Exploiting Symbolic Techniques for Partial Scan Flip Flop Selection

F. Corno, P. Prinetto, M. Sonza Reorda, M. Violante

Politecnico di Torino
Dipartimento di Automatica e Informatica
Torino, Italy

Abstract

Partial Scan techniques have been widely accepted as an effective solution to improve sequential ATPG performance while keeping acceptable area and performance overheads. Several techniques for flip-flop selection based on structural analysis have been presented in the literature. In this paper, we first propose a new testability measure based on the analysis of the circuit State Transition Graph through symbolic techniques. We then describe a scan flip flop selection algorithm exploiting this measure. We resort to the identification of several circuit macros to address large sequential circuits. When compared to other techniques, our approach shows good results, especially when it is used to optimize a set of flip-flops previously selected by means of structural analysis.

1. Introduction

The computational effort required to compute a set of test vectors for highly sequential large circuits is often prohibitive. Several Design for Testability (DfT) techniques have been developed to ease the test generation process [ABFr90]. A common approach consists in transforming the circuit by resorting to full scan. However, area and performance overhead required by full scan is often unacceptable; designers must balance the testability versus area and performance trade-off by transforming a subset of the circuit flip-flops, only, accounting to the partial scan technique. The problem is thus to select the minimum number of flip-flops to scan giving the maximum testability enhancement. Several algorithms have been proposed.

In [ChPa90] the problem is formulated and solved as an optimization one using three testability criteria. The first criterion deals both with the minimization of the number of cycles in the S-graph of the circuit and with the reduction of the circuit sequential depth. The second is based on the SCOAP controllability/observability measures, while the third deals with the reduction of the test sequence length. Results show a significant improvement in fault coverage when the first or the second testability measure are used.

In [HBFu96] Fucks et al. suggest to transform the flip-flops that are either difficult or impossible to set to ease the state justification operation. The methodology suggests to add a primary output for each flip-flop, then to perform a test generation process. Untestable faults on the new POs identify unsettable flip-flops, while aborted faults on the new POs point out those flip-flops difficult to set. Results show that the identification of states difficult to reach can effectively drive the selection of flip-flops for scan insertion.

In [BoFu96] the goal is to identify which transitions can be added to the State Transition Graph (STG) of the circuit, in order to traverse more easily the states of the machine. The methodology requires to run the ATPG and to observe which faults cannot be detected due to the inability of either propagating a fault effect to a primary output or justifying the desired state. A transition (called pseudo transition) from a reachable state to the nearest desired reachable states is then added to the STG. The flip-flops to scan are thus the ones needed to obtain the desired pseudo transitions. Results show that these techniques allow a good improvement both in fault coverage and in test generation time.

Another approach that relies on information gathered during the test generation process is presented in [XVFP96] and [XiPa96]. A testability measure based on the density of encoding is used to select an initial solution, and a variety of techniques is then used to select an optimal set of flip-flops to scan. The rationale behind these papers is to selectively break the cycles in the S-graph using circuit state information. Experimental results show the feasibility of the approach and that significant reductions in the number of flip-flops to scan can be achieved.

For a more comprehensive survey on partial scan techniques, problems, and solutions, the reader may refer to a special issue of the JETTA journal [Jett95].

In this paper we present a new testability measure and a new algorithm to solve the problem of flip-flop selection for partial scan. The main contribution of this work consists in a new approach to flip-flop selection

that does not rely on information gathered during the ATPG process. We analyze how a scanned flip-flop affects the circuit behavior by working on the State Transition Graph of the circuit (STG) represented by means of BDDs [Brya92]. The testability measure is based on the notion of *state distribution* of a STG. Several techniques to complete an approximate STG traversal are presented to deal with large sequential circuits. By resorting to our technique, we expect to select a set of flip-flops for scan insertion able to reduce the time spent during the test generation and to increase the fault coverage. Experimental results show the effectiveness of our approach and the ability to obtain good results both in terms of fault coverage and ATPG time.

We present our new testability measure in section 2 and the flip-flop selection algorithm in section 3. Section 4 describes how our approach can be extended to deal with large sequential circuits, while section 5 reports some experimental results to demonstrate the feasibility of our approach. Finally, we draw some conclusions in section 6.

2. STG Testability Measure

Great efforts are spent by test generation programs to traverse the State Transition Graph of highly sequential circuits. For instance, in HITEC [NiPa91] state justification is a very time consuming operation: a significant amount of time is used by the ATPG algorithm while trying to identify an input sequence able to transfer the circuit to a desired state from which a propagation path exists. If the desired state is an invalid one a significant amount of time is wasted to detect such a condition. Furthermore, in deeply sequential circuits, the amount of time required to justify a state can exceed the available time limit. In such a situation the ATPG produces a relatively high number of aborted faults.

The justification process can be viewed as a search on the circuit STG to find a path from an initial state, either the unknown or the reset one, to the desired state. Through the use of scan, we control the value loaded in a flip-flop, so we are able to force a state transition by applying an input vector. Therefore, by scanning a flip-flop, we add some new transitions to the circuit STG and we obtain an Extended State Transition Graph (E-STG) which, due to the increased connectivity, is easier to traverse and has a smaller sequential depth.

Working on the E-STG, the justification algorithm can choose among more paths to drive the circuit from the initial state to the desired state. As a result, the ATPG process is simplified. Furthermore, a circuit having a small sequential depth requires the ATPG to spend less time to justify any valid state, and the number of aborted faults is reduced. It is also important to consider that a reduced sequential depth allows the ATPG to spend less time in trying to justify an invalid state. The best flip-flops to scan are thus the ones that allow the APTG to easily reach all the desired states and that reduce the circuit sequential depth. Two results are expected: the reduction of the CPU time to perform the test generation, and an increased fault coverage. Other papers use similar considerations to propose alternative DfT strategies [HsPa95].

In order to select the flip-flops to scan using these concepts, a new testability measure able to address both the number of reachable states and the circuit sequential depth is defined. We first give the definition of *state distribution* of a State Transition Graph. Intuitively, it measures the average number of STG levels that must be explored to reach all the reachable states. It is a real number defined as:

$$SD = \frac{\sum_l l \cdot RS_l}{\sum_l RS_l} \qquad (1)$$

where RS_l is the number of states having minimum distance l from the initial state. In (1) we start computing the reachable states from the reset state. The literal l measures the STG level currently explored, the distance being measured in clock cycles. The state distribution SD_g for the unmodified circuit is first computed, where no transformation is performed over its memory elements during the computation.

In our testability analysis, we assume that a scanned flip-flop is fully controllable from a PI, and fully observable at a PO. We define SD_i as the state distribution of a circuit in which the flip-flop i is scanned. It is computed on the corresponding E-STG. Therefore, by comparing SD_g and SD_i we are able to evaluate the effect of a scan flip-flop insertion over the circuit behavior.

The weight of the flip-flop i is thus defined as:

$$W_i = \frac{SD_g - SD_i}{SD_g} \qquad (2)$$

This measure allows to relate the behavior of a circuit where a single flip-flop is scanned with the behavior of the unmodified circuit. A flip-flop having a high value for W_i allows to reduce the circuit sequential depth and to increase the probability for the ATPG to reach all the desired states. Conversely, a flip-flop having a low value for W_i is not well suited for scan insertion.

It is important to note that, unlike the majority of the other approaches ([XiPa96], [XVFP96] and [ChPa90]), we work on the circuit behavior represented by the STG, only.

As an example, Figure 1 represents the State Transition Graph of a circuit having two flip-flops. The STG (a) in Figure 1 has four reachable states and three levels; by applying (1) we obtain that SD_g = 2. Suppose that modifying the circuit in order to scan the flip-flip A gives the STG (b): the value of SD_A is 1.75; on the other side, scanning the flip-flop B the STG (c) is obtained, having SD_B equal to 2. The weight of the flip-flop A is W_A=0.125 while the B one is W_B=0, as stated by (2). This result means that by scanning the flip-flop A we obtain a global improvement of the circuit testability while by scanning the flip-flop B no benefits are expected.

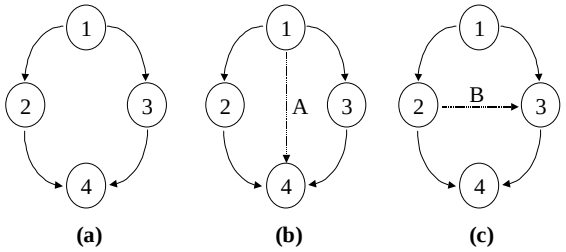

Figure 1: *An example of State Transition Graph*

3. Selection Algorithm

The testability measure defined so far assumes the availability of the circuit State Transition Graph. Since we are working on a gate-level description, we have to build the STG starting from the netlist. A first feasible approach is to obtain the valid states of the machine by means of logic simulation, as done in [XiPa96]. This approach is not exact because it depends both on the time spent for the simulation and on the input pattern distribution. As a result, a significant amount of simulation efforts can be required to traverse all the valid states of a large sequential circuit.

To apply our methodology we should also be able to evaluate equation (1) several times: once for the unmodified circuit and once for every E-STG that can be obtained from the original State Transition Graph by transforming each flip-flop. This requires to modify the circuit, to rebuild the STG and to compute the required value.

To efficiently perform such operations we resort to BDDs [Brya92] and symbolic traversal techniques. The circuit is modeled as a Finite State Machine, and is represented by the Boolean functions δ and λ. Function δ computes the next state y from the current state s and the current input x: $y = \delta(s, x)$; function λ computes the output z starting from the same information: $z = \lambda(s, x)$. The techniques resort to the adoption of *characteristic functions* [Brya92] to represent sets of inputs $\chi_X(x)$, set of states $\chi_S(s)$, the state transition relation $TR(s, y)$, and the output function $\chi_\lambda(z, x, y)$.

The transition relation is defined as follows:

$$TR(s,y) = \exists x \left(\prod_j \delta_j(s,x) \overline{\oplus} y_j \right) \qquad (3)$$

It is true for every couple (s,y) for which an input x exists that satisfies $y=\delta(s, x)$, i.e., whenever y is a valid successor to s under some input value x. The algorithm has to compute the set of reachable states, thus it works on the TR function, only. We save memory space by not representing the λ function.

```
// onset(c): size of the on-set
// of characteristic function

compute_SD(i)
{
  TR = build_TR();    // eq.(3)
  if(i != NULL)
    TR = ∃s_i∃y_i(TR);  //scan FF
  l = 0;          // seq. depth
  Ni = 0;         // sum in eq.(1)
  end = FALSE;
  current = ResetState;
  reached = ResetState;
  do {
    new = ∃s(current · TR(s,y));
    current = new-reached;
    if(current == 0)
      end = TRUE;
    else
    {
      l++;
      Ni += l*onset(current);
      reached = reached+current;
    }
  } while(end != TRUE);
  return(Ni/onset(reached));
}
```

Figure 2: *Algorithm to compute the reachable states*

Figure 2 reports the algorithm used to compute the reachable states of the circuit; two aspects deserve greater detail: how the required circuit transformations are performed and how the transition relation is computed.

The circuit transformations can be easily performed by means of BDD operators. A scanned flip-flop i is functionally equivalent to a pair composed of a PI and a

PO. In terms of BDDs this can be computed by applying the existential quantification operator to the transition relation $TR(s,y)$ with respect to the variables s_i and y_i:

$$TR'(s,y) = \exists s_i \exists y_i \left(TR(s,y) \right) \quad (4)$$

To describe how the transition relation is computed, let us consider the following expression:

$$\chi_\delta(s,x,y) = \prod_j \delta_j(s,x) \oplus y_j \quad (5)$$

The number of BDD variables required to compute (5) is equal to twice the number of flip-flops plus the number of inputs. Whenever this value exceeds a threshold related to the available memory, the resulting BDD cannot be held in memory; as a result, equation (5) cannot be computed. Two solutions can be devised: we can reduce the number of state variables, s and y, or we can reduce the number of input variables, x. The former solution resorts to the decomposition of the circuit in macros and is described in more detail in section 4. The latter solution consists in ignoring the contribution of some input variables when computing equation (5). This can be done by defining an approximated next state function δ^* that takes into account the first L input variables, only:

$$\delta_j^*(s,x) = \exists_{h \geq L} x_h \left(\delta_j(s,x) \right) \quad (6)$$

The number of variables required to represent equation (6) is thus equal to the number of flip-flops plus L, i.e., the maximum number of inputs taken into account. The value of L is empirically chosen, according to the available memory. The new state transition relation, $TR^*(s,y)$, is defined as follows:

$$TR^*(s,y) = \exists x \prod_j \left(\delta_j^*(s,x) \oplus y_j \right) \quad (7)$$

Such an approach allows us to keep reasonably low the BDD size but introduces some approximations: by removing an input variable we may add transitions in the STG. As a result, the number of reachable states computed by our algorithm is larger than or equal to the one computed over the unmodified STG. This approximation has minor effects over the flip-flop selection process because W_i is computed in (2) as a relative number over SD_g and SD_i, both affected by the same over-estimation of the reachable states.

The algorithm used to identify the best flip-flops to scan, described in Figure 3, performs its task in three steps: it computes the state distribution for the unmodified circuit, then computes the state distributions of the STGs obtained by individually transforming each flip-flop in a PI; finally, it computes the flip-flop weights using (2). The last operation performed is the sorting of the weight array in decreasing weight order. The first flip-flops in such a ranking, having the higher weights, are the ones best suited for scan.

```
weight_FFs_to_scan()
{
   SDg = compute_SD(NULL);
   for(every FF i)
   {
      SDi = compute_SD(i);
      W[i] = (SDg-SDi)/SDg;
   }
   sort(W);
}
```

Figure 3: *Algorithm to compute the FF weights*

4. Addressing larger circuits

In this section an approach to address large sequential circuits is described. To apply symbolic functions to large sequential circuits in section 4.1 we propose an approach based on the identification of some macros inside the circuit: several macros are used to represent the connected components of a circuit, then the algorithm in Figure 3 is applied to each macro. The adoption of several macros for enabling BDDs to be used on large circuit is also found in [CGPS95], where this technique was used to enhance the performance of topological ATPG. In section 4.2, we describe how to use structural information to identify which circuit flip-flops should never be selected for scan insertion, then we show how to apply our symbolic approach to identify which of the remaining flip-flops are the ones best suited for scan.

4.1. Circuit Partitioning

Symbolic techniques are very efficient, but their applicability is limited to circuits with a few tens of flip-flops. In order to address larger circuits, we split the circuit in several macros and we apply the BDD computation to each macro separately. Every macro is a *connected component*, i.e., a portion of the circuit where flip-flops are connected to each other through combinational logic, and is small enough to be handled by symbolic techniques.

Ideally, macros should be selected according to some high-level information, where the registers with the higher sequential depth are easily identified. However, when working on gate-level benchmarks, one has to resort to heuristic analysis of the netlist.

We build a macro in three steps: we first select a set of flip-flops (they will become the memory elements of

the macro), then we chose the combinational logic to insert in the macro, and finally we identify the macro inputs and outputs. Figure 4 sketches a simple macro, which is identified by a single flip-flop. The combinational logic is computed by exploring the input and output cones of every memory element of the macro. All the combinational gates that both influence the macro flip-flops and are influenced by them are selected. In Figure 4 the imaginary macro boundary, depicted as a dashed line, crosses several nets. A net going from a gate outside such a boundary to a gate inside the macro is a macro input. Conversely, a net coming from a macro gate to a gate outside the macro boundary is a macro output.

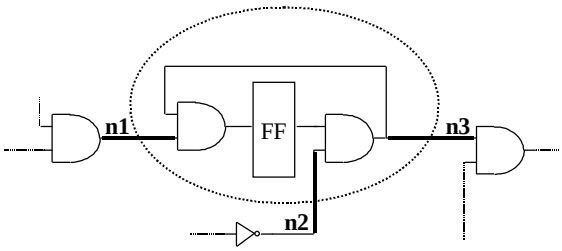

Figure 4: *A simple macro*

The idea behind macros is to project a State Transition Graph too large to fit in memory into several smaller STGs. Several sets of reachable states are thus identified and elaborated by applying our algorithm to every macro.

The memory elements are selected in such a way that all the flip-flops in a macro are closely related, i.e., the S-graph representing them must have at least one cycle. If this requirement is not satisfied the output of the symbolic calculations is useless: in fact, for most cases in a set containing n unrelated flip-flops all the 2^n states configurations are valid, and all the flip-flops would thus have the similar weights.

The task of selecting the flip-flops to insert in a macro is carried out using a small amount of resources, by performing a Depth First Search (DFS) of the flip-flops, starting from the POs. The list of flip-flops is then ordered. The i-th flip-flop is usually related to the $(i+1)$-th one in such a list because adjacent flip-flops are in the input cone of the same PO. Therefore, several sets of flip-flops can be extracted by splitting the list in sets of adjacent elements. Every set represents the memory elements of a macro. The size of these sets is selected in such a way that the number of gates in the corresponding macro can be managed by means of symbolic procedures either in an exact or approximate way.

When all the STG partitions are not overlapping, i.e., each flip-flop appears at most in a macro, it is possible to have some invalid states included in our computations since interactions between macros are not properly taken into account. To improve the results of this approach we force the macro flip-flop selection algorithm to insert every flip-flop in at least two macros. Therefore, the obtained subsets are partially overlapped. This ensures a higher probability to avoid computing invalid states.

The algorithm of Figure 3 is used to compute the weight for each flip-flop in every macro, and then the results coming from the different macros are gathered to fill a final ranking. We assign to each flip-flop the highest weight obtained across the macros in which it is included. At the end of this process we still have a sorted array holding a unique weight for each flip-flop of the circuit.

4.2. Flip Flop Preselection by Structural Analysis

The macro based approach allows us to increase the affordable size, but circuits exist that cannot be managed using such techniques. The macros obtained by the DFS exploration have too many gates to be handled by symbolic procedures. Trying to split the macros in smaller ones produces bad results, because too few flip-flops are held in the new macros. We have a high probability to obtain a State Transition Graph with few cycles, or none at all, for a macro holding four or less flip-flops, therefore our symbolic manipulations produce meaningless results when using such macros.

The idea behind the approach described in this section is to use a structural analysis to identify which flip-flops are not well suited for scan insertion, to allow the symbolic procedure to work over a reduced set of flip-flops.

We proceed as follows:

1. identify a set S of flip-flops having a known high impact on testability
2. apply a DFS ordering to S to extract the structural adjacency between flip-flops
3. split S in several macros using the approach described in section 4
4. apply the algorithm of Figure 3 to the macros.

The rationale behind this procedure is to use our selection algorithm as an optimization tool over a subset of flip-flops that are known to heavily influence the circuit testability. We first roughly identify which flip-flops are nearly useless for scan insertion by performing a testability analysis, for example by computing the SCOAP measurements [Gold79] or, as in our experiments, by using OPUS [ChPa90]. We then analyze the remaining flip-flops to identify which are the ones best suited for scan insertion. By reducing the number of

			SDSCAN			OPUS	
Circ.	FF	FFS	FC [%]	CPU [s]	M	FC [%]	CPU [s]
s298	14	1	98.1	9.17	1	98.1	4.7
s344	15	5	99.7	0.90	1	99.7	0.33
s349	15	5	99.1	0.87	1	99.1	0.37
s382	21	9	97.5	3.77	1	97.5	3.53
s386	6	5	100	0.72	1	100	0.77
s400	21	9	96.2	3.68	1	96.2	3.73
s444	21	9	94.9	4.13	1	94.9	4.07
s499	22	9	88.2	37.38	1	88.2	38.43
s510	6	5	100	1.18	1	100	0.82
s526	21	3	84.9	144.42	1	89.6	120.17
s641	19	7	99.4	9.05	1	99.4	4.90
s713	19	7	93.1	8.12	2	92.9	7.52
s820	5	2	100	9.12	1	100	9.27
s832	5	2	98.4	9.82	1	98.4	12.63
s953	29	3	99.8	4.40	2	99.9	3.23
s967	29	5	100	2.55	2	100	2.47
s1196	18	3	100	5.38	1	99.8	6.08
s1269	37	6	99.6	19.10	1	99.4	32.97
s1423	74	30	87.0	429.62	18	78.6	755.88
s1488	6	2	99.4	54.57	1	100.0	38.60
s1494	6	3	99.2	25.13	1	99.2	28.55

Table 1: *Results of SDSCAN vs OPUS*

flip-flops to analyze, we have obtained a problem solvable by means of symbolic calculations proposed in the previous sections.

It is important to emphasize that the described structural analysis for macro selection could be successfully avoided if higher-level information about the circuit behavior were available. By incidence, preliminary experimental results show that for circuits composed of Control Units and Data Paths, selecting the former ones as macros is a very effective choice.

5. Experimental Results

We have developed a prototype tool, called SDSCAN, to test our algorithm. To present data on well-known benchmarks, we adopted the ISCAS'89 circuits [BBKo89], including the Addendum ones [Adde93].

In order to compare our results with the ones provided by OPUS [ChPa90], we adopted the HITEC [NiPa91] test generation tool, and took as evaluation parameters the required CPU time and the attained Fault Coverage on the circuits modified according to SDSCAN and OPUS, respectively. For the sake of comparison, the number of flip-flops being scanned, FFS in the tables, has been selected according to [XiPa96].

Section 5.1 reports the results obtained by SDSCAN when the methodology described in section 4.1 is used, while section 5.2 reports the results obtained when structural and behavioral information are combined to obtain an optimal set of scan flip-flops, as described in section 4.2.

5.1. Selecting the Scan Flip Flop Subset

Table 1 shows the results obtained on a subset of ISCAS'89 circuits when the set of scan flip-flops is selected by reasoning over the whole circuit, as described in section 4.1. We have computed the weight for all the circuit flip-flops according to equation (2); then, the best flip-flops have been selected as described in section 3. In Table 1, FC is the fault coverage computed by HITEC when the circuit under test has FFS scanned flip-flops (taken from [XiPa96]); CPU is the time spent by HITEC to compute the test patterns, while FF is the number of flip-flops in the circuit. M is the number of macros used by SDSCAN. The time to compute the set of scan flip-flops for both OPUS and SDSCAN is negligible with respect to the one required by HITEC during the test generation, thus it has not been reported.

					SDSCAN			OPUS	
Circ.	FF	FFS	ΔFC [%]	ΔT [%]	FC [%]	CPU [s]	M	FC [%]	CPU [s]
s3271	116	20	2.4	-53.6	99.0	135	11	96.6	291
s5378	179	30	2.7	-19.5	92.7	832	9	90.2	1,034
s13207	669	80	6.8	-57.4	33.7	10,883	23	31.4	25,569
s35932	1728	150	0.1	-22.9	89.9	15,677	59	89.8	20,343

Table 2: *Results of SDSCAN on some of the larger ISCAS*

Table 1 shows that SDSCAN produces results comparable with OPUS when dealing with small and medium size sequential circuits. For most of the circuits taken into account, SDSCAN produces the same results of OPUS. By using SDSCAN, we have obtained a speed-up of 42% both for s1269 and s1423, while a fault coverage improvement of 10% is measured for s1423. Conversely, OPUS outperforms SDSCAN on s526 due to a lower number of redundant faults. Our results are also comparable with the ones obtained by OPSCAN [XiPa96].

5.2. Optimizing the Scan Flip Flop Subset

Table 2 reports the results obtained by applying the approach described in section 4.2 to some of the larger benchmarks. The idea is to identify a set of flip-flops which are good candidates for scan insertion, and then to optimize it by means of the symbolic techniques previously described. The initial set of flip-flops is computed using OPUS: by means of its testability analysis we identify *2N* flip-flops. We then perform the optimization step by computing our testability measure over the initial solution and by selecting the *N* flip-flops which have the best weight, according to equation (2).

In Table 2, the column ΔT measures the percent difference between the two CPU times, while the column ΔFC measures the percent difference between the two fault coverage values. All the experiments in this section have been performed assuming that the circuits have a synchronous reset signal. The average fault coverage improvement is in favor of SDSCAN by a 3% factor, while the average time to perform the test generation is reduced by 38% when the flip-flops for scan insertion are selected by means of SDSCAN.

6. Conclusion

A new approach to solve the problem of flip-flop selection for partial scan is presented in this paper, based on a new testability measure named *state distribution*. By exploring the State Transition Graph of the circuit under test, and combining topological and symbolic techniques, the proposed algorithm is able to produce results comparable with a valid tool such as OPUS [ChPa90].

To deal with large sequential circuits we devised two approximations: we limit to *L* the number of input variables taken into account during the symbolic analysis of the circuit behavior and, whenever required, we resort to the selection of several *macros* to represent the circuit STG. To address circuits for which these approximations are not effective in reducing the BDDs size, we propose an approach that relies both on structural information and symbolic analysis.

Experimental results show that our tool can be effectively used to select flip-flops for scan insertion and compare well with other methods previously proposed. In particular, we have obtained good improvements in terms of CPU time for test generation and attained Fault Coverage when applying our technique to the largest benchmarks circuits.

We believe that the best macro selection can be performed by exploiting functional information coming from the availability of an RT-level description of the circuit: our technique could thus benefit from the adoption of a top-down design approach and would fit well in a design flow including automatic synthesis tools.

7. References

[ABFr90] M. Abramovici, M. A. Breuer, A. D. Friedman: *Digital system testing and testable design*, Computer Science Press, New York, NY (USA), 1990.

[Adde93] These benchmark circuits are downloadable from the address **`http://www.cbl.ncsu.edu/www/CBL_Docs/Bench.html`**.

[BBKo89] F. Brglex, D. Bryant, K. Kozmiski, "Combinational profiles of sequential benchmarks cricuits", *Proc. Int. Symp. on Circuits and System*, 1989, pp. 1929-1934.

[BoFu96] V. Boppana, W. K. Fuchs, "Partial Scan Design Based on State Transition Modelling", *Proc. of the Int. Test Conf.*, pp. 538-547, 1996.

[Brya92] R. E. Bryant: *Symbolic Boolean Manipulation with Ordered Binary Decision Diagram*, ACM

Computing Surveys, Vol. 24, Nr. 3, 1992, pp. 293-318.

[CGPS95] F. Corno, U. Gläser, P. Prinetto, M. Sonza Reorda, H.-T. Vierhaus: *Improving Topological ATPG with Symbolic Techniques*, VTS'95: IEEE VLSI Test Symposium, Princeton NJ, pp. 338-343

[ChPa90] V. Chickermane, J. H. Patel, "An Optimization Based Approach to the Partial Scan Design Problem", *Proc. Int. Test Conf.*, pp. 377-386, 1990.

[Gold79] L. H. Goldstein, "Controllability/Observability Analysis of Digital Circuits", *IEEE Trans. Circuits and Systems*, Vol. 26, pp. 685-693, 1979.

[HBFu96] I. Hartanto, V. Boppana, W. K. Fuchs, "Identification of Unsettable Flip-Flops for Partial Scan and Faster ATPG", *Proc. of the ICCAD*, 1996.

[HsPa95] F. F. Hsu, J. H. Patel, "A Distant Reduction Approach to Design for Testability", *Proc. of the IEEE VLSI Test Symposyum*, pp. 158-163, 1995.

[Jett95] JETTA: Journal of Electronic Testing: Theory and Applications: special issue on partial scan methods, Vol. 7, n. 1/2, August/October 1995

[NiPa91] T. M. Niermann, J. H. Patel, "HITEC: A Test Generation Package for Sequential Circuits", *Proc. of the European Conference on Design Automation*, pp. 214-218, 1991.

[XiPa96] D. Xiang, J. H. Patel, "A Global Algorithm for the Partial Scan Design Problem Using Circuit State Information", *Proc. of the Int. Test Conf.*, pp. 548-557, 1996.

[XVFP96] D. Xiang, S**.** Venkataraman, W. K. Fuchs, J. H. Patel, "Partial Scan Design Based on Circuit State Information", *Proc. of the ACM/IEEE DAC-33*, pp. 807-812, 1996.