Lower Bounds on Test Resources for Scheduled Data Flow Graphs*

Ishwar Parulkar, Sandeep K. Gupta and Melvin A. Breuer
Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-2562.

Abstract—**Lower bound estimations of resources at various stages of high-level synthesis are essential to guide synthesis algorithms towards optimal solutions. In this paper we present lower bounds on the number of *test* resources (i.e. test pattern generators, signature analyzers and CBILBO registers) required to test a synthesized data path using built-in self-test (BIST). The estimations are performed on scheduled data flow graphs and provide a practical way of selecting or modifying module assignments and schedules such that the resulting synthesized data path requires a small number of test resources to test itself.**

I. Introduction

Estimation of data path resources during high-level synthesis enables a designer to evaluate certain aspects of a design by comparing the estimates with specified constraints. By providing quick feedback for any design decision, estimates aid the designer in exploring a number of design alternatives instead of synthesizing a complete implementation and then measuring the quality of each design. Lower bounds on resources not only greatly reduce the size of the solution space but also provide a means to measure the proximity of the final solution to the optimal one.

There is some recent work for estimating lower bounds on *functional resources* such as adders, multipliers and registers [1],[2],[3],[4]. For making a synthesized design testable using a built-in self-test (BIST) strategy, some of the registers in the design have to be modified to operate as test pattern generators (TPGs), signature analyzers (SAs), built-in logic block observers (BILBOs) or concurrent BILBOs (CBILBOs) during the test mode. One consideration in selecting a BIST strategy is the extra area needed for these test resources. A number of high- level synthesis approaches that incorporate BIST have been investigated in the recent past [5],[6],[7],[8]. In these approaches, cost functions and heuristics are used to guide allocation algorithms toward low BIST area overhead designs. None of these approaches can estimate the effect of a decision on the final number of test resources required.

In this paper we derive lower bounds on test resources for BIST. We believe this is the first work on estimating lower bounds on *test resources* for self-testable data paths in high-level synthesis. The lower bounds are tight in the sense that they can be achieved if there are no functional resource constraints. The bounds can be used as an estimate to determine the quality of a schedule and module assignment in terms of BIST area overhead of the synthesized design. The lower bound estimation technique can be used on partial schedules and module assignments to guide them towards optimal testable solutions. The bounds also serve as an independent measure for comparing the quality of different high-level synthesis systems and algorithms that perform testability optimization.

The remainder of the paper is organized as follows. In Section II we present the testability model and some basic definitions. Section III deals with the derivation of lower bounds, their computational complexity and their use in high-level synthesis. Section IV describes data relating to lower bounds on the test resources of some high-level synthesis benchmark circuits.

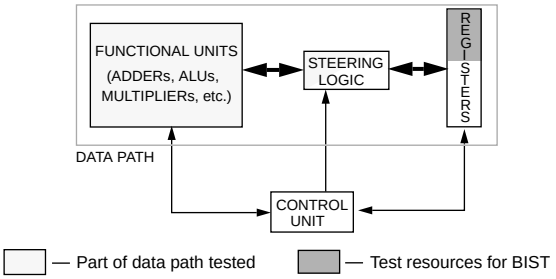

Fig. 1. Test methodology

II. Preliminaries

A. Test methodology

The high-level synthesis process assumed in this work is directed towards synthesizing data paths that are to be tested using a partial intrusion pseudo-random BIST methodology. A structural model of register-transfer level (RTL) designs synthesized by high-level synthesis systems is shown in Fig. 1. In partial intrusion BIST, a subset of registers are used in the test mode to test all the functional modules in the data path as depicted in Fig. 1. In the test mode, some of the registers in the data path are reconfigured to support test pattern generation (TPG), and some to support signature analysis (SA). By appropriate selection of test resources (TPGs and SAs) all the functional modules in the design are tested and in the process, some of the interconnections (multiplexer paths and wires) are also tested. The rest of the circuitry is assumed to be tested using functional tests. Note that in this partial-intrusion BIST methodology the test resources and paths used to generate, transport and collect test data are a subset of the functional data path. No additional data path components such as reg-

*This work was supported by the Advanced Research Projects Agency and monitored by the Department of the Army, Ft.Huachuca, under Contract No. DABT63-95-C-0042. The information reported here does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

DAC 96 - 06/96 Las Vegas, NV, USA ©1996 ACM, Inc. 0-89791-833-9/96/0006..$3.50

isters, multiplexers or interconnect are added for the purpose of testing.

For testing all the functional modules in the design using a small number of test resources, different mappings of registers to TPGs and SAs need to be considered. With respect to a module, a mapping of registers to TPGs and SAs that can be used to test the module is called a BIST **embedding** for that module. Embeddings for modules could be chosen such that a register that is a TPG for a module is an SA for a different module. In this case the register acts as a TPG and SA at different times and has to be modified to a BILBO register. The BILBO register has a higher overhead than a TPG or a SA. If the embedding chosen is such that a register is a TPG and an SA for the *same* module then that register has to act as a TPG and SA at the same time. To ensure high fault-coverage a concurrent built-in logic block observation (CBILBO) register is required [9].

Choosing embeddings that maximize the usage of registers as test resources between modules and minimize CBILBO scenarios results in a low BIST area overhead [8]. Functional constraints determine the sharing of modules (portion of data path to tested) by operations and the sharing of registers (data path components to be used as test resources) by variables. The functional constraints thus impose lower bounds on the number of test resources required to test all the modules. If additional registers and/or interconnections (not used in the functional mode) were to be used to test a circuit, only two TPGs and one SA would be required, assuming binary operations. But since test resources are selected from the set of functional registers this is not the case. Another point to note is that the objective of our testability optimization is BIST *area* overhead. Hence test concurrency (number of modules that can be tested simultaneously) does not depend on functional concurrency but rather on the sharing of test resources between modules.

B. Notation and definitions

The behavioral description is assumed to be given in the form of (1) a data flow graph (DFG) $G = (V, E)$, where V is the set of operations and E is the set of variables (operands and results of the operations), and (2) a schedule $S : V \rightarrow \{1, 2, 3, ...\}$, where $S(v)$ corresponds to the control step in which operation v is scheduled. Single operation per clock cycle is assumed and the synthesized data path is non-pipelined. All operators are assumed to be binary and commutative. Non-commutative operators can be handled by adding additional constraints. Unary operators can be treated as a special case of binary operators. The module assignment is defined as $\Pi_M : V \rightarrow M$, where M is the set of available modules. The subset of V mapped onto module M_i will be referred to as V_i. Each operation $v \in V_i$ will be referred to as an **instance** of M_i. Π_M can be viewed as a partition $\{M_1, M_2, ..., M_m\}$ of the set of operations V into m modules.

Definition 1: The **temporal multiplicity** of module M_i, $TM(M_i)$ is the number of operations from V mapped onto M_i, i.e. $TM(M_i) = \mid V_i \mid$.

Consider the scheduled DFG shown in Fig. 2 and the following module assignment. Operations $+_1$ and $+_2$ are assigned to M_1 and operations $*_1$ and $*_2$ are assigned to M_2. Thus $V_1 = \{+_1, +_2\}$ where each element is an instance of M_1 and $TM(M_1) = 2$.

Definition 2: The **input variable set** of module M_i, denoted by I_{M_i}, is the set of all the operand variables associated with each instance j of M_i. The **output variable set** of module M_i, namely O_{M_i}, is the set of all the output variables associated with each instance j of M_i.

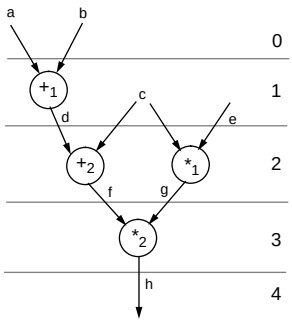

Fig. 2. A scheduled DFG

For the scheduled DFG of Fig. 2 and the above mentioned module assignment $I_{M_1} = \{a, b, c, d\}$ and $O_{M_1} = \{d, f\}$.

For a module, any register to which an input variable is assigned can be used as a TPG and any register to which an output variable is assigned can be used as an SA. The distribution of the elements of the input variable sets and output variable sets of modules determines the possible candidates for TPGs and SAs for those modules. In determining a minimal area BIST solution for a design, different BIST embeddings for a module are explored. Embeddings that share test resources between modules are desired since they reduce the total number of test resources required to test the data path. Thus for maximizing the sharing of TPGs between modules, it is desirable to assign registers such that each register has variables in common with as many input variable sets as possible. Similarly the number of output variable sets with which each register has at least one common variable should be maximized. Also, a CBILBO register is expensive since its area is approximately twice that of a normal register. In a globally minimal BIST area overhead solution, a register could be modified into a CBILBO register even though it is not necessary to do so. However a situation where modifying a register to a CBILBO is *absolutely necessary* for good fault coverage is the one which results in high BIST area overhead. A detailed description of the conditions for maximizing sharing of test resources and minimizing essential CBILBOs during register and interconnect assignment and the associated algorithms is given in [8].

III. Lower bounds on test resources

A necessary condition for a module assignment to produce a valid circuit implementation is that the operations corresponding to a shared resource (i.e. all $v \in M_i$) do not execute concurrently. A schedule determines the module assignment solution space. Module assignment in turn determines the input and output variable sets of modules. The variables in these sets can be distributed across registers to ensure maximum sharing of registers as test resources between modules resulting in low BIST area overhead. A schedule also determines the lifetimes of variables and affects their compatibility when being assigned to registers. Thus a schedule and a module assignment together affect what can be achieved by register and interconnect assignment in terms of the potential of sharing registers as test resources and avoiding essential CBILBOs. There is a great deal of flexibility in register and interconnect assignment in terms of optimizing for BIST

area overhead. However, the optimum that can be achieved is bounded by the schedule and module assignment. Typically, several schedules and module assignments that have a desirable latency and functional area can differ significantly in their test resource requirements. For a given schedule and module assignment, establishing what can be achieved in terms of register and interconnect assignment to minimize test resources is a key question in incorporating testability overhead optimization techniques in the scheduling phase of high-level synthesis.

A. Lower bounds on the number of TPGs and SAs

We address the following two questions. Given a scheduled DFG and a module assignment Π_M, among all register and interconnect assignments that can be associated with the given schedule and module assignment 1) what is the *lower bound* on the number of TPGs required to generate patterns to test all the modules, and 2)what is the *lower bound* on the number of SAs required to compress test responses for all the modules?

Note that we are interested in finding the lower bounds on the BIST area while relaxing the functional area constraints. The *total number* of registers or amount of interconnect required to achieve this bound is not being considered. A data path that achieves these bounds could have a higher functional area than another data path that might not meet the bounds on the number of TPGs or SAs.

Consider the following example of a scheduled DFG shown in Fig. 3(a) and the following two module assignments.

Assignment I:(Fig. 3(b)) Operations '$+_1$' and '$+_3$' are assigned to one module and operation '$+_2$' is assigned to a second module. $M_1 = \{+_1, +_3\}$ and $M_2 = \{+_2\}$.

Assignment II:(Fig. 3(c)) Operations '$+_1$' and '$+_2$' are assigned to one module and operation '$+_3$' is assigned to a second module. $M_1 = \{+_1, +_2\}$ and $M_2 = \{+_3\}$.

Consider all possible register and interconnect assignments for the above two module assignments. Any register which is assigned at least one variable from the output variable set of a module can be used as a SA for that module.

For Assignment II, variables a and c can be assigned to the same register since their lifetimes do not overlap and this register can be used as a SA to test both M_1 and M_2 since $a \in O_{M_1}$ and $c \in O_{M_2}$. Hence the lower bound on the number of SAs in this case is $LB_{\#SAs} = 1$.

For Assignment I it is not possible to find such a register assignment. In this case $O_{M_1} = \{a, c\}$ and $O_{M_2} = \{b\}$. Since the lifetime of b overlaps with the lifetimes of both a and c it cannot be assigned to a register to which a or c is assigned. Hence for any register assignment the minimum number of SAs required to test M_1 and M_2 is $LB_{\#SAs} = 2$.

An analogous situation occurs in the case of the input variables of operations corresponding to the lower bound on the number of TPGs. The concepts of *output storage concurrency*, *inputs storage concurrency* and *maximal concurrent operation set* defined below help model the dependence of BIST area on schedules and module assignments.

Definition 3: a) The output variable of an operation $v \in V$ is the variable corresponding the outgoing edge of v in G. For a scheduled DFG, $G = (V, E)$ the **output storage concurrency** of a subset of operations $Q \subseteq V$, denoted by $C_{OS}(Q)$, is the maximum number of output variables of operations in Q alive at the same time.

b) An input variable of an operation $v \in V$ is a variable corresponding to an incoming edge of v in G. For a scheduled DFG, $G = (V, E)$ the **inputs storage concurrency** of a subset of operations $Q \subseteq V$, denoted by $C_{IS}(Q)$, is the maximum number of input variables of operations in Q alive at the same time.

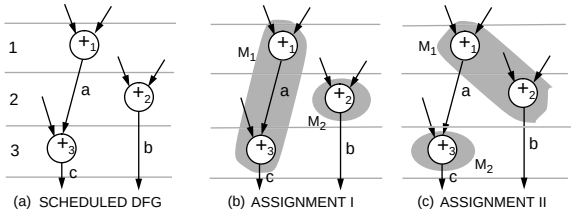

Fig. 3. Effect of module assignment on $LB_{\#SAs}$

The output storage concurrency is a measure of the minimum number of storage locations that would be required to store the outputs of the operations in Q. For the scheduled DFG in Fig. 3, for the subset of operations $Q_1 = \{+_1,+_2\}$, the output storage concurrency $C_{OS}(Q_1) = 2$ since *both* the output variables a and b are alive at the same time. Similarly for $Q_2 = \{+_1,+_3\}$, $C_{OS}(Q_2) = 1$ and for $Q_3 = \{+_1,+_3,+_2\}$, $C_{OS}(Q_3) = 2$.

Definition 4: Given a scheduled DFG, $G = (V, E)$ and a module assignment $\Pi_M = \{M_1,M_2,...,M_m\}$ a **maximal concurrent operation set** $V_{C_{max}}$ is a set of m operations, each of which is assigned to a different module.

A maximal concurrent operation set refers to the maximal set of operations that can be executed simultaneously in real time because each of them have a dedicated hardware resource to which they are mapped by the module assignment. Note that in the actual behavior (scheduled DFG) these operations may not be required to execute concurrently. However the module assignment has assigned resources such that they *can* execute concurrently. For Assignment II discussed previously there are two maximal concurrent operation sets $V^1_{C_{max}} = \{+_1,+_3\}$ and $V^2_{C_{max}} = \{+_2,+_3\}$.

Using the concepts of *maximal concurrent operation set*, *output storage concurrency* and *inputs storage concurrency* we can compute the lower bound on the number of SAs and TPGs required for any data path synthesized from a particular schedule and module assignment. The proofs of Theorem 1 and of the lemmas and theorems to follow are presented in [10].

Theorem 1: For a given scheduled DFG $G = (V, E)$ and module assignment Π_M, the lower bound on the number of SAs required to test all the modules in the data path is

$$LB_{\#SAs} = \min_{\forall\, V_{C_{max}}} C_{OS}(V_{C_{max}}),$$

where the minimum is over all maximal concurrent operation sets of Π_M.

For Assignment I in the previous example, the maximal concurrent operation sets are $V^1_{C_{max}} = \{+_1,+_2\}$ and $V^2_{C_{max}} = \{+_3,+_2\}$. Hence the lower bound on the number of SAs is $LB_{\#SAs} = \min\{ C_{OS}(V^1_{C_{max}}), C_{OS}(V^2_{C_{max}}) \} = \min\{2, 2\} = 2$. For Assignment II, the maximal concurrent operation sets are $V^1_{C_{max}} = \{+_1,+_3\}$ and $V^2_{C_{max}} = \{+_2,+_3\}$. Hence $LB_{\#SAs} = \min\{ C_{OS}(V^1_{C_{max}}), C_{OS}(V^2_{C_{max}}) \} = \min\{2, 1\} = 1$.

Similarly, the lower bound on the number of TPGs can be computed using the *input* storage concurrency of the maximal concurrent operation sets.

Theorem 2: For a given scheduled DFG $G = (V, E)$ and a module assignment Π_M, the lower bound on the number of TPGs required to test all the modules in the data path is

$$LB_{\#TPGs} = \min_{\forall\ V_{C_{max}}} C_{IS}(V_{C_{max}}),$$

where the minimum is over all maximal concurrent operation sets of Π_M.

The lower bounds derived above are *tight*. Given complete flexibility in assigning registers and interconnect without any area constraint, these bounds can be achieved. Theorems 1 and 2 indicate that to lower the BIST area overhead of the synthesized data path a module assignment that has a maximal concurrent operation set with low output (and inputs) storage concurrency is preferable.

B. Lower bounds on CBILBOs

Any self-adjacent register *may* be modified into a CBILBO for the test mode. However, a CBILBO is absolutely required to test a module (or is *essential*) only if all possible BIST embeddings of the module use the same register as a TPG and SA. Thus, self-adjacency is only a *necessary* condition for a register to be an essential CBILBO. To determine the lower bound on the number of CBILBOs we first need to know the module assignment condition that, when followed by *any* possible register and interconnect assignment, results in a self-adjacent register.

Lemma 1: Given a scheduled DFG $G = (V, E)$ and module assignment $\Pi_M = \{M_1, M_2, ..., M_m\}$ a self-adjacent register is created in the data path irrespective of the register and interconnect assignments if and only if there exists a module M_i such that $I_{M_i} \cap O_{M_i} \neq \phi$. The registers to which any element(s) of $I_{M_i} \cap O_{M_i}$ is assigned are self-adjacent registers.

Lemma 1 states that variables that are members of the input variable set as well as the output variable set of a module form a self-adjacent register. So any register to which these variables are assigned will form a self-adjacent register. However, the temporal multiplicity of a module determines whether the self-adjacent register has to be an essential CBILBO [8]. We have defined the concept of an *essential concurrent operation* and *storage concurrency* to find lower bounds on the number of CBILBOs.

Definition 5: An operation is an **essential concurrent operation** if it is an element of *all* maximal concurrent operation sets of a module assignment.

From the definition of a maximal concurrent set, the module to which an essential concurrent operation is assigned has only *one* operation assigned to it, i.e., the temporal multiplicity of the module is 1. The module assignment condition for an essential CBILBO is stated next as Theorem 3.

Theorem 3: Given a schedule and a module assignment Π_M, a CBILBO is essential to test module M_i for all register and interconnect assignments if $TM(M_i) = 1$ (i.e. only one operation is assigned to M_i) and $I_{M_i} \cap O_{M_i} \neq \phi$.

If a module assignment has the property stated in Theorem 3, the register assignment and interconnect assignment cannot avoid a CBILBO. If the register and interconnect assignment is performed without regard for functional area but with the sole objective of minimizing CBILBOs, the number of CBILBOs would depend on the number of essential concurrent operations with overlapping input and output variable sets.

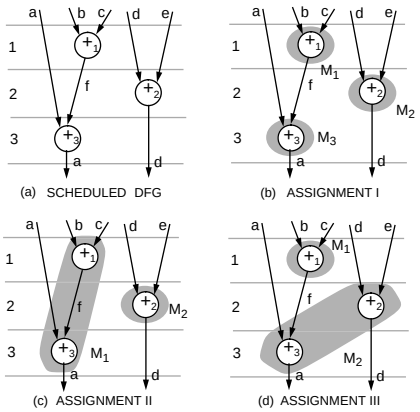

Fig. 4. Module assignments and $LB_{\#CBILBOs}$

Definition 6: The **essential concurrent operation set** V_{ess} of a module assignment is a maximal set of operations that exist in all the maximal concurrent operation sets of the module assignment. The set of modules corresponding to the operations in V_{ess} shall be denoted by M_{ess}.

The essential concurrent operation set associated with a module assignment is thus the set of *all* essential concurrent operations of that module assignment.

Definition 7: The **storage concurrency** of a set of variables Q, denoted as $C_S(Q)$, is the maximum number of variables in Q that are alive at the same time.

Theorem 4: Given a scheduled DFG $G = (V, E)$ and module assignment $\Pi_M = \{M_1, M_2, ..., M_m\}$, the lower bound on the number of CBILBOs required to test all the modules is

$$LB_{\#CBILBOs} = C_S\Big(\bigcup_{\forall M_i \in M_{ess}} (I_{M_i} \cap O_{M_i})\Big)$$

Consider the scheduled DFG shown in Fig. 4(a). Note that the output and one input variable of operations '$+_3$' and '$+_2$' is the same, namely, a and d. This occurs in the case of iterative computations. For scheduling purposes the loop is broken. Consider the following three module assignments for the scheduled DFG.

Assignment I: (Fig. 4(b)) Operations '$+_1$', '$+_2$' and '$+_3$' are all assigned to different modules, i.e. $M_1 = \{+_1\}$, $M_2 = \{+_2\}$ and $M_3 = \{+_3\}$. Since $I_{M_2} \cap O_{M_2} = \{d\} \neq \phi$ and $I_{M_3} \cap O_{M_3} = \{a\} \neq \phi$, according to Lemma 1 registers to which variables a and d will be assigned will be self-adjacent registers. There is only one maximal concurrent operation set $V_{C_{max}} = \{+_1, +_2, +_3\}$ and each of the operation is an essential concurrent operation. The intersection of the input and output variable sets is non-empty only for two of the three modules to which these essential concurrent operations are assigned, namely M_2 and M_3. The lower bound on the number of CBILBOs according to Theorem 4 is $C_S(\{a, d\}) = 2$ since both variables are alive at the same time.

Assignment II: (Fig. 4(c)) In this assignment the essential concurrent operation '$+_3$' in Assignment I is assigned to the same module as operation '$+_1$'. Now we have $M_1 = \{+_1, +_3\}$ and $M_2 = \{+_2\}$. This module assignment has only one essential concurrent operation, namely operation '$+_3$'. Since for module M_2 to which this essential concurrent operation is assigned, $I_{M_2} \cap O_{M_2} = \{d\} \neq \phi$, according to Theorem 4 the

lower bound on the number of CBILBOs is $C_S(\ \{d\}\ ) = 1$. Note that $I_{M_1} \cap O_{M_1} = \{a, f\} \neq \phi$. So registers to which variables a and f will be assigned will be self-adjacent registers. Note that assignment of variable f results in a self-adjacent register only because '$+_1$' and '$+_3$' are assigned to the same module. Also even though M_1 has self-adjacent registers it does not require a CBILBO as $TM(M_1) = 2$.

Assignment III: (Fig. 4(d)) In this assignment we have $M_1 = \{+_1\}$ and $M_2 = \{+_2, +_3\}$. This module assignment also has only one essential concurrent operation as in Assignment II, namely operation '$+_1$'. However for the module M_1 to which this essential concurrent operation is assigned , $I_{M_1} \cap O_{M_1} = \phi$. Hence the lower bound on the number of CBILBOs is 0. Note that $I_{M_2} \cap O_{M_2} = \{a, d\}$ so registers to which a and d are assigned will be self-adjacent registers but a register assignment can be found such that M_2 does not require a CBILBO.

Thus, to lower the number of CBILBOs in the data path the number of essential concurrent operations that have intersecting input and output variable sets should be reduced. The lower bound can be achieved by relaxing the functional area constraint. A register assignment that achieves this lower bound could have more than the minimum number of functional registers required.

C. Use of lower bound estimation in synthesis algorithms

The ability to predict area-performance characteristics of designs without actually synthesizing them is vital to produce quality designs in a reasonable time. Computation of lower bounds on *test resources* provides a synthesis system with a quick way of evaluating the testability overhead of the design. More specifically, the proposed lower bounds can be used for the following.

1. To select schedules from a set of schedules with the same latency and resource requirement. For a given behavior, different schedules are possible such that they have the same latency and satisfy the same module constraints but have different lower bounds for test resources.
2. Given a schedule, to find a module assignment that requires few test resources. For a given schedule, different module assignments have different lower bounds on test resources. The lower bound estimation technique can be used to compare different module assignments or to *incrementally* perform module assignment for a solution with a low test resource requirement.
3. To trade-off latency for area. Generally, latency is traded-off for a reduction in the number of modules. Often, increasing latency by a few clock steps does not reduce the module requirement. However, it could affect the lower bounds on test resources and thus BIST area.
4. To prune the search space and direct the search during register and interconnect assignment towards low testability overhead designs.
5. To provide a common base for evaluating the quality of different high-level synthesis systems and algorithms that have testability overhead as an optimization objective.

We have developed efficient algorithms for quick computation of the lower bounds. The algorithm for computing lower bounds on the number of TPGs and SAs has worst case complexity $O(L \cdot m \cdot (\frac{n}{m})^m)$, where L is the latency of the schedule, n is the number of operations (nodes in the DFG) and m is the number of modules assigned. The algorithm is efficient in spite of the exponential complexity because the number of modules m is usually small. Furthermore, properties of module assignments and maximal concurrent operation sets can be used to reduce the size of the exponential space. The lower bound on CBILBOs can be computed in $O(L \cdot m)$ time. A detailed description of the theory and algorithms can be found in [10].

TABLE I Lower bounds for minimum latency schedules of *diffeqn*

Sched.	Type of sched.	Lat. (L)	# mod. (m)	Lower bounds		
				SA	TPG	CBILBO
S_1	ALAP	4	5	3	5	1
S_2	ASAP	4	5	2	4	0
S_3	Inter.	4	5	3	4	0

TABLE II Lower bounds for *diffeqn*

Sched.	Lat. (L)	# mod. (m)	Lower bounds		
			SA	TPG	CBILBO
S_1	6	4	2	4	1
S_2	6	4	2	4	0
S_3	6	4	1	4	0

IV. Experimental results

To demonstrate the use of the proposed lower bound computation in evaluating the testability qualities of schedules and module assignments, we applied it to some well-known high level synthesis benchmarks: 1) the 2nd order differential equation - *diffeqn*, 2) the Tseng data flow graph - *Tseng*, 3) the auto regression filter element - *AR Filter*, and 4) the 5th order elliptic wave filter - *ewf* [11].

Table I depicts bounds for three different schedules of *diffeqn*. The minimum latency for this benchmark is 4. As-late-as-possible (ALAP) scheduling and as-soon-as-possible (ASAP) scheduling are two popular scheduling techniques for achieving minimum latency schedules. Both ASAP and ALAP schedules of the *diffeqn* require 5 modules. It can be seen that the lower bound on SAs, TPGs and CBILBOs required if schedule S_2 is used is lower than the lower bounds of schedule S_1. The bounds for an intermediate schedule, S_3, with the same latency are also shown. Three more schedules each with a latency of 6 and using 4 modules are shown in Table II. These results demonstrate that schedules that are equally attractive from a functional resource and latency point of view can differ greatly in the minimum test resource requirement.

The *Tseng* benchmark does not have any variable that is an input as well as an output variable of the same operation. Hence according to Theorem 3, the lower bound on CBILBOs is zero. We investigated the lower bounds on SAs and TPGs for all possible schedules and all possible module assignments associated with each schedule for this benchmark. Table III shows the lower bounds on SAs and Table IV shows the lower bounds on TPGs. Each entry in the tables corresponds to the minimum lower bound among all possible schedules and module assignments for that particular latency and number of modules. For example, among all module assignments using 6 modules that were possible for different schedules of latency 5, the minimum lower bound on the number of SAs was 3. A '$-$' entry indicates that no schedule and module assignment

TABLE III Variation in SA lower bounds for *Tseng*

Latency (L)	Number of modules (m)							
	1	2	3	4	5	6	7	8
4	–	–	1	2	2	3	3	4
5	–	1	1	1	2	3	3	4
6	–	1	1	1	2	2	2	4
7	–	1	1	1	2	2	2	4
8	1	1	1	1	2	2	2	4

TABLE IV Variation in TPG lower bounds for *Tseng*

Latency (L)	Number of modules (m)							
	1	2	3	4	5	6	7	8
4	–	–	2	3	3	4	5	5
5	–	2	2	3	3	4	5	5
6	–	2	2	3	3	4	4	5
7	–	2	2	3	3	4	4	5
8	2	2	2	3	3	4	4	5

TABLE V Lower bounds for *AR Filter*

Sched.	Lat. (L)	Module assign.	# mod. (m)	Lower bounds		
				SA	TPG	CBILBO
S_1 ALAP	8	M_1	12	5	16	0
		M_4	12	4	12	0
		M_3	12	2	8	0
S_2 ASAP	8	M_1	12	5	12	0
		M_2	12	3	8	0
		M_3	12	2	8	0

solution is possible for that (L, m) value of the DFG. It can be observed that the bounds increase as the number of modules increases. Note that the *actual* functional area corresponding to the modules is not being considered here. The actual functional area depends on the particular module assignment and a higher number of modules does not necessarily imply a larger functional area [12]. Tables III and IV indicate that the lower bounds have a strong correlation to the *number* of modules. They also demonstrate that among two module assignments Π^1_M and Π^2_M such that $\mid \Pi^2_M \mid < \mid \Pi^1_M \mid$, assignment Π^2_M might be desirable from the test resources point of view even if $Area(\Pi^2_M) > Area(\Pi^1_M)$.

Table V shows the bounds for different module assignments of the ASAP and ALAP schedules for the *AR Filter* benchmark. The latency of both schedules is 8 and the minimum number of modules for this latency is 12. All module assignments in Table V use 12 modules. Table VI shows the bounds for different module assignments for a schedule of the *ewf* benchmark. The latency of the schedule used is 19. These results show that a significant variation in test resources exist for different module assignments of the same schedule as well as different schedules of the same latency.

The lower bound estimates can be used to compare the quality of the synthesized designs in terms of BIST resources. The closer the number of BIST resources are to the lower bounds, the better the quality of the synthesis algorithms in synthesizing low BIST overhead designs. In Table VII, the BIST resources required for design synthesized by the approach in [8] are compared to the lower bounds. *ex2* is a DFG taken from [13]. *Tseng1* and *Tseng2* are different module assignments of the *Tseng* benchmark. The lower bound on test resources was achieved in most of the cases which indicates that the synthesis algorithm performed well in optimizing test resources and that the proposed bounds are achievable *even when functional area constraints are imposed.*

V. Conclusions

In this paper we have derived tight lower bounds on *test resources* that would be required to test the synthesized data path using partial intrusion BIST. The lower bound estimation is performed on scheduled data flow graphs, and given complete flexibility in the assignment of registers and interconnect, the bounds can be achieved. The bounds give a mechanism for comparing the quality of area-performance competitive schedules and module assignments with respect to test resource requirement. The bounds along with a library of test register modules can give an estimate of the actual test area overhead.

TABLE VI Lower bounds for *ewf* (L = 19)

Module assign.	# mod. (m)	Lower bounds		
		SA	TPG	CBILBO
M_1	8	3	6	2
M_2	8	2	4	1
M_3	8	1	3	0

TABLE VII Actual test resources v/s lower bounds

DFG	# TPGs		# SAs		# CBILBOs	
	Act.	LB	Act.	LB	Act.	LB
ex2	4	4*	3	3*	1	0
Tseng1	5	5*	4	3	1	0
Tseng2	3	3*	2	2*	0	0*
diffeqn	3	3*	2	2*	1	1*

* Lower bound achieved

The theory on *test resource* bounds can be used in conjunction with that for estimating *functional resources* so that the *total* area of the synthesized design can be accurately estimated.

References

[1] Y. Hu, A. Ghouse, and B.S. Carlson. Lower Bounds on the Iteration Time and the Number of Resources for Functional Pipelined Data Flow Graphs. In *Proc. Intn'l Conf. Comp. Design*, pages 21–24, October 1993.

[2] A. Sharma and R. Jain. Estimating Architectural Resources and Performance for High-Level Synthesis Applications. *IEEE Trans. on VLSI Systems*, 1(2):175–190, June 1993.

[3] S. Chaudhari and R.A. Walker. Computing Lower Bounds on Functional Units before Scheduling. In *Proc. 7th Intn'l Symp. on High-level Synthesis*, pages 36–41, May 1994.

[4] S.Y. Ohm, F.J. Kurdahi, and N. Dutt. Comprehensive Lower Bound Estimation from Behavioral Descriptions. In *Proc. Intn'l Conf. Computer-Aided Design*, pages 182–187, October 1994.

[5] L. Avra. Allocation and Assignment in High-level Synthesis for Self-testable Data Paths. In *Intn'l. Symp. on Circuits and Systems*, pages 463–472, Aug. 1991.

[6] H. Harmanani and C. Papachristou. An Improved Method for RTL Synthesis with Testability Tradeoffs. In *Proc. Intn'l Conf. on Computer-Aided Design*, pages 30–35, November 1993.

[7] I.G. Harris and A. Orailoglu. SYNCBIST:SYNthesis for Concurrent Built-In Self-Testability. In *Proc. Intn'l Conf. Comp. Design*, pages 101–104, October 1994.

[8] I. Parulkar, S.K. Gupta, and M.A. Breuer. Data Path Allocation for Synthesizing RTL Designs with Low BIST Area Overhead. In *Proc. 32nd Design Automation Conf.*, pages 395–401, June 1995.

[9] L.T. Wang and E.J. McCluskey. Concurrent Built-In Logic Block Observer (CBILBO). In *Intn'l. Symp. on Circuits and Systems*, pages 1054–1057, 1986.

[10] I. Parulkar, S.K. Gupta, and M.A. Breuer. *Estimating BIST Resources in High-level Synthesis.* CEng Tech. Report 96-06, Univ. of Southern California, Dept. of Elect. Engineering - Systems, March 1996.

[11] N. Dutt and C. Ramachandran. *Benchmarks for the 1992 High-level Synthesis Workshop.* Tech. Report 92-107, Univ. of California, Irvine, 1992.

[12] K. Kucukcakar and A. Parker. Data Path Trade-offs using MABAL. In *Proc. 27th Design Automation Conf.*, pages 511–516, June 1990.

[13] C. Papachristou, S. Chiu, and H. Harmanani. A Data Path Synthesis Method for Self-Testable Designs. In *Proc. 28th Design Automation Conf.*, pages 378–384, June 1991.