996, pp. 2–8.

[19] "TPI: Experimental data," Nov. 2007. [Online]. Available: http://users.ecs.soton.ac.uk/ssk06r/data/TPI.tgz

Saqib Khursheed received B.E. degree in Computer Engineering from NED University, Karachi, Pakistan, in 2001 and M.Sc. degree in Computer Engineering, from King Fahd University (KFUPM), Dhahran, Saudi Arabia, in 2004. He served as a Lecturer in KFUPM from 2005 to 2007. He is currently working towards his Ph.D. degree in Electronics and Computer Science, in Southampton University, UK. His research interests include: SoC testing, test compaction and multi-objective optimization.

Urban Ingelsson (S'06) received the M.Sc. degree in Computer Science and Engineering in 2005 from Linköping University, Sweden. He is now with the Southampton University, working towards the Ph.D. degree in Electronics and Computer Science.

He spent eight months of 2004 in the Digital Design and Test group of Philips Research in Eindhoven, the Netherlands as a part of his M.Sc. thesis project. His research interests include testing of digital systems and low-power design.

Paul Rosinger received the B.Sc. in Computer Science from the Technical University of Timisoara, Romania, in 1999, and the Ph.D. in Electronics and Computer Science from the Southampton University, United Kingdom, in 2003.

He is currently working as a postdoctoral research-fellow at Southampton University. His current research interests include testing of digital systems, low power embedded systems and reconfigurable architectures.

Bashir M. Al-Hashimi (M'99-SM'01) received the B.Sc. degree (with 1st-class classification) in Electrical and Electronics Engineering from the University of Bath, UK, in 1984 and the Ph.D. degree from York University, UK, in 1989. Following this he worked in the microelectronics design industry and in 1999, he joined the School of Electronics and Computer Science, Southampton University, UK, where he is currently a Professor of Computer Engineering and Director of the Pervasive System Center. He has authored one book on SPICE simulation, (CRC Press, 1995), and coauthored two books, Power Constrained Testing of VLSI circuits (Springer, 2002), and System-Level Design Techniques for Energy-Efficient Embedded Systems (Springer, 2004). In 2006, he edited the book, System-on-Chip: Next Generation Electronics (IEE Press, 2006). He has published over 200 papers in journals and refereed conference proceedings. His current research interests include low-power system-level design, system-on-chip test, and reliable nano design.

Prof. Al-Hashimi is a Fellow of the IEE and a Senior Member of the IEEE. He is the Editor-in-Chief of the IEE Proceedings: Computers and Digital Techniques, an editor of the Journal of Electronic Testing: Theory and Applications (JETTA), and is a member of the editorial board of the Journal of Low Power Electronics, and the Journal of Embedded Computing. He was the General Chair of the 11th IEEE European Test Symposium (UK 2006) and he is the Technical-Programme Chair of DATE 09. He is the coauthor of the James Beausang Best Paper Award at the 2000 IEEE International Test Conference relating to low power BIST for RTL data paths, and a co-author of a paper on test data compression which has recently been selected for a Springer book featuring the most influential work over the 10 years of the Design Automation and Test in Europe (DATE) conference.

Peter Harrod (M'80-SM'99) graduated with a B.Sc(Eng) from the University of the Witwatersrand in 1976 and with M.Sc and Ph.D. degrees from the University of Manchester Institute of Science and Technology in 1978 and 1982 respectively.

He is a Consultant Engineer at ARM Ltd in Cambridge UK where he works on embedded CPU designs, chiefly in the areas of design for test and debug. Dr. Harrod is a Fellow of the IET and has served on several IEEE standards and conference program committees.