An Integrated Approach to Behavioral-Level Design-For-Testability Using Value-Range and Variable Testability Techniques

Sandhya Seshadri‡ and Michael S. Hsiao‡
‡ Mentor Graphics Corporation, Warren, NJ
‡ Department of Electrical & Computer Engineering, Rutgers University, Piscataway NJ

Abstract
This research applies formal dataflow analysis and techniques to high-level DFT. Our proposed approach improves testability of the behavioral-level circuit description (such as in VHDL) based on propagation of the value ranges of variables through the circuit’s Control-Data Flow Graph (CDFG). The resulting testable circuit is accomplished via controllability and observability computations from these value ranges and insertion of appropriate testability enhancements, while keeping the design area-performance overhead to a minimum.

I Introduction
Traditionally, VLSI design and test processes have been kept separate, with test considered only at the end of the design cycle. However, in contemporary design flows, test merges with design much earlier in the process, creating a design-for-test (DFT) process flow, the goal of which is to produce hierarchically testable designs. A design is said to be hierarchically testable, if the input (output) ports of every module in the design hierarchy, are easily controllable (observable) through system inputs (outputs) [1].

Classical DFT strategies have been conventionally limited to the end of the design phase, when a detailed gate-level description of the design is available [6, 7, 8]. Testability features that are incorporated into the design at the post-synthesis step result in area and delay penalties. These structural testability enhancement techniques fall into the following categories: Built-in self-test (BIST), non-scan and scan-based DFT. In general, the amount of scan required to get an acceptable fault coverage varies from design to design. At-speed testing of a circuit is not possible when scan tests are used, due to scanning in and out of flip-flop values.

Recent studies [1, 2, 3, 4, 5, 13, 15] have shown that if testability is not addressed during behavioral synthesis, many modules and registers in the resultant register-transfer-level (RTL) circuit may not be testable for the operations and/or variables mapped to them. The problem is exacerbated in the presence of loops, constants and reconvergent fanout in the control-data flow graph (CDFG) corresponding to the design.

In [9], the authors proposed a partial-scan selection mechanism that works on the CDFG of a design derived from its behavioral description. Hard-to-test areas of the design are identified by the evaluated controllability and observability measures, and test point insertion techniques such as those described in [5] are applied. Lastly, flip-flops corresponding to variables that are hard to control and observe are selected for partial-scan to maximize the impact on testability of the design. The controllability measures obtained in this fashion, however, do not consider the value ranges of the variables and the probabilities of propagating these value ranges to the nodes in the CDFG. In [17], the authors propose a behavioral testability enhancement technique based on the analysis of values and variable probabilities obtained from profiling of the high-level description. The simulations conducted during profiling would have to be exhaustive in order to obtain all of the values and their probabilities for every variable.

The contribution of our work is to enhance the testability of a behavioral-level design by accurately pin-pointing the hard-to-test regions of the circuit based on an entirely new approach involving a formal dataflow analysis, rather than by inspection or profiling. The improved accuracy of the controllability and observability measures of the variables are a result of applying a value range propagation technique [10, 11, 14, 16]. Although the techniques used in [10, 11, 14, 16] were developed to improve the accuracy of static value and branch prediction in compilers, we have observed that they can be adapted in our DFT work.

A is initialized here

\[ A := A + 1 \]

\[ Y := f(A) \]

\[ Z := g(Y) \]

Loop Exit

Figure 1: A Simple Example

Figure 1 shows an example of a variable A incremented within a loop in the high-level description. The variables Y and Z are defined to be functions of A and Y, respectively, inside the loop. Variable Y can be hard to control to any specific value if the function \( f() \) is non-trivial. Similarly,
Y can be hard to observe because of $g()$. Furthermore, the nature of the loop makes controlling and observing Y even more difficult. Our approach aims to determine the range of values that the variables A and Y can attain as well as the probability of these variables attaining these values. A static single assignment representation (SSA) is used as a data-flow representation for propagating values and their probabilities through a VHDL program. This information enables us to compute accurate measures of controllabilities and observabilities of the variables at different instances throughout the VHDL code. Based on the testability measures computed, a combination of control-flow enhancement, overloading of selected variables, and partial-scan of selected variables are applied correspondingly to improve the circuit testability, while keeping the area-performance overhead to a minimum.

The remainder of the paper is organized as follows. Section II briefly explains the static single assignment (SSA) representation and value range propagation; Section III describes the use of SSA variables and their value ranges to testability enhancement; experimental results are discussed in Section IV, and Section V concludes the paper.

II Preliminaries

The technique we propose takes as input a high-level behavioral description of the design in a hardware description language such as VHDL. A data-flow representation geared towards propagating values through a static single assignment (SSA) representation [11, 14] is used. In optimizing compilers, the SSA form and the CDFG are used to represent data-flow and control-flow properties of programs, lending efficiency and power to a useful class of optimizations. Such program optimizations include code motion and elimination of partial redundancies, as well as constant and variable propagation. The salient features of the SSA representation are as follows:

1. Each definition of a variable is assigned a unique name. A define of a variable V takes place when V receives a new value.

2. At points in the program, a new name is generated to combine the results from several definitions of a variable. Such a join is performed via a $\phi$-function. The function determines which of the definitions to use, based on the flow-path traversed in the VHDL code.

3. Each use of a variable makes use of exactly one name generated from either of the two rules above. Use of V occurs when V is needed in the definition of itself or another variable.

Because the SSA representation distinguishes every pair of variable definition locations, it is more suitable for analyzing, propagating, and merging value ranges than the abstract syntax tree (AST). In addition, with the SSA representation, we are able to pin-point the exact instances (e.g., at the entry of the loop or at a fan-out point) where a variable is difficult to control. The behavioral level description makes it convenient to mix the control logic with data-path. The SSA representation and value range propagation also take into account how the variables relate to one another, since all variables that depend on one another are considered during the analysis.

A basic block is a straight-line code sequence with no transfers in or out, except at the beginning or end of the block. A simple assignment in the functional description results in the application of rule (1), creating a new component SSA variable. When rule (2) is applied to create a new component SSA variable, the resulting $\phi$-function assignment has the form

$$x = \phi(p, q, ...),$$

where $x$ gets the value $p$ if the control flows into the basic block via the first in-edge; on the other hand, $x$ is assigned the value $q$ if the basic block is reached via the second in-edge, and so on. In our SSA representation, we are concerned with $\phi$-function assignments made at the beginning of each loop structure and at a join of the fan-outs of each branch condition in the high-level description.

The notation to describe a set of $m$ weighted value ranges is represented as \{\text{W}_i[\text{L}_i : \text{U}_i : \text{S}_i], \ldots\}, where \text{W}_i is the probability of the particular range applying at runtime, \text{L}_i and \text{U}_i give the lower and upper bounds of each range, \text{S}_i is the stride, and $i = 1, 2, 3, \ldots m$. \text{S}_i is the size of the step taken in going from the lower bound \text{L}_i to the upper bound \text{U}_i. Consider a weighted value range $0.8[3 : 11 : 2]$ for variable V, for example. V assumes values in stride intervals of $S = 2$, starting with the lower bound \text{L} = 3 and reaching the upper bound of \text{U} = 11. Thus, at runtime, the set of possible values that V can take is \{3, 5, 7, 9, 11\}, and the probability of V being assigned a value from this set is $W = 0.8$. Weighted value ranges of variables in the SSA representation are then propagated until an output assignment for each variable is determined. The propagation of the weighted value ranges of variables through the data flow graph is done in the same way that constants are propagated in traditional constant propagation [11].

We will use Figure 2 to illustrate how value ranges and testability measure are computed. An example VHDL description is shown on the right, with the corresponding SSA representation shown on the left. The candidate variables we will discuss are $X$, $Y$, and LOOPING. In this example, variable $X$ has 5 defines, 5 uses, and 2 joins (at the for-loop and end of branch) in the VHDL code; so there are seven SSA components, $X0$ to $X6$.

Let us look at the variable $X$ first. The first instance of $X$ being defined is assigned an SSA variable $X0$, and it is initialized to 0 at the beginning of the SSA flow-graph; thus, the weighted value range for $X0$ is simply $1.0[0:0:0]$. Next, we extract the information of the join at the top of the for-loop which is controlled by $X$ (where two different versions of $X$ meet); therefore, we allocate another SSA component, $X1$, to $X$. We compute the value range for SSA component $X1$ by merging the value ranges of the two
different versions. Because one of the inputs at this join, $X_6$, has not been computed, it may take several iterations through the SSA flow-graph to determine the value range for $X_1$. The value ranges of the other SSA components of $X$ are computed similarly.

For branch structures, consider a branch with $f$ paths with probability from the $k^{th}$ branch path equal to $P_k$, $k = 1, 2, 3, ...$, $f$. Assuming that the variable from the $k^{th}$ path contributes a set of $n$ value ranges $\{W_i^k[|L_i^k : U_i^k| : S_i^k]\}$, $i = 1, 2, 3, ... , n$, the contribution from this path to the resulting set of value ranges can be computed as

\[
\{P_k\{W_i^k[|L_i^k : U_i^k| : S_i^k]\}\}, k = 1, 2, 3, ... , f,
\]

\[
\text{and } i = 1, 2, 3, ... , n.
\]

The total number of value ranges attributed to the destination variable of the $\phi$-function assignment is then

\[
\text{nb} = \sum_{k=1}^{f} \text{number value ranges}(k^{th} \text{ SSA variable}).
\]

For example, variable $Y$ has a $\phi$-function assignment to its component SSA variable $Y_3$ at the join of branch $X_2 \leq 1$. Here, the first fan-in path contributes a set of value ranges $\{1.0[-1 : 0 : 1]\}$ and the second contributes a set of value ranges $\{1.0[3 : 10 : 1]\}$. The probability of a branch (i.e., the true path of the branch is taken) can be computed as the fraction of the value range of branch-condition variable for which the branch condition evaluates to true. For instance, the SSA branch statement $X_2 \leq 1$ is taken if and only if the value of $X_2$ is either 0 or 1. Given that $X_2$ has a value range of $1.0[0 : 9 : 1]$, probability of the branch being taken is 20%. With the probabilities of the 2 paths being 0.2 and 0.8, the set of value ranges of $Y_3$ becomes $\{0.2[1.0[-1 : 0 : 1]], 0.8[1.0[3 : 10 : 1]]\}$.

Finally, for the variable $\text{LOOPING}$, there are three SSA components. The SSA component $\text{LOOPING2}$ represents the assignment to $\text{LOOPING}$ after we exit the loop, thus the probability for its value range is

\[
(1 - P_{\text{loop}}) \# \text{ iterations through the loop}.
\]

Because the loop iterates ten times, the probability of staying in the loop is simply 10/11, or $P_{\text{loop}} = 0.91$. Unrolling the loop will not ease the computation of all the value ranges of the variables since we will have only have more variables to analyze and more instances of $\phi$-functions to compute. Table 1 shows the final value ranges for all component SSA variables of $X$, $Y$, and $\text{LOOPING}$.

### III High-Level DFT Framework

Our approach considers all internal variables defined in the behavioral-level. The list of candidate variables for the new DFT framework does not incorporate any primary input variables and output variables that do not feed back into the design. Each variable in the candidate list is composed of many SSA variables corresponding to the given high-level behavioral description. The value range information on each of the constituent SSA variables is then applied toward the derivation of controllability and observability measures for the corresponding variable, as will be explained in the following subsections.

#### A Controllability analysis

A simple or a $\phi$-function assignment to a component SSA variable $v_j$ of a variable $V$ results in a set of $n$ value ranges $\{W_i^k[|L_i^k : U_i^k| : S_i^k]\}$, $i = 1, 2, 3, ... , n$. Given a value range

![Figure 2: A Behavioral Description and Its SSA Representation.](image-url)
W[L : U : S] of v_j and a value a in the range, the controllability value \( C_a \) of v_j is a measure of the difficulty in assigning a to v_j. The value of \( C_a \) is computed as

\[
C_a = 1/P_a.
\]  

(1)

where \( P_a \) is the probability of v_j assuming the value a.

Starting from the lower bound L in a range W[L : U : S], it is necessary to sequence the circuit many cycles in order to reach the upper bound U. Each iteration takes v_j away from L toward U by a stride of S, and the total number of iterations, ST, to go from L to U is

\[
ST = \frac{U - L}{S}.
\]  

(2)

If p is the probability of each iteration, then the probability P of reaching U from L is computed as

\[
P = p^{ST}.
\]  

(3)

The probability \( P_L \) of assigning the value of L to v_j is

\[
P_L = W,
\]  

(4)

and the probability \( P_U \) of v_j is computed as

\[
P_U = W \times P.
\]  

(5)

The controllability \( C \) of v_j over the entire set of values in the value range W[L : U : S] is then computed as the average of the controllability of v_j to the lower and upper bounds of the range. Thus,

\[
C = \frac{1}{2} \left( \frac{1}{P_L} + \frac{1}{P_U} \right).
\]  

(6)

To begin computation of controllability measures, the controllability values for all non-input variables are initially set to infinity, and zero for all primary inputs and constant values. The set of values that a primary input can assume is determined by the number of bits used to represent the input in the high-level description. Further, each value in this set is assumed to be equiprobable. For example, in the behavioral description, if a primary input \( I \) is a 5-bit NATURAL, then \( I \) can assume any integer value between 0 and 32 with equal probability.

Each assignment in the SSA representation occurs in a single time step. Both simple assignments and \( \phi \)-function assignments need to be evaluated. Each type of assignment can be inside or outside a loop, and the computation of the testability measures will differ accordingly.

### 1. ASSIGNMENTS INSIDE A LOOP:

**a. Simple Assignments:** These statements are of the form \( v_j = ... \), where \( v_j \) is the destination component SSA variable of the variable \( V \) in the VHDL code. Assuming that the probability of staying in the loop is \( P_{loop} \), then from Equations 4, 5 and 6, the controllability value \( C \) for a destination SSA variable \( v_j \) with a value range \( W[L : U : S] \) is computed as

\[
C = \frac{1}{2} \left[ 1 + \frac{1}{W \times (P_{loop})^{ST}} \right],
\]  

(7)

where \( ST \) is the number of cycles it requires to go from L to U, as computed in Equation 2.

The density of \( v_j \), denoted as \( D \), corresponding to a particular value range is the ratio of the number of unique values that \( v_j \) can take in the given value range to the total number of values that the candidate variable can take. Let the total number of unique values a variable \( V \) can take be \( T \). The value of \( T \) depends on how many bits \( b \) are used to represent the variable \( V \) in the high-level description. For example, if \( V \) is of type BIT, \( T = 2 \); likewise, \( T = 256 \) if \( V \) is an 8-bit NATURAL, and so on. Thus, the density of a variable for a given value range is

\[
D = \frac{ST + 1}{T}.
\]  

(8)
Example 1: Let us consider the simple assignment \( Y_2 = X_4 + 1 \) from Figure 3, which occurs inside a loop \( X_1 \leq 9 \). The probability of staying in the loop is \( P_{\text{loop}} = 0.91 \), as shown in Section II. Because the variable \( Y \) is of type integer data with a value range of -15 to +15, \( T = 31 \). The set of value ranges of \( Y_2 \) was derived to be \{1.0\[3 : 10 : 1]\}. We now compute \( ST \) and \( D \).

\[
ST = (U - L)/S = (10 - 3)/1 = 7,
\]

\[
D = [ST + 1]/T = (7 + 1)/31 = 0.258.
\]

The controllability of \( Y_2 \) in this value range is then

\[
C = [1/W + 1/(W \times (P_{\text{loop}})^{ST})]/2 = [1/1.0 + 1/(1.0 \times 0.91^7)]/2 = 1.468.
\]

For an SSA variable with a set of \( n \) value ranges \{\( V_{R1}, V_{R2}, ..., V_{Rn} \)\} (where each \( V_{Ri} \) is of the form \( W_i[L_i : U_i : S_i] \)), the corresponding \( D_i \) and \( C_i \) are used to compute the density and controllability for the entire SSA variable. They can be computed by

\[
D_{v_j} = \sum_{i=1}^{n} D_i, \quad \text{and} \quad C_{v_j} = \sum_{i=1}^{n} C_i \times D_i / D_{v_j}.
\]

In this particular example, because \( n = 1 \), \( D_{Y_2} \) and \( C_{Y_2} \) for the component SSA variable \( Y_2 \) are simply

\[
D_{Y_2} = \frac{1}{i=1} D_i = D_1 = 0.258,
\]

\[
C_{Y_2} = \frac{1}{i=1} C_i \times D_i / D_{Y_2} = C_1 \times D_1 / D_{Y_2} = 1.468.
\]

b. \( \phi \)-function Assignments: A \( \phi \)-function assignment combines the different versions of the variable \( V \) from all the paths of a branch condition. Such an assignment is made to the component SSA variable \( Y_3 \) in Figure 2. The set of value ranges of \( Y_3 \) is a confluence of the sets of value ranges of the SSA variables \( Y_1 \) and \( Y_2 \), contributed by the fan-outs of the branch condition \( X_2 \leq 1 \). This is represented as \( Y_3 = \phi(Y_1, Y_2) \).

When a \( \phi \)-function assignment to the destination SSA variable \( v_j \) occurs inside a loop with probability, \( P_{\text{loop}} \), of staying in the loop, the set of value ranges of such a variable can be represented as:

\[
\{P^k\{W^k_i[L^k_i : U^k_i : S^k_i]\}\}, \quad k = 1, 2, 3,..., f, \quad i = 1, 2, 3,..., n,
\]

where \( S^k_i \neq 0 \).

The controllability value \( C^k_i \) for \( v_j \), in a value range \( W_i[L_i : U_i : S_i] \) from \( k \)th fan-in path can be computed as:

\[
C^k_i = [1/(P^k \times W_i) + 1/((P^k \times W_i) \times (P_{\text{loop}})^{ST})]/2.
\]

The density of \( v_j \) corresponding to the value range is the ratio of the total number of unique values that it can take in that range to the total number of values \( T \) that the candidate variable \( V \) can take:

\[
D^k_j = [ST + 1]/T,
\]

where \( ST = [U_i - L_i]/S_i \).

The controllability for the component SSA variable \( v_j \) over its complete set of value ranges, from \( k \)th path, can then be computed by the following equations:

\[
C^k_j = \sum_{i=1}^{n} C^k_i \times D^k_i / D_{v_j}, \quad \text{where}
\]

\[
D^k_i = \sum_{i=1}^{n} D^k_i.
\]

Similarly, the controllability of the entire component SSA variable \( v_j \) over its complete set of value ranges, from all \( f \) paths is computed by:

\[
C_{v_j} = \sum_{k=1}^{f} C^k_j \times D^k_j / D_{v_j}, \quad \text{where}
\]

\[
D_{v_j} = \sum_{k=1}^{f} D^k_j.
\]

Example 2: Consider the \( \phi \)-function assignment \( Y_3 = \phi(Y_1, Y_2) \) inside the loop with condition \( X_2 \leq 9 \), shown in Figure 4. The probability of staying in the loop is \( P_{\text{loop}} = 0.91 \), and the total number of unique values that \( Y \) can take is \( T = 31 \). The set of value ranges of \( Y_3 \) is \{0.2\[1.0\[-1 : 0 : 1]\], 0.8\[1.0\[3 : 10 : 1]\]\}; thus, the number of value ranges in the set is \( nb = 2 \). The controllability of \( Y_3 \) contributed by the first value range (first branch path) is

\[
C^1_1 = [1/(P^1 \times W_1) + 1/((P^1 \times W_1) \times (P_{\text{loop}})^{ST})]/2.
\]
= \left[ \frac{1}{0.2 \times 1.0} + 1/(0.2 \times 1.0) \times (0.91)^3 \right] / 2 = 5.247.

Likewise, the density of Y3 corresponding to this value range is \( D_{Y3} = [ST_1 + 1] / T = 0.065 \).

Because only one value range is from the first branch path, the controllability and density values of the component SSA variable \( v_j \) from the first fan-in are the same as the above: \( C_{Y3}^1 = C^1 = 5.247 \), and \( D_{Y3}^1 = D^1 = 0.065 \).

We can compute the controllabilities of Y3 for the second fan-in in a similar fashion, obtaining \( C_{Y3}^2 = 1.834 \), and \( D_{Y3}^2 = 0.258 \).

Now, the density and controllability of Y3 over the complete set of value ranges from both fan-ins using Equations 15 and 16 yield

\[
D_{Y3} = \sum_{k=1}^{2} D_{Y3}^k = 0.323, \quad \text{and} \\
C_{Y3} = \sum_{k=1}^{2} C_{Y3}^k \times D_{Y3}^k / D_{Y3} = 2.521.
\]

If \( \{ W^k_i [L^k_i : U^k_i : S^k_i] \} \), \( i = 1, 2, 3, \ldots, n \), is the set of n value ranges of a destination SSA variable \( v_j \) of a simple or \( \phi \)-function assignment inside a loop, and \( W[L : U : S] \) is one of the value ranges such that \( L = U \) and \( S = 0 \), then the controllability \( C \) of \( v_j \) in this value range is:

\[
[C = \left( 1 / W + 1 / (W \times (P_{loop}^{\text{iter}})) \right) / 2].
\]

where, \( \text{iter} \) = no. of iterations resulting in value range.

The variable \( v_j \) has a density:

\[
D = 1 / T.
\]

Example 3: The variable LOOPING1 in Figure 3 is the destination of a simple assignment inside the loop structure with condition \( X2 \leq 9 \). As before, the probability of staying in the loop is \( P_{loop} = 0.91 \) and the total number of unique values that LOOPING can take is \( T = 2 \). The set of value ranges of LOOPING1 is \{1.0[true : true : 0]\}; thus, \( nb = 1 \). The controllability of LOOPING1 in this value range is computed using Equation 17 to be

\[
C = \left( 1 / 1.0 + 1 / (1.0 \times (0.91)^{10}) \right) / 2 = 1.784
\]

\[
D = 1 / 2 = 0.5
\]

In this particular example, because \( n = 1 \), the density \( D_{\text{LOOPING1}} \) and the controllability \( C_{\text{LOOPING1}} \) over the entire set of value ranges for the component SSA variable LOOPING1 are simply 0.5 and 1.784 respectively.

2. ASSIGNMENTS OUTSIDE A LOOP:

The set of value ranges of a component SSA variable \( v_j \) of a variable V outside a loop is represented as

\( \{ W_i [L_i : U_i : 0] \} \), where \( i = 1, 2, 3, \ldots, n \)

a. Simple Assignments: A simple assignment that occurs outside a loop could be in a basic block that either does or does not follow from the exit edge of the loop. In the former case, the controllability \( C \) and density \( D \) of the destination SSA variable in a particular value range can be computed by simply setting \( ST = 0 \) in the Equations 7 and 8 respectively. In the latter case, the controllability \( C \) and density \( D \) of the destination SSA variable are computed as

\[
C = 1 / (W \times (1 - P_{loop}^{\text{iter}})), \quad \text{and}
\]

\[
D = 1 / T.
\]

As explained before, the probability of exiting the loop is \( (1 - P_{loop})^{\text{iter}} \), where \( \text{iter} \) is the number of iterations of the loop. In both cases, the density and controllability of the component SSA variable \( v_j \) over its entire set of \( n \) value ranges can be computed by the same Equations 9 and 10, respectively.

b. \( \phi \)-function Assignments: \( \phi \)-function assignments that occur outside a loop can be in a basic block that
may or may not follow from the exit edge of the loop. In the former case, the controllability \( C^k_i \) and density \( D^k_i \) of the destination SSA variable in value range \( i \) can be computed by simply setting \( ST_i = 0 \) in Equations 11 and 12 respectively. In the latter case, the controllability \( C^k_i \) and density \( D^k_i \) of the destination SSA variable in a value range are computed as

\[
C^k_i = \frac{1}{1/(P^k \times W_i \times ((1 - P_{loop})^{iter})}, \quad (21)
\]

\[
D^k_i = \frac{1}{1/T}. \quad (22)
\]

In both cases, the controllability and density of the component SSA variable \( v_j \) over its set of value ranges from fan-in \( k \) are computed the same way as in Equations 13 and 14; thus, the equations for controllability and density of the component SSA variable \( v_j \) from all \( f \) fan-ins the same as Equations 15 and 16.

3. LOOP STRUCTURES: The loop structure has two fan-ins: the first fan-in is from the block before the loop and the second fan-in results from the backward edge of the loop with a probability of \( P_{loop} \). The exit edge from the loop has a probability of \( 1 - P_{loop} \). The loop control SSA variable typically has a set of value ranges given by \( \{P^1[L_1 : U_1 : S_1], P^2[L_2 : U_2 : S_2] \} \), where \( P^1 \) represents the probability of the first fan-in, bringing a set of value ranges \( \{W_1[L_1 : U_1 : S_1],\ i = 1, 2, 3, ..., n \} \) and \( P^2 \) represents the probability \( P_{loop} \) of the backward edge of the loop, with a value range of \( 1.0[L_2 : U_2 : S_2] \). The controllability and density values of such a loop control SSA variable are computed as in the case for the branch structure with \( f = 2 \) fan-ins. However, without loss of generality, we consider the simple case, where the \( a \)-function assignment results in a merge of the two sets of value ranges into a single value range set \( \{1.0[L : U : S] \} \).

**Example 5:** Consider the component SSA variable \( X_1 \), which is the destination variable of a \( a \)-function assignment at the top of the loop \( X \) controls. The first fan-in to the loop contributes a value range of \( 1.0[0:0:0] \) corresponding to \( X_0 \), while the second fan-in contributes a value range of \( 1.0[1:10:1] \) from \( X_6 \) along the backward edge of the loop. The two ranges are combined during the assignment to yield a value range of \( 1.0[0:10:1] \) for \( X_1 \). The controllability of \( X_1 \) over this value range is then computed as

\[
C_{X_1} = \frac{1}{1/(W \times (P_{loop})^{STX_1})}/2 = 1.784
\]

and the corresponding density value is computed as

\[
D_{X_1} = [ST_{X_1} + 1]/T = 11/16 = 0.6875, \quad \text{where} \quad ST_{X_1} = [U - L]/S = 10.
\]

**B Observability analysis**

The observability \( O \) of a variable, \( V \), is a measure of the ease of propagating its values to a primary output. If a component SSA variable \( z_k \) of a primary output \( Z \) is the destination of an assignment using a component SSA variable \( v_j \) of \( V \), then the observability of \( V \) as a result of the assignment, is computed as

\[
O_V = \frac{1}{P}. \quad (23)
\]

where \( P \) is the probability of the assignment to \( z_k \).

To begin computation of observability measures, the observability values for all non-output variables are initially set to infinity, and the zero for all primary outputs.

**Case 1:** Fan-outs of a branch structure: Consider a branch with \( f \) fan-outs, each with a probability of \( P_k \), \( k = 1, 2, 3, ..., f \). Here the observability of variable \( V \) is

\[
O_V = 1/\max(P_k), \quad (24)
\]

where the maximum of the fanout probabilities is considered over all branch fan-outs.

**Case 2:** Inside a loop structure: Consider a loop with a probability \( P_{loop} \) of staying in the loop. The observability of the variable \( V \) is computed as

\[
O_V = 1/P_{loop}. \quad (25)
\]

**Case 3:** In a basic block: If the assignment of a component SSA variable \( v_i \) is in a basic block with \( q \) in-edges (each with a probability of \( P_q \)), the observability of the corresponding variable \( V \) is

\[
O_V = 1/\max(P_q). \quad (26)
\]

If one of the in-edges to the block is from a loop structure of \( iter \) iterations, then the value of \( P_q \) corresponding to this in-edge is computed as \( (1 - P_{loop})^{iter} \). The observability values for each of its \( n \) component SSA variables, \( O^i_V \), \( i = 1, 2, ..., n \), are calculated, and the final observability value of \( V \) is computed as

\[
O_V = \min(O^i_V). \quad (27)
\]

**Example 6:** In the SSA representation of Figure 2, because none of the component SSA variables of \( X \) or LOOPING is assigned to a component SSA variable of the primary output \( Z \), \( O_X = O_{LOOPING} = \infty \). On the other hand, since \( Y_3 \) is assigned to component \( Z_1 \) of the primary output \( Z \) and this assignment occurs inside the loop \( X \leq 9 \), which has a loop probability \( P_{loop} = 0.91 \), the observability of \( Y \) as a result of this assignment is

\[
O_Y = \frac{1}{P_{loop}} = 1.099.
\]

Also, this is the only assignment of a component SSA variable of \( Y \) to a primary output SSA variable. Hence the final observability value of \( Y \) is also \( O_Y = 1.099 \).

**C Proposed testability enhancement**

After the controllability, observability, and density measures have been obtained for all of the SSA components of
each variable $V$ in the behavioral description, the component SSA variables are ranked in the decreasing order of their densities. The highest density measure $D_{v_j}$ and the corresponding component SSA variable $v_j$ are noted. The density $D_V$ of the variable $V$ is computed as

$$D_V = \sum_{k=0}^{j} D_{v_k}$$

(28)

where $v_k (0 \leq k \leq j)$ is a SSA component variable that is not used in a $\sigma$-function assignment and whose set of value ranges can contribute at least one unique value to $D_V$. We assume non-overlapping of each SSA component’s value ranges; however, Equation 28 can be adjusted to take overlapping ranges into account. The controllability $C_V$ of the variable $V$ is then computed as

$$C_V = \sum_{i=0}^{j} C_{v_i}$$

(29)

where $v_i (0 \leq i \leq j)$, is a SSA component variable that is not used in a $\sigma$-function assignment. Thus, for example in Figure 4, the SSA variable $Y_3$ has the highest density of 0.323. Hence $j = 3$ in the Equations 28 and 29. Since the variables $Y_1$ and $Y_2$ are already used in the $\sigma$-function assignment to $Y_3$, the controllability $C_Y$ is computed as

$$C_Y = C_{Y_0} + C_{Y_3} = 1.0 + 2.521 = 3.521$$

For the purpose of computing $D_Y$, the only SSA component variable considered is $Y_3$, since the value range of $Y_0$ is a subset of the value range of $Y_3$. Hence the value of $D_Y$ is $D_Y = D_{Y_3} = 0.323$.

The SSA variable $X_1$ has the highest density 0.6875; thus, $j = 1$ for Equations 28 and 29. However, only $X_1$ is used for computing the density and controllability of $X$, since $X_0$ is used in the $\sigma$-function assignment to $X_1$. The controllability and density values for $X$ are computed to be 2.784 and 0.6875, respectively. Similarly, controllability of $LOOPING$ is 2.8710 and the corresponding density is 1.

Ties in highest density between SSA component variables of $V$ are broken by the highest value of $D_V$ and the lowest value of $C_V$ computed from Equations 28 and 29 respectively, using each of the tied SSA component variables.

The variables are then ranked in the decreasing order of their controllability to density ratios $CDR_V$, with the hardest to control variables (having the highest ratios) first. The controllability-to-density ratios of $LOOPING$, $Y$ and $X$ are 2.8710, 10.90 and 4.05, respectively. A similar ranking is obtained for decreasing order of observability values, with the hardest to observe variables (having the highest observability) first. Such a ranking for the internal variables gives $X$, $LOOPING$, $Y$ as the order.

If a variable $V$ has a controllability to density ratio of $CDR_V$ and a corresponding observability value of $O_V$, the testability of the variable is computed as $T_V = 0.8CDR_V + 0.2O_V$. Using this equation, the testability values of $X$, $Y$ and $LOOPING$ are $\infty$, 8.72 and $\infty$, respectively.

We propose three testability enhancements schemes based on the ranking of variables. If a variable is difficult to control but easy to observe, it is considered for either overloading or test-point insertion into the high-level control flow. For all other cases, partial-scan for the registers corresponding to these variables is proposed. The proposed technique is a greedy algorithm that tries to minimize the number of variables in the behavioral description that are difficult to control and observe. Area-performance considerations would then determine the number of such variables that should be chosen, as well as the choice of enhancement technique. Based on the above proposal, $X$ is a good candidate for partial-scan, while the controllability of $Y$ could be improved by inserting a test-point at the branch condition $X \leq 1$.

IV Experimental Results

Experiments were conducted to demonstrate the effectiveness of the proposed DFT framework. High-level synthesis HLSynth92 [18] and HLSynth95 [19] benchmarks GCD, Barcode, Diffeq and DHRC were used in these experiments. All circuit descriptions are in VHDL; logic synthesis and test generation for each circuit were performed on an HP C200 workstation with 64 MB of RAM. The proposed technique was applied to evaluate and enhance the testability in each high-level circuit description. Then, test pin insertion and overloading of selected variables are applied via modification of the VHDL codes. Partial scan for the selected variables is applied by converting the corresponding registers of the hard-to-control and hard-to-observe variables. Gate-level implementations of all circuits were synthesized using a commercial logic synthesis system; the synthesis directives were set to minimize area and critical path delay for all versions of each circuit. Different combinations of these three enhancement techniques were implemented to each benchmark circuit.

Table 2 shows the area and delay of the original and various testability-enhanced benchmark circuits. For instance, GCD is the original benchmark circuit for the greatest-common-divider circuit with no testability enhancements. In this circuit, the estimated area is 1348 units of standard inverter area; its critical-path delay (CP Delay) is 69.629 ns; finally, 1304 library primitives (Prims) are used to synthesize this circuit. GCD.pscan uses partial-scan for two internal variables $xvar$ and $yvar$ selected by our behavioral-level testability analysis. Next, GCD.t1.pscan uses one test-pin insertion and partial scan for one internal variable $yvar$. The final version, GCD.t2, is GCD with two test pins inserted in the loop; specifically, the first test pin is AND-ed with the loop condition while the second test pin is OR-ed with the result. GCD.t2 is similar to the two-test-pin approach proposed in [5], except that in our technique, the variable selection and test-pin insertion is based on value-range informa-
tion, rather than by inspection. The additional area overhead for GCD\textsubscript{t2} is only 1 unit of standard inverter area, and its delay is increased by 3.57 ns. The different versions for the other benchmark circuits can be interpreted in a similar manner, where Barcode\textsubscript{t2}, Diff eq\textsubscript{t2}, and DHRC\textsubscript{t2} all have two test pins added to the selected internal variables. The circuit DHRC\textsubscript{t2mod} also uses two test pins, but these two points are inserted differently from DHRC\textsubscript{t2} due to our dataflow analysis. The circuit Diff eq\textsubscript{t3} has three test pins. Note that the area overhead in all testability-enhanced circuits is very small, except for Barcode\textsubscript{t2}, where an additional 925.6 units of the standard inverter area was needed.

The ATPG results for all circuits using a commercial ATPG are reported in Table 3. The backtrack limit for all versions of each benchmark was set to 2000. In this table, the total number of collapsed faults is first given for each circuit, followed by the number of detected faults, number of untestable faults, size of the test set, fault coverage, test efficiency, ATPG execution time in seconds, and test application time (TAT) measured in number of clock cycles necessary to apply the test. Consider the circuit GCD without any testability enhancements, the fault coverage obtained was 84.88% with 1023 test vectors, and the ATPG time was 8261 seconds. When we merely partial-scan the two variables that were hardest to test, (arriving at the circuit GCD\textsubscript{pscan}), fault coverage increases 10.26% to 95.14%, but at the cost of greater test application time (12192 cycles) due to scan in and out of vectors. In the third case (GCD\textsubscript{t1}\textsubscript{pscan}), after using one test pin for one internal variable and partial-scan another internal variable that was hard to both control and observe, the fault coverage increased to 96.83%, while the test size decreased from 1023 to 640 vectors and the ATPG time also decreased to 1503 seconds. Test application is also less than GCD\textsubscript{pscan} since we need only scan one register.

Finally, using two test pins on selected internal variables for GCD\textsubscript{t2}, the fault coverage increased to 98.23%, the test size decreased to 443 vectors, the ATPG time also decreased to only 989 seconds, and test application time is the lowest. The two test pins in this case were inserted the same way as in [5]; however, due to different commercial synthesis and ATPG systems, different fault coverages, test set sizes, and ATPG times were obtained; likewise, the discrepancies due to different commercial ATPGs were observed for other circuits as well.

In Barcode, with two test pins inserted, fault coverage increased from 80.91% to 87.70%. Further improvements were achieved in Barcode\textsubscript{t2}\textsubscript{pscan} and Barcode\textsubscript{pscan} by partial-scanning selected variables. ATPG results for Diff eq are already very good; thus the added testability enhancements did not improve coverages much. Finally, in DHRC, when we select the 2 test-pin insertion, we insert them at different places than that in [5], resulting in a different circuit, DHRC\textsubscript{t2mod}. Note that with this new insertion, the coverages are better than DHRC\textsubscript{t2}. Partial-scanning the hard-to-test variables in DHRC\textsubscript{pscan} results in best coverage, since it gives us direct improvement in both controllability and observability over these variables, but at a cost of 24864 test cycles. Overall, the designer and test engineer decide whether to partial-scan the identified hard-to-test variables, overload the variables, or insert test-pins, using the results from the proposed testability analysis.

V Conclusions
A high-level DFT framework based on formal analyses of weighted value range propagation of variables in CDFG’s has been proposed. In this framework, a SSA representation is first extracted from a VHDL behavioral description, followed by analyses of variables’ value ranges. Controllability, observability, and density values for each internal variable are then computed using the value ranges

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Test Pins</th>
<th>Scan DFFs</th>
<th>Overloading DFFs</th>
<th>Est. Area</th>
<th>Est. CP Delay</th>
<th>Prims.</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCD</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1318.00</td>
<td>69.629</td>
<td>1304</td>
</tr>
<tr>
<td>GCD\textsubscript{pscan}</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>1418.60</td>
<td>70.410</td>
<td>1490</td>
</tr>
<tr>
<td>GCD\textsubscript{t1}\textsubscript{pscan}</td>
<td>1</td>
<td>16</td>
<td>0</td>
<td>1391.40</td>
<td>72.351</td>
<td>1456</td>
</tr>
<tr>
<td>GCD\textsubscript{t2}†</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1349.00</td>
<td>73.200</td>
<td>1301</td>
</tr>
<tr>
<td>Barcode</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>895.30</td>
<td>25.707</td>
<td>882</td>
</tr>
<tr>
<td>Barcode\textsubscript{t2}†</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>1951.40</td>
<td>48.525</td>
<td>1571</td>
</tr>
<tr>
<td>Barcode\textsubscript{t2}\textsubscript{pscan}</td>
<td>2</td>
<td>8</td>
<td>16</td>
<td>1848.50</td>
<td>49.081</td>
<td>1584</td>
</tr>
<tr>
<td>Barcode\textsubscript{pscan}</td>
<td>0</td>
<td>24</td>
<td>0</td>
<td>1080.80</td>
<td>25.745</td>
<td>1051</td>
</tr>
<tr>
<td>Diff eq</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27033.10</td>
<td>162.656</td>
<td>17158</td>
</tr>
<tr>
<td>Diff eq\textsubscript{t2}†</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27036.70</td>
<td>165.846</td>
<td>17161</td>
</tr>
<tr>
<td>Diff eq\textsubscript{t3}</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>27041.20</td>
<td>165.857</td>
<td>17163</td>
</tr>
<tr>
<td>DHRC</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7458.50</td>
<td>54.191</td>
<td>6966</td>
</tr>
<tr>
<td>DHRC\textsubscript{t2}†</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>8106.00</td>
<td>58.630</td>
<td>7743</td>
</tr>
<tr>
<td>DHRC\textsubscript{t2mod}</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>7573.50</td>
<td>53.075</td>
<td>6936</td>
</tr>
<tr>
<td>DHRC\textsubscript{pscan}</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>7511.60</td>
<td>54.231</td>
<td>7635</td>
</tr>
</tbody>
</table>

†Testability-enhanced circuits similar to [5]  
Est. CP Delay: Estimated Critical-Path Delay in nanoseconds  
Est. Area: Estimated area in units of std. inverter area  
Prims.: Number of library primitives used
Table 3: ATPG Results For Proposed Technique

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Total Faults</th>
<th>Det. Faults</th>
<th>Unt. Faults</th>
<th>Test Vectors</th>
<th>Fault Cov%</th>
<th>ATPG Eff%</th>
<th>ATPG Time</th>
<th>TAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCD</td>
<td>2033</td>
<td>1726</td>
<td>108</td>
<td>1023</td>
<td>84.88</td>
<td>86.42</td>
<td>2.29h</td>
<td>1023</td>
</tr>
<tr>
<td>GCD_pscan</td>
<td>2103</td>
<td>2001</td>
<td>48</td>
<td>381</td>
<td>95.14</td>
<td>95.32</td>
<td>6.9m</td>
<td>12192</td>
</tr>
<tr>
<td>GCD_f1_pscan</td>
<td>2065</td>
<td>1999</td>
<td>77</td>
<td>640</td>
<td>98.63</td>
<td>97.31</td>
<td>25.1m</td>
<td>10240</td>
</tr>
<tr>
<td>GCD_t2</td>
<td>1993</td>
<td>1859</td>
<td>105</td>
<td>443</td>
<td>98.23</td>
<td>98.65</td>
<td>16.5m</td>
<td>443</td>
</tr>
<tr>
<td>Barcode</td>
<td>1301</td>
<td>1053</td>
<td>99</td>
<td>1713</td>
<td>80.91</td>
<td>85.66</td>
<td>25m</td>
<td>1713</td>
</tr>
<tr>
<td>Barcode_t2</td>
<td>3334</td>
<td>2924</td>
<td>309</td>
<td>3004</td>
<td>87.70</td>
<td>89.10</td>
<td>39.9m</td>
<td>3004</td>
</tr>
<tr>
<td>Barcode_t2_pscan</td>
<td>3050</td>
<td>2776</td>
<td>291</td>
<td>2765</td>
<td>91.01</td>
<td>92.11</td>
<td>27.2m</td>
<td>22120</td>
</tr>
<tr>
<td>Barcode_pscan</td>
<td>1375</td>
<td>1281</td>
<td>67</td>
<td>333</td>
<td>93.16</td>
<td>94.22</td>
<td>9.1m</td>
<td>7992</td>
</tr>
<tr>
<td>DiffEq</td>
<td>59510</td>
<td>58379</td>
<td>282</td>
<td>214</td>
<td>98.10</td>
<td>99.53</td>
<td>14.2m</td>
<td>214</td>
</tr>
<tr>
<td>DiffEq_t2</td>
<td>59526</td>
<td>58532</td>
<td>270</td>
<td>334</td>
<td>98.33</td>
<td>99.55</td>
<td>13.9m</td>
<td>334</td>
</tr>
<tr>
<td>DiffEq_t3</td>
<td>59540</td>
<td>58656</td>
<td>265</td>
<td>616</td>
<td>98.77</td>
<td>99.99</td>
<td>11.4m</td>
<td>616</td>
</tr>
<tr>
<td>DHRC</td>
<td>11192</td>
<td>9296</td>
<td>524</td>
<td>845</td>
<td>83.06</td>
<td>91.93</td>
<td>5.5h</td>
<td>845</td>
</tr>
<tr>
<td>DHRC_t2</td>
<td>12409</td>
<td>10867</td>
<td>974</td>
<td>2917</td>
<td>87.57</td>
<td>94.44</td>
<td>4.4h</td>
<td>2917</td>
</tr>
<tr>
<td>DHRC_t2mod</td>
<td>11621</td>
<td>10418</td>
<td>677</td>
<td>2217</td>
<td>89.65</td>
<td>95.14</td>
<td>3.73h</td>
<td>2217</td>
</tr>
<tr>
<td>DHRC_pscan</td>
<td>11218</td>
<td>10657</td>
<td>423</td>
<td>3108</td>
<td>95.00</td>
<td>96.02</td>
<td>3.2h</td>
<td>24864</td>
</tr>
</tbody>
</table>

†Testability-enhanced circuits similar to [5]

TAT: Test Application Time in number of clock cycles

obtained. Our proposed technique results in accurate measures of variable testability, and different combinations of testability enhancements can be applied based on the testability evaluation results. The effectiveness of our approach is demonstrated by the experimental results; in particular, fault coverages and test efficiencies have been increased significantly with ATPG execution times reduced for the benchmark circuits studied, while area and performance overheads are kept at a minimum.

References