An RTL Methodology to Enable Low Overhead Combinational Testing

Subhrajit Bhattacharya Sujit Dey Bhaskar Sengupta
C&C Research Laboratories, NEC USA
Princeton, NJ 08540
{sb,dey,bhaskar}@ccrl.nj.nec.com

I. INTRODUCTION

Combinational test pattern generation remains a popular testing methodology because of its ability to generate test patterns for large industrial circuits with high test efficiency. Full-scan [1] is the conventional methodology which enables the testing of sequential circuits using test patterns generated by combinational ATPG. However, application of combinational testing to sequential circuits using full-scan methodology may be prohibitively expensive, due to the high area overhead and test application time associated with full-scan.

Several gate level techniques like partial scan have been proposed to reduce the area overhead of full scan implementations. However, to avoid requiring substantially less scan FF's than full-scan, partial scan requires an efficient sequential ATPG tool. Recently, a gate-level technique has been proposed to reduce the area overhead of scan design by establishing paths in the scan chain using existing logic [2] instead of using scan FFs. The technique requires setting the primary inputs to establish the free scan paths, thus limiting its applicability.

Reduction of test application time has been addressed in several ways, like arranging scan flip-flops in parallel scan chains [3, 4], and reconfiguring scan chains [5]. In the parallel scan chain approach, the number of parallel scan chains, and hence the number of vectors that can be shifted in parallel, is limited by the minimum of the number of primary inputs and primary outputs of the circuit. The reconfigurable scan chain approach [5] is limited by the ability of the circuit to be decomposed into a set of kernels, which are disjoint portions of logic that can be tested independently.

Recently, several RT-level and behavioral level design and synthesis-for-testability approaches have been proposed to generate easily testable circuits for partial scan, sequential ATPG, and BIST testing methodologies [6]. The proposed approaches include RT-level scan selection [7, 8], modification of the behavioral description of a design to improve the testability of the synthesized circuit [9, 10], and considering testability during the behavioral synthesis process [11, 12, 13, 14, 15]. The high-level techniques are mostly applicable to data-flow/arithmetic intensive designs, like DSP filters and processor designs, characterized by the dominance of functional units in the data paths. Recently, an RT level technique was proposed which enables circuit testing using combinational test patterns [16]. The methodology uses existing paths between registers, going through multiplexors, to load a combinational test pattern into the circuit FFs without actually having to use scan FFs.

A technique was proposed later [17] that can also use existing paths through functional units. However, appropriate constants (identity elements) need to be added to the side inputs of the units to create I-paths [18]. Also, the paths in the test mode cannot contain cycles or fanout, constraints that are eliminated in the method proposed here.

This paper introduces RT-SCAN, a register transfer (RT) level test synthesis technique which enables the testing of complex sequential circuits by combinational test patterns without the use of scan, eliminating the high overheads associated with full-scan testing. The methodology involves both DFT to prepare the sequential circuit for combinational testing, as well as synthesizing the necessary sequential circuit input patterns from the given combinational test patterns. Unlike the existing high-level methods, the new approach is applicable to any general design, including control-flow intensive designs, where the control-flow and hence random logic, multiplexors, counters (incrementers), buses, and registers, dominate the circuit, with the existence of a few arithmetic units.

Utilizing the RT-level description of a circuit, we introduce alternatives to traditional scan chain structures for scanning in and out patterns into the circuit FFs. The alternative structures allow the FFs to be connected using interconnects and components that already exist in the RT level circuit, such as multiplexors, adders, subtracters, incrementers, multipliers, thus minimizing the test area overhead. Also, the proposed structures need not necessarily be acyclic. Finally, the structures allow parallel loading of FFs, thus reducing test application time. The technique also involves merging of incrementers which frequently appear in control-flow intensive designs in order to further reduce the test area overhead.

The proposed RT-SCAN methodology has been applied to several RT-level designs. To verify the advantages of RT-SCAN, we performed sequential testing, combinational testing using full scan, and combinational testing using RT-SCAN. As expected, sequential test pattern generation faired poorly, producing test efficiency as low as 46%, while combinational testing using full-scan could produce test efficiency of 100% for all the circuits, though with high area and test application time overheads. Experimental results demonstrate the ability of the RT-SCAN methodology to achieve 100% test efficiency using combinational test patterns, with significantly reduced area overhead and test application time compared to conventional full scan implementations, making combinational testing an economically viable testing methodology.

Section III introduces the RT-SCAN methodology. The alternative test structures which can be used for loading combinational patterns into circuit FFs are explained. The test structures can utilize existing RTL components such as adders and subtracters, and can also have loops. The advantages of using such test structures in minimizing the test area overheads and test application times are demonstrated. In Section IV, it is shown that merging incrementers can introduce additional paths between registers, further minimizing test area overheads. Implementation details of the RT-SCAN methodology and experimental results are discussed in Section V followed by conclu-

ED&TC '97 on CD-ROM

sions in Section VI.

II. Overheads of Full Scan

Traditional scan implementations incur high area overhead due to the extra test circuitry required, and require a large test application time. Consider the RT level circuit shown in Figure 1(a) which is part of the implementation of an X.25 protocol. The circuit has four 8 bit registers R1, R2, R3 and OUT for a total of 32 FFs, an 8 bit input I1, and the output of register OUT is a primary output.

In a full scan implementation of the circuit of Figure 1(a), each of the 32 FFs in the circuit have to be replaced by a scan FF, consisting of the functionality of a regular FF with a multiplexor at the input. In addition, there is extra routing required to connect the scan chain, and to connect the test pin to the 32 multiplexors in the scan FFs. In an actual scan implementation, buffers have to be added between the test pin and the multiplexor inputs because of the high fanout of the test signal. As can be seen from the results in Table 3, significant area overheads are associated with full scan implementations of the benchmarks circuits.

The second drawback of the scan scheme is the high test application time associated with it. Assume that a combinational test pattern generator generates 100 patterns to test all the faults in the combinational part of the circuit of Figure 1(a). Since there are 32 FFs in the circuit, 32 clock cycles are required to load each pattern into the FFs, followed by 1 clock cycle to apply the test pattern. Finally, another 32 clock cycles are required to flush out the FFs after the last test pattern has been applied. Thus, testing the original circuit with a traditional full scan implementation would require $(100 \times 33) + 32 = 3332$ clock cycles. Testing circuits with several thousand FFs may require unacceptable number of clock cycles.

III. Using Existing RT Level Paths for Reducing DFT Overheads

In this section, we introduce RT-SCAN as a RT Level DFT methodology which enables the testing of complex sequential circuits by combinational patterns without the high area overhead and long test application time of full scan implementations. To achieve the above goals, RT-SCAN maximally reuses existing parallel paths connecting the registers, inserting new paths between registers only if necessary.

The first step of the RT-SCAN methodology consists of extracting existing RTL paths which may be used for loading circuit registers with any desired pattern, and also for observing the contents of the circuit registers. The graph of such existing paths is called the connectivity graph and is explained in Section A. In the second step, the RT-SCAN algorithm identifies the set of paths which will be used for loading/observing the circuit registers, maximally reusing the paths of the connectivity graph in the process. The RT-SCAN graph and the corresponding RT-SCAN implementation is explained in Section B. Generating the sequence of primary input vectors for applying a combinational test pattern to the RT-SCAN implementation is explained in Section C. Finally, the DFT overhead reduction by using RT-SCAN instead of full scan is illustrated in Section D.

A. The Connectivity Graph of an RTL Circuit

Reconsider the RTL circuit of Figure 1(a). Several parallel paths connecting registers through multiplexors and the ALU unit already exist in the circuit, such as the path from the input I1 to the register R1, and the paths from the registers R1 and R2 to register OUT through the ALU unit. Such paths can be utilized for the purpose of scanning in and out a test pattern without paying the extra cost associated with using scan FFs for the purpose. The RT-SCAN methodology maximally reuses such existing parallel paths in creating the RT-SCAN implementation for reducing area and test application time overheads. The RTL circuit registers and the subset of the RTL circuit paths that are considered by RT-SCAN for creating the RT-SCAN implementation are captured by the connectivity graph of the circuit. We next formally define the connectivity graph of a circuit.

Definition 1 Connectivity Graph: *For every primary input, primary output, register, constant, and functional (FU) unit of the RTL circuit, there is a node in the corresponding connectivity graph. Connectivity graph edges connect distinct nodes if there is a path between the two corresponding nodes in the RTL circuit which goes through only multiplexors.*

Consider again the RTL circuit of Figure 1(a) and its connectivity graph shown in Figure 1(b). The connectivity graph not only includes the nodes for the registers of the circuit of Figure 1(a), but it also includes a node for the adder unit. All edges between nodes of the the connectivity graph correspond to paths through only multiplexors in the RTL circuit. However, since the connectivity graph can have nodes corresponding to functional units, there can be a path between two register nodes in the connectivity graph which goes through a functional unit node. For example, the path from register nodes R1 and R2 to register OUT goes through the adder node in the connectivity graph of Figure 1(b). Also, since there is an edge between two nodes of the connectivity graph only if the two nodes are distinct, none of the self-loops around the registers of the circuit of Figure 1(a) are present in its connectivity graph.

B. The RT-SCAN Graph

The RT-SCAN graph defines the set of paths which will be used for loading/observing the every register in the circuit. The goal of the RT-SCAN algorithm is to create an RT-SCAN graph from the connectivity graph of a circuit by maximally reusing the paths of the connectivity graph to reduce the overhead of DFT. The RT-SCAN graph and the corresponding RT-SCAN implementation is explained in this Section.

Definition 2 RT-SCAN Graph: *An RT-SCAN graph has a node for every register node of the corresponding connectivity graph. The RT-SCAN graph may also contain other nodes of the connectivity graph such as primary inputs, primary outputs and FU nodes. The edges in an RT-SCAN graph satisfy the following properties:*

1. *There is only one edge incident to a node of type register, primary output, and single input FUs such as incrementers. There are two edges incident to a node which corresponds to a two input FU such as an adder or multiplier.*
2. *For every register node, there is a path from an input node to the register node.*
3. *For every register node, there is a path from the register node to an output node.*
4. *There can be a fanout edge from a node if at most one of the fanout edges go to a node which is not an output node.*

The RT-SCAN graph of Figure 1(c) has all the edges of the connectivity graph of Figure 1(b) except for the edges from R1 to OUT

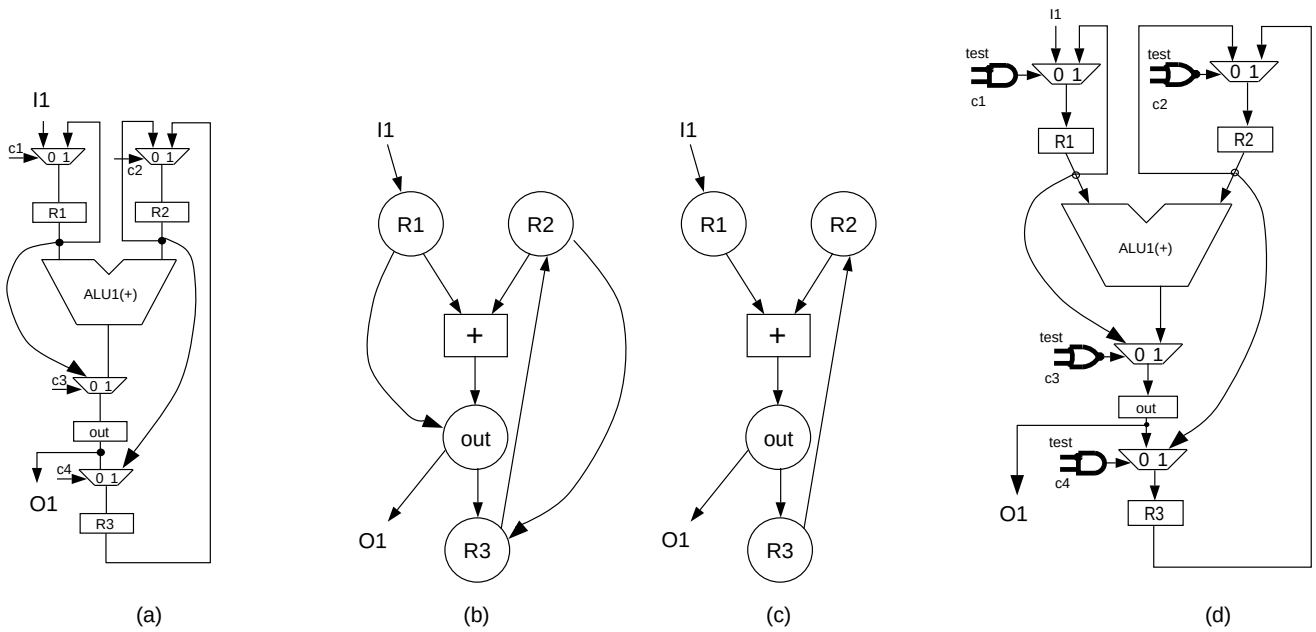

Figure 1: (a) A sub circuit of the X25 circuit and its (b) connectivity graph, (c) RT-SCAN graph, and (b) final RT-SCAN implementation.

and from R2 to R3. However, it may be necessary to have edges in the RT-SCAN graph which did not exist in the connectivity graph, as in the example of Section IV. The circuit shown in Figure 1(a) and its RT-SCAN graph of Figure 1(c) is used to produce the RT-SCAN implementation shown in Figure 1(d). It can be seen that when `test` is low, the RT-SCAN implementation of Figure 1(d) behaves as the original circuit in Figure 1(a), and when the `test` input is high, the paths of the RT-SCAN implementation which are exercised correspond to the RT-SCAN graph of Figure 1(c).

To maximally reuse existing paths in the RTL circuit in the RT-SCAN implementation we allow the use of paths containing adders and loops. For instance, the RT-SCAN graph of Figure 1(c) has a path from registers R1 and R2 to register OUT through an adder node. Also note that the RT-SCAN graph path $R2 \rightarrow OUT \rightarrow R3 \rightarrow R2$ forms a loop. Thus, in the test mode, the paths of the RT-SCAN implementation which are exercised will also have adders and loops. As shown subsequently, even though the RT-SCAN paths exercised in the test mode may have adders and loops, the registers can be loaded with any test pattern, and also the contents of the registers are observable at primary outputs.

Let the length of a path in the RT-SCAN graph be the number of registers on the path. Let the length of the shortest path from a primary input to the output of register X be denoted by $L(X)$. For a primary input node I, $L(I)$ is 0. Let LOADTIME be defined as $LOADTIME = max(\forall R, L(R))$.

Theorem 1 *Given the register values of an RT-SCAN implementation at clock cycle t, any combinational test pattern can be loaded into the registers and subsequently applied for testing the circuit in clock cycle $(t + LOADTIME + 1)$.*

The formal proof of the theorem can be found in [19]. However, in Section C we illustrate the process of loading a test pattern into the registers of an RT-SCAN implementation.

When testing a circuit, it is not only important to be able to load a particular vector into the circuit FFs, but it is also necessary to be able to observe the result of a test application captured in the FFs to detect the presence of faults. It can be shown that if only one register of an RT-SCAN graph captures a fault effect, the effect can always be observed at an output node. If the fault effect is captured in more than one register, then there is a possibility of fault masking and consequently, in such cases it will not be possible to detect the fault effect at the output. However, as can be seen from Table 3, the RT-SCAN test sets achieved close to 100% test efficiency for all four circuits tested. Thus, the RT-SCAN implementations do not seem to suffer significantly from fault masking effects.

C. Loading Combinational Patterns into Registers

In this section, the process of generating the sequence of primary input vectors for applying a combinational test pattern to the RT-SCAN implementation is explained. To explain how a test pattern can be loaded into the registers of the RT-SCAN graph of Figure 1(c), we first introduce the assignment statements corresponding to an RT-SCAN graph. The values of the register nodes of the RT-SCAN graph at any given time t can be expressed in terms of the values of the register nodes and input nodes at time $(t-1)$. The assignment expressions describing the values of the registers of the RT-SCAN graph of Figure 1(c) are given in Figure 2. For example, the last assignment expression of Figure 2 implies that the value of the register node OUT at time t is a sum of the values of the register nodes R1 and R2 at time $(t-1)$.

Let us assume that the pattern in the fifth row of Table 1 has to be loaded into the register nodes of the RT-SCAN graph of Figure 1(c). We assume that each node of the RT-SCAN graph of Figure 1(c) has a bitwidth of 4 for ease of illustration. Also, assume that the values of the registers in clock cycle 1 are known, and are given by the first row of Table 1. The above assumption is reasonable, since most of the circuits encountered in practice have resettable FFs. We next compute the values to be applied to the input I1 to load the registers of the

$$
\begin{aligned}
R1(t) &\leftarrow I1(t-1) \\
R2(t) &\leftarrow R3(t-1) \\
R3(t) &\leftarrow OUT(t-1) \\
OUT(t) &\leftarrow R1(t-1) + R2(t-1)
\end{aligned}
$$

Figure 2: Equation describing the RT-SCAN graph of Figure 1(e).

Clk	R1	R2	R3	OUT	I1
1	0000 (S)	0000 (S)	0000 (S)	0000 (S)	1010 (B)
2	1010 (B)	0000 (F)	0000 (F)	0000 (F)	0111 (B)
3	0111 (B)	0000 (F)	0000 (F)	1010 (B)	1001 (B)
4	1001 (B)	0000 (F)	1010 (B)	0111 (B)	1111 (B)
5	1111 (S)	1010 (S)	0111 (S)	1001 (S)	0010 (S)

Table 1: Illustration of the synthesis of sequential test patterns from combinational test patterns.

RT-SCAN graph with the test pattern given in the 5th row of Table 1.

Forward Simulation. The first step in determining the input values consist of a forward simulation step. During the forward simulation step, the values of registers at time step $t > 1$ which depend only on the initial values of the registers at time $t = 1$ are calculated. Thus, the forward simulation process begins with the second clock cycle. Consider the first assignment statement of Figure 2 for the RT-SCAN graph of Figure 1(c), $R1(t) \leftarrow I1(t-1)$. According to the statement, the value of `R1` at time 2 depends on the value of the input `I1` at time 1, and not solely on the register values at time 1. Hence, `R1`'s value at time 2 is not calculated during forward simulation. Consider the second equation, $R2(t) \leftarrow R3(t-1)$. Since the value of `R3` is known at time 1 and is "0000", the value of `R2` can be calculated at time 2 and is also "0000". Similarly, the value of `OUT` at time 2 can be calculated as the sum of the values of `R1` and `R2` at time 1. Since in clock cycle 1, `R1` and `R2` have values "0000" and "0000", the value of `OUT` at time 2 is "0000". In a similar fashion, every equation describing the RT-SCAN graph is used to calculate the values of registers which depend solely on the values of the registers in clock cycle 1. After the simulation has been completed for the 2nd clock cycle, it is repeated for the 3rd clock cycle using the values of the registers in the 2nd clock cycle which were computed in the previous step. The forward simulation step is repeated one more time for clock cycle 4. The values computed during the forward simulation process are shown in Table 1 with a "(F)" tag next to them.

Backward Simulation. The second step of test synthesis consists of a backward simulation which starts at clock cycle 4. During the backward simulation step at time 4, values are assigned to the nodes of the RT-SCAN graph in clock cycle 4, based on the value of the nodes in clock cycle 5. Consider for example the assignment expression $R1(t) \leftarrow I1(t-1)$. Since `R1` has a value of "1111" in clock cycle 5, `I1` is assigned a value of "1111" in the 4th clock cycle. Next consider the assignment expression $OUT(t) \leftarrow R1(t-1) + R2(t-1)$. The value of `OUT` at time 5 is "1001". Also, the value of `R2` in the 4th clock cycle depends solely upon the initial register values and was calculated during the forward simulation step to be "0000". Thus, the value of `R1` during the 4th clock cycle must be "1001" since "1001" and "0000" add up to "1001". The backward simulation process starts at clock cycle 4 and is repeated for clock cycles 3, 2 and 1. At the end of the backward simulation process, all the input values required to apply the test pattern given by the 5th row of Table 1 have been computed. The values computed during the backward simulation step are shown with the tag "(B)" next to the values. As stated in Theorem 1, it can be shown that any test pattern can be loaded into the registers of an RT-SCAN graph in a fixed number of clock cycles, if the initial values of the registers are known.

D. Overhead of the RT-SCAN implementation

Since for the circuit of Figure 1(a) every edge of the RT-SCAN graph (Figure 1(c)) also exists in the connectivity graph (Figure 1(b)), it implies that for every edge in the RT-SCAN graph, a path through only multiplexors already exists in the circuit of Figure 1(a). However, to connect the registers using these existing paths in the test mode, a `test` input and four gates are added to the circuit of Figure 1(a). The resulting RT-SCAN implementation is shown in Figure 1(d).

For the circuit of Figure 1(a), the corresponding RT-SCAN implementation shown in Figure 1(d) requires 4 extra gates only as compared to the 32 1-bit multiplexors that would be required for a full scan implementation of the circuit. Another cost associated with the above schemes concerns the extra wiring introduced by the scan implementations. Consider the path which exists from `I1` to `R1` in the original circuit of Figure 1(a). A traditional scan implementation would not only have to substitute the 8 FFs of register `R1` with scan FFs, it would have to route the input `I1` to the FFs corresponding to `R1`, introducing 8 extra interconnects. In the RT-SCAN implementation, to connect `I1` to `R1`, the test input has to be routed to only one AND gate. The output of the AND gate replaces the control signal for the mux in the original circuitry. Hence, there is no extra overhead of routing the output of the AND gate to the mux. As can be seen from Table 3, the number of extra interconnects needed by the RT-SCAN implementations can be significantly lower than that of the full scan implementations.

Also, since the edges in the RT-SCAN graph of Figure 1(c) connects 8 bit registers and correspond to 8 parallel paths, the test application time of the RT-SCAN implementation is reduced by a factor of 8, and would be 504 clock cycles as compared to 3332 for full scan.

IV. Merging Incrementers to Improve Register Connectivity

A control-flow intensive design is characterized by the presence of several counter variables which regulate the control flow, including the execution of loops and conditionals. Figure 3(a) shows the connectivity graph of an RTL circuit implementing a typical control-flow intensive design, a barcode reader [20]. Consider the registers which store counter variables *black*, *white*, and *actnum*. Each of the registers has the following property: the register is initialized to some value (here 0), and can store only its incremented value, produced by the associated incrementer. Each such register-incrementer block acts as a counter, the register is termed a counter register and the incrementer is termed a counter incrementer. A counter register is typically not accessible from any other register: Figure 3(a) shows that the registers black, white, and actnum do not have a path from any other register. Hence, to include counter registers in the scan chain, the RT-SCAN technique requires to add extra buses and multiplexors. The corresponding RT-SCAN graph is shown in Figure 3(b). Three extra 8-bit wide buses and multiplexors are required to include the three counter registers in the full-scan chain.

The connectivity of counter registers can be improved by merging, whenever possible, the corresponding incrementers. Consider

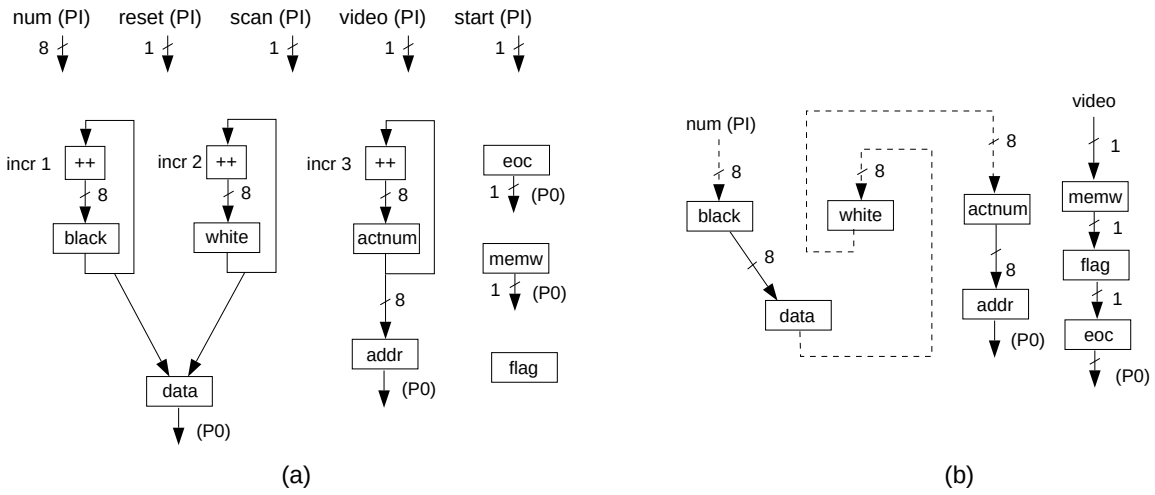

Figure 3: (a) Connectivity, and (b) RT-SCAN graph for the Barcode circuit before incrementer merging.

merging the two incrementers, incr2 and incr3, shown in Figure 3(a). The incrementers can be merged because they are never needed in the same clock cycle [19]. The resultant structure, consisting of the previous multiplexors, the two registers *white* and *actnum*, and the merged incrementer *incr23*, is shown in Figure 4(a). Though one incrementer is reduced, an extra multiplexer is needed to determine which of the register values, *white* or *actnum*, will be incremented. Since the cost of a multiplexer and an incrementer is about the same (both about 3000 grids after technology mapping to SIS library), incrementer merging would not be attractive for conventional high-level/RT-level synthesis. However, incrementer merging results in the corresponding registers to become connected to each other, as shown in Figure 4(a). Hence, though merging two incrementers does not change the area of the circuit, it helps to minimize the RT-SCAN area overhead. The connectivity graph and RT-SCAN graph of the barcode circuit after merging the two incrementers *incr2* and *incr3* are shown in Figures 4 (b) and (c) respectively. Because the counter registers *white* and *actnum* are now connected to each other, the RT-SCAN graph shows that the extra 8-bit bus and multiplexor required to connect register *actnum* previously (Figure 3(b)) is not needed after incrementer merging (Figure 4(c)).

Merging two incrementers is useful to enhance connectivity of the corresponding registers only when (a) at least one of the registers is not accessible from any other register, and (b) each incrementer feeds only its corresponding counter register. The conditions under which two incrementers can be merged, and a technique to check whether the merging condition is satisfied, can be found in [19].

V. Experimental Results

The algorithm for the proposed RT-SCAN testing scheme, comprising of (a) incrementer merging, (b) exploiting RTL paths through adders, incrementers, subtracters, multipliers and muxes for RT-SCAN graph creation, and (c) sequential test pattern synthesis have been implemented in C. Details of the algorithm can be found in [19]. In this section, we summarize the results of applying the proposed RT-SCAN testing scheme to four benchmark RT-level designs [20], implementing the GCD algorithm, a barcode reader, the `change_maker` process in a vending machine controller, a process from the x.25 protocol suite, and an industrial circuit `NECckt`. The RTL circuits were synthesized to equivalent gate-level circuits using NEC's RTL synthesis system, VARCHSYN.

	RT Level			Gate Level	
	FUs	**Muxes**	**Reg**	**Transistor Pairs**	**FFs**
GCD	6	146	5	2811	52
Barcode	8	102	8	1764	47
Change_maker	6	134	8	1910	31
x25	2	162	10	1929	60
NECckt	7	987	110	48,034	840

Table 2: Circuit Size Statistics.

The characteristics of the resultant circuits, in terms of the RT-level and gate-level components used, are shown in Table 2. For example, the GCD circuit has six functional units (column **FUs**): one 16-bit subtract unit, three 16-bit comparator units, and two 1-bit comparator units. There are 146 one bit multiplexors in the GCD circuit (column **Muxes**), and five registers (column **Reg**). The total number of FFs for all the registers is 52 (column **FFs**), while the total number of gates for GCD is 1093 (column **Gates**).

To verify the advantages of RT-SCAN, we performed sequential testing, combinational testing using full scan, and combinational testing using RT-SCAN. The results of the experiments are reported in Table 3. For each RT-level design, the row **ORIG** shows the original circuit without any scan, the row **FSCAN** shows the full-scan implementation of the circuit, and the row **RT-SCAN** shows the circuit produced by applying the proposed RT-SCAN technique to the original RT-level circuit. For each of the three versions, the test hardware used is reported, including scan FFs (SFF), multiplexors (M), gates (G), pins (P), and interconnects (I). For example, in the full-scan implementation of the GCD circuit, 105 extra interconnects are needed: 53 interconnects to thread the 52 scan FFs into a scan chain, and 52 more interconnects to route the test input to the 52 scan FFs. The three extra pins are for `scanin`, `scanout`, and `test`. The RT-SCAN implementation of the GCD circuit requires 4 1 bit multiplexors, 5 gates, and 15 interconnects. Since *RT-SCAN* uses circuit inputs and outputs for loading and unloading patterns into circuit FFs, it requires only one extra pin, the `test` pin. The column **Test Area Overhead** reports active area overhead of each test scheme in terms of the

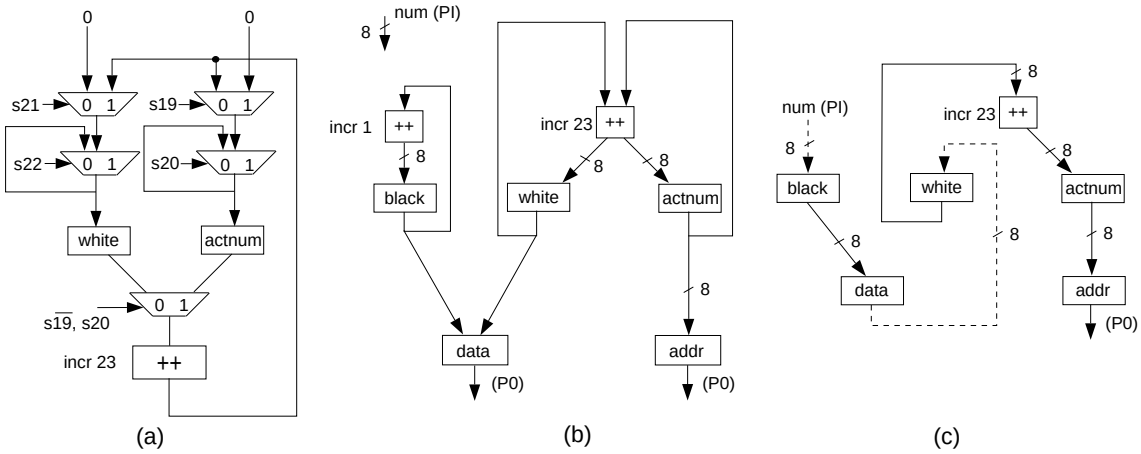

Figure 4: Barcode: (a) changed part of datapath after by incrementer merging. (b) Connectivity and (c) RT-SCAN graphs after incrementer merging.

number of extra transistor pairs required. To measure the active area overhead of the full scan implementation, the overhead of using a scan FF in place of a regular FF, and the overhead of buffers inserted due to the high fanout of the test pin were considered. The area overheads of RT-SCAN implementations were due to the extra muxes and gates required. For the GCD circuit, the full scan implementation required 565 extra transistor pairs, for an overhead of 20% while the RT-SCAN implementation required 26 extra transistor pairs for an overhead of only 1%. The overhead of full scan averaged over the four implementations was 25%. The corresponding overhead of the RT-SCAN implementations was only 4.4%.

The tables also show the results of test generation for the different test schemes. For the original circuit (without scan), test pattern generation was performed using the sequential ATPG tool, HITEC [21]. For the full scan circuit, ATPG was done by using a combinational ATPG tool. For the RT-SCAN circuit, combinational test patterns were first generated using a combinational ATPG tool, and then automatically converted to sequential test patterns as explained in Section C. Finally, the generated sequential test patterns were applied by a sequential fault simulator to measure the effectiveness of the patterns on the RT-SCAN circuit.

The columns **# of Faults**, **FC%** and **TE%** report the total number of faults in the circuit, the percentage fault coverage and the percentage test efficiency obtained for the original, full-scan, and RT-SCAN circuits by sequential ATPG, combinational ATPG, and RT-SCAN. While sequential ATPG could only achieve only 83% test efficiency for GCD, both combinational ATPG and RT-SCAN obtained 100% test efficiency. Note that the RT-SCAN method could achieve comparable test efficiency as the traditional full-scan scheme, but the RT-SCAN implementation required significantly less area overhead than the full-scan implementation.

The last column reports the reports the number of clock cycles needed to apply the test vectors, also called the test application time or **TAT**. For the GCD example, testing the full scan implementation will require 18008 clock cycles as compared to 1619 clock cycles for the RT-SCAN implementation, a reduction of more than 11 times.

A. Application to Industrial Design

RT-SCAN was applied to a large in-house RT-level design, `NECckt`, which has 7 functional units, 987 multiplexers, 110 registers, as shown in Table 2; the equivalent gate-level circuit has 840 FFs and 48,034 transistor pairs. Results of test generation, and application of RT-SCAN are shown in Table 3. Sequential test pattern generation using Hitec produced an unacceptably low test efficiency of 34%. While both full-scan and RT-SCAN could achieve 100% test efficiency, the RT-SCAN implementation reduced area overhead by 86% by exploiting the large number of register to register paths through multiplexors present in the circuit.

VI. Conclusions

This paper introduces a low overhead test methodology, *RT-SCAN*, applicable to RT Level designs. The methodology enables using combinational test patterns for testing the circuit, as done by traditional full-scan or parallel-scan schemes. However, by exploiting existing connectivity of registers through multiplexors and functional units, RT-SCAN reduces area overhead and test application times significantly compared to full-scan and parallel-scan schemes. Unlike most of the existing high-level test synthesis and test generation schemes which can be most effectively applied to data-flow/arithmetic intensive designs like DSPs and processor designs, the RT-SCAN test scheme can be applied to designs from any application domain, including control-flow intensive designs.

References

[1] E. Eichelberger and T. Williams. A Logic Design Structure for LSI Testability. In *14th Design Automation Conference, ACM/IEEE*, pages 462–468, June 1977.

[2] C. Lin, T-C. Lee, M. Marek-Sadowska, and K-C. Chen. Cost-Free Scan: A Low-Overhead Scan Path Design Methodology. In *Proceedings of ICCAD*, pages 528–533, November 1995.

[3] S.M. Reddy and R. Dandapani. Scan Design Using Standard Flip-Flops. *IEEE Design and Test*, pages 52 – 54, Feb 1987.

[4] B. Vinnakota and N. K. Jha. Synthesis of Sequential Circuits for Parallel Scan. In *Proc. of the European Conference on Design Automation*, pages 366–370, March 1992.

[5] S. Narayanan and M.A. Breuer. Reconfigurable Scan Chains: A Novel Approach To Reduce Test Application Time. In *Proc. of ICCAD*, pages 710 – 715, 1993.

[6] K. Wagner and S. Dey. High Level Synthesis for Testability: A Survey and Perspective. In *Proc. Design Automation Conference*, pages 131–136, June 1996.

		Test Hardware	Test Area Overhead (transistor pairs)	# of Faults	FC	TE	TAT
GCD	ORIG	-	-	2247	81.0%	83.0%	776
	FSCAN	52 SFF + 105 I + 3 P	565	2213	98.0%	100%	18008
	RTSCAN	4 M + 5 G + 15 I + 1 P	26	2386	98.0%	100%	1619
Barcode	ORIG	-	-	1459	62.0%	72.0%	566
	FSCAN	47 SFF + 95 I + 3 P	510	1421	90.2%	100%	5903
	RTSCAN	23 M + 8 G + 39 I + 1 P	108	1526	91.7%	100%	751
Change	ORIG	-	-	1677	72.0%	78.0%	349
	FSCAN	31 SFF + 63 I + 3 P	334	1669	90.2%	100%	4607
	RTSCAN	7 M + 10 G + 28 I + 1 P	49	1751	94.8%	99.9%	831
x25	ORIG	-	-	1677	43.0%	46.0%	70
	FSCAN	60 SFF + 121 I + 3 P	653	1659	98.0%	100%	7746
	RTSCAN	28 M + 8 G + 43 I + 1 P	128	1874	98.4%	100%	977
NECckt	ORIG	-	-	41156	32.0%	34.0%	2132
	FSCAN	840 SFF + 1681 I + 3 P	9072	41234	98.7%	100%	$2 * 10^6$
	RTSCAN	289 M + 65 G + 420 I + 1 P	1286	42108	98.9%	100%	$1 * 10^5$

Table 3: Test generation results and overheads for different test strategies.(SFF = Scan FF,I = Interconnect,P = Pin,M=Mux,G = Gate)

[7] V. Chickermane, J. Lee, and J.H. Patel. Addressing Design for Testability at the Architectural Level. *IEEE Transactions on Computer-Aided Design*, 13(7):920–934, July 1994.

[8] J. Steensma, F. Catthoor, and H. De Man. Partial Scan at the Register-Transfer Level. In *Proceedings of the International Test Conference*, pages 488 – 497, October 1991.

[9] C.-H. Chen, T. Karnik, and D.G.Saab. Structural and Behavioral Synthesis for Testability Techniques. *IEEE Transactions on Computer-Aided Design*, 13(6):777–785, June 1994.

[10] T. Thomas, P. Vishakantaiah, and J.A. Abraham. Impact of Behavioral Modifications For Testability. In *Proceedings of the 12th IEEE VLSI Test Symposium*, pages 427–432, April 1994.

[11] T. C. Lee, N. K. Jha, and W. H. Wolf. Behavioral Synthesis of Highly Testable Data Paths under Non-Scan and Partial Scan Environments. In *Proc. Design Automation Conf.*, pages 292–297, 1993.

[12] M. Potkonjak, S. Dey, and R. Roy. Behavioral Synthesis of Area-Efficient Testable Designs Using Interaction Between Hardware Sharing and Partial Scan. *IEEE Transactions on Computer-Aided Design*, 14(9):1141–1154, September 1995.

[13] S. S. K. Chiu and C. Papachristou. A Built-In Self-Testing Approach For Minimizing Hardware Overhead. In *Proceedings of the International Conference on Computer Design*, 1991.

[14] L. Avra. Allocation and Assignment in High-Level Synthesis for Self-Testable Data Paths. In *Proceedings of the International Test Conference*, 1991.

[15] I.G. Harris and A. Orailoğlu. Microarchitectural Synthesis of VLSI Designs with High Test Concurrency. In *Proc. Design Automation Conference*, pages 206–211, June 1994.

[16] S. Bhattacharya and S. Dey. H-SCAN: A High Level Alternative to Full-Scan Testing With Reduced Area and Test Application Overheads . In *VLSI Test Symposium*, April 1996.

[17] R. B. Norwood and E. J. McCluskey. Orthogonal Scan: Low Overhead Scan for Data Paths. In *International Test Conference*, pages 659–668, October 1996.

[18] M.S. Abadir and M.A. Breuer. A Knowledge Based System for Designing Testable VLSI Chips. *IEEE Design & Test of Computers*, 2(4):56–68, August 1985.

[19] S. Bhattacharya, S. Dey, and B. Sengupta. An RTL Methodology to Enable Low Overhead Combinational Testing . Technical Report 96-C016, C&C Research Labs, NEC USA, April 1996.

[20] Benchmarks available in the HLSW92 directory via anonymous ftp from ftp.cbl.ncsu.edu. High-Level Synthesis Workshop, 1992.

[21] T. M. Niermann and J. H. Patel. HITEC: A Test Generation Package for Sequential Circuits. In *Proc. EDAC*, pages 214–218, 1991.