IEEE 13th Asian Test Symposium (ATS'04), pp. 32-39, Nov. 2004.

Power-Constrained DFT Algorithms for Non-Scan BIST-able RTL Data Paths

Zhiqiang You[1] Ken'ichi Yamaguchi[2] Michiko Inoue[1] Jacob Savir[3] Hideo Fujiwara[1]
[1]*Nara Institute of Science and Technology, Ikoma, Nara 630-0192, Japan*
{you-z, kounoe, fujiwara}@is.naist.jp
[2]*Nara National College of Technology, Yamato-Koriyama, Nara 639-1080, Japan*
yamaguti@info.nara-k.ac.jp
[3]*Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ*
savir@njit.edu

Abstract

This paper proposes two power-constrained test synthesis schemes and scheduling algorithms, under non-scan BIST, for RTL data paths. The first scheme uses boundary non-scan BIST, and can achieve a low hardware overhead. The second scheme uses generic non-scan BIST, and can offer some tradeoffs between hardware overhead, test application time and power dissipation. A designer can easily select an appropriate design parameter based on the desired tradeoff. Experimental results confirm good performance and practicality of our new approaches.

1. Introduction

A non-scan built-in self-test (BIST) is a promising approach that can realize at-speed test with a short application time. However, some BIST schemes already reported in the literature suffer from a high hardware overhead. Moreover, the excessive power consumption during these BIST schemes constitutes a considerable problem in some applications.

The techniques in [1-2] propose a test synthesis and scheduling algorithm under power constraints for BISTed register-transfer level (RTL) data paths. These proposed techniques use adjacent non-scan BIST [3], may exhibit a high hardware overhead due to the use of an excessive number of reconfigured registers.

This work was supported in part by Japan Society for the Promotion of Science (JSPS) under Grants-in-Aid for Scientific Research No. 14658092

Masuzawa et. al. [4] propose a BIST methodology for RTL data paths that uses a boundary non-scan BIST scheme. The approaches in [5,6] improve the method in [4] by introducing concurrent testing, exploiting time division between existing test pattern generators (TPGs), so that two different input ports of a module can share the same TPG. However, these previous works did not consider the problem of power consumption during test. Since these methods propagate test patterns, and test responses, simultaneously through modules in the data path, multiple modules dissipate power in order to test a single module. For some applications, this high power dissipation is unacceptable. Hence, we need to explore design for testability (DFT) schemes that limit this power consumption during test. In [3] TPGs and response analyzers (RAs) are placed not only at the chip boundary, but also inside the data path itself. We will continue to utilize this approach in this paper as well.

In this paper, we introduce two power-constrained DFT algorithms. The first focuses on achieving low hardware overhead (referred to in the paper as "problem 1"). The second algorithm explores possible trade-offs between hardware overhead, test application time, and power dissipation (referred to in this paper as "problem 2").

This paper is organized as follows. Section 2 introduces some basic concepts, such as the data path digraph, and outlines the problems to be solved. Section 3 addresses the power constraints for problem 1, and shows algorithms for performing the test and still meeting the given constraints. Section 4 addresses the same issues for problem 2. Section 5 reports on some experimental results using our proposed schemes. Section 6 concludes with a brief summary.

2. Preliminaries

2.1. The Data Path Digraph

A data path [4] consists of hardware elements and lines. Hardware elements, in this context, include primary inputs (PIs), primary outputs (POs), registers (Rs), multiplexers (MUXes), and functional modules (Ms) that have any number of input ports and one output port. Since the multiplexing function can be embedded within an M, we will use the term M in this wider sense of its capability (including multiplexing). Input patterns enter the circuit through the PIs, and exit through the POs. Input values enter into a hardware element through its input ports, and exit through its output port. Every input port of an M is reachable from some PIs, and every output port of an M is observable at some POs.

Similar to the definition in [4], we define a data path digraph $G=(V, A)$ as follows.

•$V=V_H \cup V_{IN} \cup V_{OUT}$, where

– V_H is the set of nodes that correspond to all hardware elements in the data path. Let $V_H = V_M \cup V_R \cup V_{OTH}$, where, V_M, V_R and V_{OTH} are the set of nodes which represent modules, registers and other hardware elements respectively.

– V_{IN} is the set of nodes which correspond to all input ports in the data path, and

– V_{OUT} is the set of nodes which correspond to all output ports in the data path.

• $A= A_1 \cup A_2 \cup A_3$, where

– $A_1=\{(x, y) \in V_{OUT} \times V_{IN} \mid$ output port x is connected to input port y by a line$\}$,

– $A_2=\{(y, u) \in V_{IN} \times V_H \mid y$ is an input port of $u\}$, and

– $A_3=\{(u, x) \in V_H \times V_{OUT} \mid x$ is an output port of $u\}$.

Notice that in the digraph, every PI and PO is mapped into a pair of nodes, and not a single node. For example, Fig. 1. shows a data path fragment with its associated digraph.

An input port $i_j \in V_{IN}$ is also one of the input ports of a node $u_M \in V_M$, if they are connected by an arc in A_2. We denote the arc emanating from node u_M by e_M, and the head node of the arc e by h_e. The sequential depth of a path is the number of register elements along the path.

2.2. Definitions

We define the following two concepts.

Definition 1: A data path is *boundary non-scan BIST-able* if each module M in the data path can be tested as follows.

There exists a TPG for each input port of M, and an RA (response analyzer) for the output port of M such that

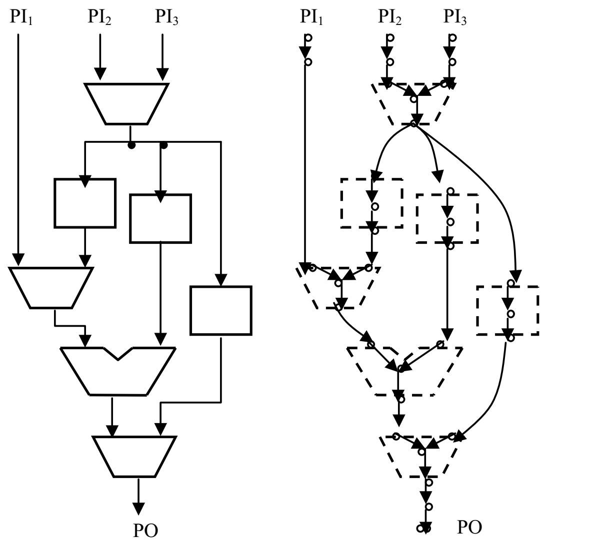


Figure 1. A data path and its associated digraph

(I-i). TPGs and RAs are placed only at PIs and POs respectively.

(I-ii). There are paths that propagate test patterns generated by the TPGs to the input ports of M, and test responses of M to the corresponding input ports of the RA, concurrently, without any conflict of control signals.

(I-iii). For any two input ports of any M, test patterns can either be propagated to these from two different TPGs, or from the same TPG, provided it has different sequential depths leading to these two ports. ■

Notice that we allow test patterns to be propagated through a module M using its *thru input function,* if such a function exists. Thus, a module with a thru input can be operated in a transparent mode to pass test patterns generated upstream to other components downstream.

In Definition 1, the control signals include select signals for MUXes; hold inputs for registers, and thru inputs for functional modules.

Definition 2: A data path is *non-scan BIST-able* if each module M in the data path can be tested as follows.

There exists a TPG for each input port of M, and an RA for the output port of M, such that properties (II-i), (I-ii), and (I-iii) in Definition 1&2 hold.

(II-i). TPGs and RAs can be placed at PIs and POs respectively, and any register inside the data path can be a candidate for augmentation into a TPG or an RA. ■

In boundary non-scan BIST, and non-scan BIST schemes, we categorize control paths that propagate test patterns from TPGs to inputs of a module under test. We distinguish, therefore, between the following cases:

Type 1: A control pattern can be chosen such that no two input ports of M share a TPG.

Type 2: Some input ports share a TPG with paths of different sequential depths.

Type 3: Some input ports share a TPG, and the control path for one of its input port passes through another input and output ports of this same module (See Fig. 2).

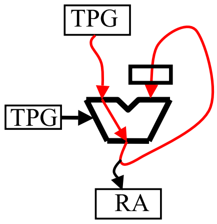


Figure 2. A module with a type 3 path

An observation path propagates test responses from a module output to an RA. In the sequel, we will refer to both control paths and observation paths simply as test paths.

2.3. Problem Description

Two problems have been formulated in [3] and are repeated here. Let $f_H(h,t)$ be a hardware-intensive cost function, such that $f_H(h_1,t_1) < f_H(h_2,t_2)$ if $h_1 < h_2$ or ($h_1 = h_2$ and $t_1 < t_2$). The "hardware" argument (h) reflects hardware overhead (HOH), and the "time" argument of the function (t) reflects test application time (TAT).

Problem 1: Minimize the hardware overhead of a given data path under a boundary non-scan BIST, and a test scheduling algorithm, subject to a given power constraints. Stating it more formally,

Given:
- **Input:** a data path and peak power dissipation limit P_{max}.

Task:
- **Output:** a boundary non-scan BIST-able data path, and a test schedule, satisfying P_{max}, that achieves:
- **Objective:** minimizing f_H(HOH,TAT), i.e. minimize hardware overhead. ■

To achieving this task we are allowed to add DFT elements, such as linear feedback shift registers (LFSRs), multiple-input signature registers (MISRs), test MUXes (T_MUXes), hold functions for registers, and thru-functions for functional modules.

Problem 2: Given a design parameter α, design a non-scan BIST-able data path, and a test scheduling algorithm, under a given power constraints. More formally,

Given:
- **Input:** a data path, co-optimization ratio α ($0 \le \alpha \le 1$), and a peak power dissipation limit P_{max}.

Task:
- **Output:** a non-scan BIST-able data path, and a test-schedule, satisfying P_{max}, that achieves:
- **Objective:** minimize

$$\alpha \cdot HOH + k(1-\alpha) \cdot TAT$$ [1] ■

In order to achieve this task, we are allowed to add DFT elements, such as Built-In Logic-Block Observations (BILBOs) [7], concurrent BILBOs (CBILBOs) [8], LFSRs, MISRs, T_MUXs, hold functions for registers, and thru-functions for functional modules.

3. Power Constrained DFT Algorithm for Problem 1

3.1. Algorithm Description

This algorithm consists of the following three phases.

Phase 1. Convert the given data path to a boundary non-scan BIST-able one utilizing the following steps:
1. Eliminate *critical arcs* for modules.
2. Add thru-functions for functional modules M whenever necessary.

Phase 2. Determine the test paths for each module. If the power constraint is violated, consider adding minimum number of T_MUX to bypass some paths to reduce power. Determine the test paths again until the modules can be tested one by one, while satisfying the power constraint.

Phase 3. Schedule the test.

3.2. Critical Arc Elimination

The real estate area of a T_MUX is usually higher than that of a module-embedded thru-function. There are, however, instances where only T_MUXes can be used to establish testability. The instances where T_MUX are necessary to enforce testability are their need in eliminating critical arcs (to be introduced in this section). We, therefore, consider adding a minimum number of T_MUXes into the data path only when it is necessary.

Definition 3: For a data path digraph G and an arc e, let G_e be a digraph obtained from G by deleting e. An arc e is critical for a node $u_M \in V_M$ if one of the following three cases holds (for the sake of simplicity we state the conditions for modules with two ports only):

---

[1] k is a unit conversion constant with value $|k| = 1$.

Case 1: None of the input ports of u_M is reachable from any PI in G_e, and the sequential depth of any path from h_e to the two ports is identical.

Case 2: None of the input ports of u_M is reachable from any PI in G_e; the sequential depths of any path from he to the two ports are different, and no PO is reachable from u_M in G_e.

Case 3: Let u_{M_1}, u_{M_2} and u_{M_3}, be members of V_M, and let e_{M_1}, and e_{M_2} be the outgoing arcs from nodes u_{M_1}, and u_{M_2} respectively. Arcs e_{M_1}, and e_{M_2} are critical for u_{M_3} if no PO is reachable from u_{M_3} in $G_{e_{M_1}}$ or $G_{e_{M_2}}$, and input port i_j, of u_{M_3} is unreachable from any PI in $G_{e_{M_j}}$, for $j=1,2$, respectively. ■

If e is a critical arc of u_M, we say u_M is *dominated* by e.

Theorem 1: If all modules have thru-functions for their input ports, a data path is boundary non-scan BIST-able if and only if (iff) there does not exist a critical arc in its associated digraph. ■

If more than one module are dominated by a critical arc, the order by which we handle these modules plays a key role in reducing the overall hardware overhead. To determine this order, we introduce notions that reflect the relationship between two dominated modules, called a *down-stream module* (DSM), and an *up-stream module* (USM).

For a dominated node $u_M \in V_M$ of a data path digraph G, let $E(u_M)$ be the set of critical arcs of u_M.

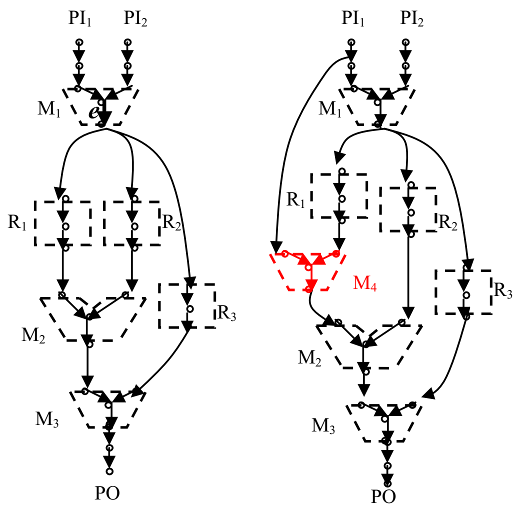
Definition 4: For two dominated nodes u_M and u_M', we say that u_M is the *up-stream module* (USM), iff u_M is a predecessor of u_M' in the digraph G', such that $G'.V=G.V$, $G'.A=G.A-E(u_M)-E(u'_M)$, or conversely, we say that u_M' is the *down-stream module* (DSM) iff u_M' is a successor of u_M in the G' digraph, provided the dominating critical arc does not meet the condition stated in case 1 of definition 3. ■

From the above definition, the following theorem follows.

Theorem 2: If M is the USM of M′, the critical arcs of both M and M′ can be eliminated by introducing a T_MUX to add a path from one PI to some other input port of M. Similarly, if M′ is a DSM of M, the critical arcs of both M and M′ can be eliminated by introducing a T_MUX to add a path from the output port of M′ to some PO. ■

Fig 3. illustrates how to eliminate a critical arc. From Definition 1, and the original data path digraph (Fig 3(a)), we find that both modules, M_2 and M_3, have one critical arc *e,* which is denoted by a boldface line in Fig. 3(a). M_2 is the predecessor of M_3, in other words, M_2 is the USM of M_3. Therefore, according to Theorem 2, addition of a T_MUX (M_4, in Fig. 3(b)) to establish a path from PI1 to one input port of M_2, eliminates the critical arc e for both modules. The data path digraph after adding the T_MUX for e is shown in Fig. 3(b).

The problem of adding a minimum number of T_MUXes to eliminate critical arcs is equivalent to the minimum prime-implicant covering problem, which is known to be NP-hard. We, therefore, use a heuristic algorithm to determine the selection of the dominated modules that will be used to add the extra paths to.



(a). Before adding a T_MUX **(b). After adding a T_MUX for *e***

Figure 3. Example of adding a T_MUX to eliminate a critical arc

3.3. Thru-Function Addition

After adding the necessary T_MUXes, we consider adding a minimum number of thru-functions, whose hardware overhead is usually lower than that of a T_MUX, in order to achieve boundary non-scan BIST-ability.

Theorem 3: If there exists a module M*,* that is an immediate successor of another functional module M′, then an addition of a thru-function to M′ is needed to test M. ■

After adding the necessary thru-functions, it may still not ensure that the data path is boundary non-scan BIST-able. We may need to add more thru-functions. In Fig. 4 there is no critical arc. However, a thru function from Q to PO needs to be added in order to facilitate vector propagation through module M_2.

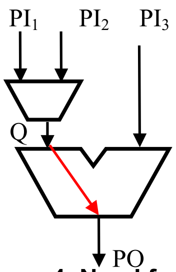


Figure 4. Need for adding a thru-function

3.4. Control Paths and Observation Paths Determination

After the thru-function addition, the data path is boundary non-scan BIST-able. We now determine the control paths and observation path for each module using the shortest, power-weighted, path.

3.5. Bypassing Overly Power Consuming Paths

In a boundary non-scan BIST scheme, TPGs and RAs are placed only at PI and PO sites respectively. Therefore, some modules may end up having long test paths, thus dissipating an extended amount of power. If some modules have long test paths, which dissipate more power than P_{max}, we try to bypass some of them by inserting T_MUXes. In this case, if two or more modules share a portion of their test paths (sub-paths), these modules might be able to share the added bypass as well. In this stage, we search for a minimum number of common sub-paths, so that when being bypassed, the underlying modules satisfy the given power constraints. This problem is also equivalent to the minimum prime-implicant covering problem. We, therefore, use a heuristical algorithm, where we always select the common sub-path such that, if bypassed, it reduces the maximum sum-of-powers for modules involved. Finally, we add the needed T_MUXes to bypass these sub-paths so identified.

3.6. Test Scheduling

We proceed to obtain the test incompatibility graph defined similarly to that given in [9].

Definition 5: Two modules M_1 and M_2 are test incompatible, if one of the following conditions holds.

i. The observation path of M_1 is joined with the test path of M_2.
ii. The control section associated with M_1 is of type 3, and joins the test path of M_2. ■

Since modules can share TPGs and parts of control paths, the power dissipated in these LFSRs and parts of these control paths, need not be accounted for, when considering all modules under test. We, therefore, introduce the following concept.

Definition 6: *Essential power dissipation* is:

i the power consumed by the module itself and its associated observation path, if the test path of the module is either of type 1 or of type 2.

ii the power dissipated in the tested module, its associated observation path, and its feed-around portion of the control path, if the test path of the module is of type 3. ■

For example, the hardware elements on the bold lines of Fig 5 (line feeding the RA and the feedback line) dissipate essential power for the module and its type 3 path.

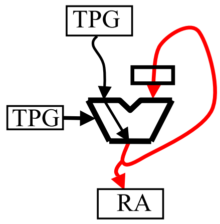


Figure 5. Essential power for a module with type 3 path

After bypassing the overly power-consuming sub-paths, we create the incompatibility graph. In this graph, the nodes are the tested modules, and edges only exist between incompatible modules. We extend the scheduling algorithm from [10] for concurrent testing of multiple modules. In [10] the power is evaluated as the sum of the powers consumed by the individual modules. In our extended algorithm, presented here, two important features come to light:

a. By sharing control paths of different tested modules, we decrease the total consumed power.
b. If it so happens that two modules activate secondary paths off their main test paths, and the paths reach different ports of the same MUX, and since we cannot stop the activity at the MUX, the total power consumed is larger than the sum of the powers of their individual stand-alone paths.

The approach in [10] schedules modules based on the "necessary" power dissipation. Here we consider “unnecessary” power dissipation, as well as essential power dissipation.

4. Power Constrained DFT Algorithm (Tabu Search-Based) for Problem 2

```
Algorithm: Power constrained test synthesis
and scheduling algorithm for Problem 2
(PCTSP2)
1.  Generate an initial solution;
2.  S_current ← S_init;
3.  repeat{
4.     for every register and functional module{
5.        for every possible move that is not in tabu list{
6.           Obtain data-path D_i
7.           if D_i is non-scan BIST-able{
8.              Schedule the test, get S_i;
9.           If Power constraints met -Compute
             TAT (T_i) and HOH (H_i);
10.          }
11.    }
12. }
13.  Find S_k for which
     α·H_i + (1−α)·T_i
     is minimum;
14. S_current ← S_k;
15.  Record the move into tabu list;
16. If this solution is the best so far, then
17.  set S_best ← S_k;
18. }
19. until #iterations>Min{N_itr1 , N_itr1}
```

Figure 6. PCTSP2 algorithm

Fig. 6. summarizes the tabu search-based algorithm [11]. Line 1 starts with an initial solution, taken as the solution for Problem 1. Lines 3-19 are the heart of the optimization process. For every register and functional module, we try every possible move[2], which is not in the tabu list (lines 4-5). After a move, if the data path D_i is non-scan BIST-able, proceed to schedule the test (S_i). If it meets the power constraints, compute the test application time (T_i), and hardware overhead (H_i), (lines 6-9). Here, we treat the internal test registers as either PIs or POs, depending on whether they are used to generate values, or capture responses. We, then, search for a solution[3] S_k that minimizes the value of the cost function $\alpha \cdot H_i + (1-\alpha) \cdot T_i$, and set $S_{current}=S_k$. This move is then recorded in the tabu list (line 15). If this solution turns out to be the best one so far, we set $S_{best}= S_k$. The algorithm ends when either the maximum number of iterations is reached (N_{itr1}), or the maximum number of iterations since the last obtained best solution exceeds some predetermined value (N_{itr2}).

5. Experimental Results

We have conducted experiments on the data paths of LWF, Paulin, Tseng, and JWF. Table 1 shows the characteristics of these data paths. Columns #PI, #PO, #Reg, #MUX, #M, denote the number of PIs, POs, registers, MUXes and functional modules, respectively. Columns “Bit” and “Area” denote bit-width, and the equivalent area as synthesized and reported by the Synopsys Design Compiler.

Table 1. Circuits characteristics

Circuit	#PI	#PO	Bit	#Reg	#MUX	#M	Area
LWF	2	2	32	5	5	3	6714
Paulin	2	2	32	7	11	4	36114
Tseng	3	2	32	6	7	7	23234
JWF	5	5	32	14	25	3	20373

We first treat modules of type 1 test paths. Let T_M be the test application time for a MUX, $T_M= T_u$, where T_u is an integer unit. We assume that the test application time of an adder (T_+), subtractor (T_-), multiplier (T_*), constant-input multiplier ($T_{*'}$), AND gate ($T_\&$), and OR gate ($T_|$) are $T_+=T_-=5T_u$, $T_*=20T_u$, $T_{*'}=3T_u$ and $T_\&=T_|=4T_u$, respectively. The test application time of a module with test path of either type 2, or type 3, are assumed to be $T_{type\ 2}=1.5T_{type\ 1}$, and $T_{type\ 3}=2T_{type\ 1}$, respectively. Let P_u be a standard unit of power. Using the technique in [12], we further assume that the power dissipations for MUX (P_M), AND gate ($P_\&$), OR gate ($P_|$), register (P_{Reg}), adder (P_+), subtractor (P_-), multiplier (P_*), constant-input multiplier ($P_{*'}$), BILBO (P_{BIL}), and CBILBO (P_{CBIL}), are $P_M=P_\&=P_|=P_u$, $P_{Reg}=P_+=P_-=5P_u$, $P_*=20P_u$, $P_{*'}=P_{BIL}=P_{CBIL}=10P_u$, respectively. The hardware overhead, in our experiments, has been determined from the Synopsys Design Compiler for DFT elements.

Tables 2-5 display the experimental results of the Power-Constrained Test Synthesis and Scheduling algorithm for Problem 1 (PCTSP1), Problem 2 (PCTSP2),

[2] A move is a general term for adding/removing thru functions in a module; reconfiguring a register into a BILBO, or CBILBO, adding a hold function to a register, or removing of some previously added hardware.

[3] A solution is a complete test scheduling with established values for TAT, HOH, and the resulting power.

Proceedings of the 13th Asian Test Symposium (ATS 2004)
0-7695-2235-1/04 $20.00 © 2004 IEEE

and the power-driven optimization TCSC (PTCSC) methods. TCSC is our previous methodology [6]. We have extended it here mainly in order to save power by assigning fixed values to unused control signals. Columns α, P_{max}, Pow, HOH and TAT are the co-optimization ratio, peak power dissipation limit, actual peak power dissipation, hardware overhead, and test application time, respectively. Notice that for a fixed P_{max},, the hardware overhead decreases with the increase of α. By the same token, the test application time increases with the increase of α. There is, therefore, a tradeoff between HOH and TAT. Notice that when P_{max} is increasing, the hardware overhead and test application time are both decreasing due to a potentially higher test activity. If we relax the peak power dissipation limit, we can use this relaxation in power to schedule more modules in a given test session, or, equivalently may need less hardware to test the modules in a given test session.

Table 2. Experimental results for the data path of LWF

Method	α	P_{max} (P_u)	Pow (P_u)	HOH (%)	TAT (T_u)
PCTSP2	0	60	59	32.4	15.5
		65	65	33.4	12
		70	65	33.4	12
	0.5	60	58	14.3	23.5
		65	58	12.4	23.5
		70	68	9.1	23.5
	1	60	58	14.3	23.5
		65	58	12.4	23.5
		70	68	9.1	23.5
PCTSP1		60	60	21.0	22.5
		65	64	15.7	24
		70	68	9.1	23.5
PTCSC		-	69	14.3	15

Table 3. Experimental results for the data path of Tseng

Method	α	P_{max} (P_u)	Pow (P_u)	HOH (%)	TAT (T_u)
PCTSP2	0	72	70	27.1	65.5
		82	82	29.5	44
		92	92	25.1	41
	0.5	72	70	15.4	78
		82	81	9.6	65
		92	92	8.7	59
	1	72	70	12.1	78
		82	81	9.6	65
		92	86	7.3	93.5
PCTSP1		72	72	12.1	76.5
		82	81	10.2	65
		92	92	9.3	59
PTCSC		-	77	11.8	103

In Table 4, for the case of α=1 and P_{max}=60, notice that PCTSP2 enjoys lesser hardware overhead than PCTSP1. This is because in the non-scan BIST scheme we can add more kinds of DFT elements that will make the approach more hardware-efficient. For cases other than α=1, the results are pretty much the same.

In Tables 2-4, when P_{max} is large enough, the hardware overheads of PCTSP1 and PCTSP2 (for α=1) are lower than that of PTCSC. This shows that our methodology is more efficient, even when there are no power constraints.

Table 4. Experimental results for the data path of Paulin

Method	α	P_{max} (P_u)	Pow (P_u)	HOH (%)	TAT (T_u)
PCTSP2	0	60	60	25.3	53.5
		100	99	25.1	31
		140	137	19.8	28
	0.5	60	58	7.0	72.5
		100	87	5.8	61.5
		140	114	3.1	71.5
	1	60	60	6.4	91.5
		100	100	4.9	91.5
		140	114	3.1	71.5
PCTSP1		60	58	7.9	89
		100	99	4.9	91.5
		140	114	3.1	71.5
PTCSC		-	112	3.4	82

Table 5. Experimental results for the data path of JWF

Method	α	P_{max} (P_u)	Pow (P_u)	HOH (%)	TAT (T_u)
PCTSP2	0	70	70	25.9	20
		100	95	30.2	20
		130	122	33.0	19
	0.5	70	70	4.2	41
		100	80	3.1	41
		130	80	3.1	41
	1	70	70	4.2	41
		100	80	3.1	41
		130	80	3.1	41
PCTSP1		70	70	4.2	50
		100	80	3.1	41
		130	80	3.1	41

In Table 6 TM stands for T_MUX; T stands for "thru functions", H stands for "hold-functions"; B stands for BILBO, and C stands for CBILBO. For this table, in the case of α=1 (biased towards saving on hardware), more T_MUXs and thru-functions were added to share TPGs, and hence to reduce the hardware overhead. On the other hand, for the case of α=0 (biased in favor of saving TAT), more BILBOs and CBILBOs were added to achieve a short test application time. To summarize, notice that CBILBOs are efficient in reducing TAT, while T_MUXes and thru-functions are efficient in achieving a low HOH.

Table 6. Added DFT elements and their overhead figures for the data path of Paulin

Method	α	P_{MAX} (P_u)	HOH	#TM	#T	#H	#B	#C
PCTSP2	0	60	25.3	0	8	0	1	6
		100	25.1	0	8	0	1	6
		140	19.8	0	7	0	2	4
	0.5	60	7.0	3	7	1	1	0
		100	5.8	2	7	0	1	0
		140	3.1	1	7	0	0	0
	1	60	6.4	3	7	0	1	0
		100	4.9	4	6	0	0	0
		140	3.1	1	6	0	0	0
PCTSP1		60	7.9	9	6	0	0	0
		100	4.9	4	6	0	0	0
		140	3.1	1	6	0	0	0

6. Conclusions

This paper proposed two power constrained DFT algorithms for two non-scan BIST schemes for RTL data-paths. The first proposed algorithm is for a boundary non-scan BIST scheme. Experimental results have shown that this method is efficient in achieving a low hardware overhead. The second algorithm is for a generic non-scan BIST scheme. We use a Tabu search algorithm to explore the solution space. Experimental results presented here show that there is a tradeoff between hardware overhead, test application time, and power dissipation. A chip designer may utilize these tradeoffs to prioritize one such parameter over the rest.

Acknowledgement

The authors wish to thank Drs. Satoshi Ohtake and Tomokazu Yoneda, of the Nara institute of Science and Technology, for their valuable comments. Thanks are also due to Mr. Tsuyoshi Iwagaki, and other students, from the laboratory of Prof. Fujiwara.

References

[1] N. Nicolici and B. M. Al-Hashimi, Power-Constrained Testing of VLSI Circuits. *Kluwer Academic Publishers*, Boston, MA, 2003.

[2] N. Nicolici and B. Al-Hashimi, "Power Conscious Test Synthesis and Scheduling for BIST RTL Data Paths," in *Proc. Int'l Test Conf. (ITC'2000)*, pp. 662-671, 2000.

[3] Z. You, M. Inoue and H. Fujiwara, "On the Non-Scan BIST Schemes under Power Constraints for RTL Data Paths", Digest of papers *4rd Int'l Workshop on RTL and High Level Testing. (WRTLT'2003),* pp. 14-21, 2003.

[4] T. Masuzawa, M. Izutsu, H. Wada and H. Fujiwara, "Single-Control Testability of RTL Data Paths for BIST," in *Proc. 9th Asian Test Symposium. (ATS'2000)*, pp. 210-215, 2000.

[5] K. Yamaguchi, H. Wada, T. Masuzawa and H. Fujiwara, "A BIST Method Based on Concurrent Single-Control Testability of RTL Data Paths," in *Proc. 10th Asian Test Symposium. (ATS'2001)*, pp. 313-318, 2001.

[6] K. Yamaguchi, M. Inoue and H. Fujiwara, "Hierarchical BIST: Test-Per-Clock BIST with Low Overhead," Digest of papers *3rd Int'l Workshop on RTL and High Level Testing. (WRTLT'2002),* pp. 42-47, 2002.

[7] B. Koenemann, J. Mucha and G. Zwiehoff, "Built-in logic block observation techniques," in *Proc. Int'l. Test Conf. (ITC'79)*, pp. 37-41, October, 1979.

[8] L.T. Wang and E.J. McCluskey, "Concurrent Built-in logic block observer (CBILBO)," in *Proc. The Int. symp. On Circuits and systems*, pp. 1054-1057, 1986.

[9] R. Chou K. Saluja, and V. Agrawal, "Scheduling Tests for VLSI Systems Under Power Constraints," *IEEE Trans on VLSI systems*, Vol. 5(2), .pp. 175-185, June 1997.

[10] V. Muresan, X. Wang, V. Muresan and M. Vladutiu, "The left Edge Algorithm and the Tree Growing Technique in Block-Test Scheduling under Power Constraints," in *Proc. of the 18th IEEE VLSI Test Symposium (VTS'2000)*, pp. 417-422, 2000.

[11] F. Glover and M. Laguna. *Tabu search*, In C.R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, pp. 70–150. McGraw-Hill Book Company, 1995.

[12] E. Macii, M. Pedram and F. Somenzi, "High Level Power Modeling, Estimation, and Optimization," *IEEE Trans. on Computers*, vol. 44, pp. 234-247, Feb. 1995.