Testability Analysis of Circuits using Data-Dependent Power Management

José C. Monteiro, João P. Marques Silva
Dept. of Electrical and Computer Engineering, IST/INESC
Rua Alves Redol, 9, 134, 1000 Lisboa, Portugal.
jcm@inesc.pt, jpms@inesc.pt

Abstract

Power dissipation has recently emerged as one the most critical design constraints. *Data-dependent* power management techniques are among the most effective for power reduction. Depending on some input conditions, the clock driving some of the registers in the circuit is inhibited, thus reducing the switching activity in the fanout of those registers.

The use of data-dependent power management techniques creates some interesting testability problems. The signals used to inhibit the clock can dramatically reduce the observability of the nodes in the circuit. In this paper, we first describe an approach for the complete testing of power-optimized circuits using these techniques. We then present results that show that using state-of-the-art techniques, ATPG for the power managed circuit is not significantly more difficult than for the original circuit.

Keywords

Low power, power management, test pattern generation, propositional satisfiability

1 INTRODUCTION

Two aspects have combined to make power consumption one of the most critical design parameter. On one hand, the rapid increase in clock frequencies, chip complexity and scale of integration is creating significant heat dissipation problems. High operating temperatures can affect the circuit's reliability and reduce the lifetime of the system. In order to dissipate the heat that is generated, special packaging and cooling systems may be required, leading to higher costs.

On the other hand, we have the proliferation of portable devices. For personal communication applications like hand-held mobile telephones, battery lifetime may be the decisive factor in the success of the product.

Power reduction techniques have been proposed at all design levels, from system to device. It has been demonstrated at the gate and system levels that large power savings are possible merely by cutting down on wasted power, commonly referred

to as power management. At the system level, this involves shutting down blocks of hardware during a period of time for which they are not being used.

Several methods have been presented that perform the shutdown of a section of the circuit on a clock-cycle base. Depending on some input conditions, the clock driving some of the registers in the circuit is inhibited, therefore reducing the switching activity in the fanout of those registers. These techniques are referred to as *data-dependent* power management techniques.

The use of data-dependent power management techniques creates some interesting testability problems. In order to detect the input conditions that allow for the disabling of the clock signal, some extra circuitry has to be added to the original circuit. Since the functionality of the circuit is not being altered, this logic is redundant, thus dramatically reducing the observability of some of the nodes in the circuit.

In this paper, we first describe an approach for the complete testing of power-optimized circuits using data-dependent power management. The approach we propose is based on a two input vector sequence. The first input vector guarantees that we can always set the output of the latches to the required value, even if the clock is disabled in the second input vector.

Using this testing strategy, we then present results that show that using state-of-the-art techniques, automatic test pattern generation for the power managed circuit is not significantly more difficult than for the original circuit.

2 POWER MANAGEMENT TECHNIQUES

In a synchronous digital system controlled by a global clock, a generally accepted model for the power dissipated by a gate is given by:

$$P_{avg} = 0.5 \times C_{load} \times (V_{dd}^2/T_{cyc}) \times E(transitions), \quad (1)$$

where P_{avg} denotes the average power, C_{load} is the load capacitance, V_{dd} is the supply voltage, T_{cyc} is the global clock period, and $E(transitions)$ is the expected number of gate output transitions per global clock cycle (Najm 1993).

It follows directly from Equation 1 that one way to reduce power consumption is to reduce the switching activity $E(transitions)$ in a circuit. So called power management techniques that shutdown hardware for periods of time in which it is not producing useful data are effective methods of reducing the power consumption of a circuit. Shutdown can be accomplished by either turning off the power supply or by disabling the clock signal.

System-level approaches identify idle periods for entire modules and turn off the clock for these modules for the duration of the idle periods (Chandrakasan, Sheng & Brodersen 1992, Chapter 10). Detection and shutdown of unused hardware is done automatically in current generations of Pentium and PowerPC processors. The trend is that future generation of processors will provide software controls for selective hardware shutdown, feature already available in the Fujitsu SPARClite processor.

Scheduling algorithms that maximize the shutdown period of execution units in a system have been presented (Monteiro, Devadas, Ashar & Mauskar 1996). Given throughput constraints and the execution units available, operations are scheduled such that those that generate controlling signals are computed first, thus indicating the flow of data through the circuit. Power is saved by only activating hardware modules involved in computing the final result, all other modules being shutdown.

At the logic level, some shutdown techniques, such as *precomputation* (Alidina, Monteiro, Devadas, Ghosh & Papaefthymiou 1994), *guarded evaluation* (Tiwari, Ashar & Malik 1995) and *gated-clock finite state machines* (Benini, Siegel & Micheli 1996), have been proposed recently and are among the most efficient power optimization techniques at this design level. These techniques are named *data-dependent* power management techniques as power management is achieved on a clock-cycle basis, function of the input conditions at the beginning of each clock cycle.

The *precomputation* method (Alidina et al. 1994) adds a simple combinational circuit (the precomputation logic) to the original circuit. Under certain input conditions, the precomputation logic disables the loading of all or a subset of the input registers. Under these input conditions, no power is dissipated in the portions of the original circuit with only disabled registers as inputs. We will analyse this technique in more detail in the next section.

Guarded evaluation (Tiwari et al. 1995) identifies cones internal to the circuit that can be shut down under certain input conditions. In the process, it creates new transition barriers (guards) in the form of additional latches or OR/AND gates. Instead of adding the precomputation logic to generate the clock disabling signal, guarded-evaluation uses signals already existing in the circuit.

The *gated-clock finite state machines (FSM)'s* approach (Benini et al. 1996) is based on identifying self-loops in a Moore FSM*. If the FSM enters a state with a self loop, the clock is turned off. In this situation, the inputs to the combinational logic block do not switch, and thus we have virtually zero power dissipation in that block. When the input values cause the FSM to make a state transition, the clock signal is again allowed to function normally. Techniques to transform locally a Mealy machine into a Moore machine are presented so that the opportunity for gating the clock is increased.

All these techniques achieve power reduction by stopping transitions at the inputs from propagating to combinational logic blocks. However, this has the undesirable consequence of making the testing of the circuit more difficult. Since for some input combinations the loading of the registers is disabled, it is not possible to have all combinations at the inputs of the combinational logic blocks.

We present a method for the automatic test pattern generation of data-dependent power managed circuits in Section 4. Although the problem is similar for the three techniques presented above, we will focus on the precomputation technique in the remainder of this paper. For this reason, in the next section we analyze this technique in more detail.

---

*In a Moore FSM, the outputs are completely defined by the present state, whereas in a Mealy FSM the outputs depend both on the present state and primary input lines.

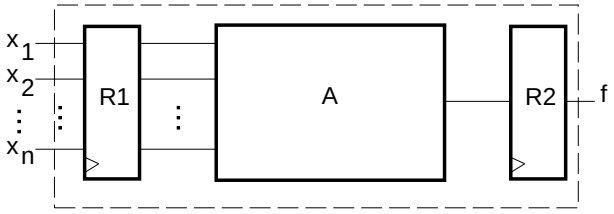

Figure 1 Circuit before precomputation.

3 PRECOMPUTATION FOR LOW POWER

Consider the circuit of Figure 1, where the combinational logic block A is bounded by registers R_1 and R_2. While R_1 and R_2 are shown as distinct registers in Figure 1 they could, in fact, be the same register (as in a FSM).

The precomputation architecture is shown in Figure 2. The inputs to the block A have been partitioned into two sets, corresponding to the registers R_1 and R_2. The output of A feeds the register R_3. The Boolean functions g_1 and g_2 are the *predictor* functions and have the same inputs as register R_1. g_1 and g_2 are defined so that:

$$g_1 = 1 \Rightarrow f = 1. \tag{2}$$

$$g_2 = 1 \Rightarrow f = 0. \tag{3}$$

Therefore, during clock cycle t if either g_1 or g_2 evaluates to a 1, the load enable signal of the register R_2 is set to be 0. This implies that the outputs of R_2 during clock cycle $t+1$ do not change. However, since the outputs of register R_1 are updated, the function f will evaluate to the correct logical value. A power reduction is achieved because only a subset of the inputs to block A change, implying reduced switching activity.

Note that g_1 and g_2 add to the delay of paths that originally ended at R_1 but now pass through g_1 or g_2 and the NOR gate before ending at the load enable signal of the register R_1. Therefore, caution should be used so that this transformation is applied on non-critical signals or logic blocks.

The choice of g_1 and g_2 is critical. On one hand, we should include as many input conditions as possible in g_1 and g_2, i.e., we want to maximize the probability of g_1 or g_2 evaluating to a 1. On the other hand, g_1 and g_2 correspond to extra logic that is added to the circuit, consequently increasing power consumption. To obtain reduction in power with marginal increases in circuit area and delay, g_1 and g_2 have to be significantly less complex than f. The precomputation architecture of Figure 2 ensures this by making g_1 and g_2 depend on significantly fewer inputs than f. Methods to automatically determine the precomputation logic of a circuit are described in (Alidina et al. 1994).

Using these same principles, different precomputation architectures have been proposed in (Alidina et al. 1994, Monteiro, Rinderknecht, Devadas & Ghosh 1995).

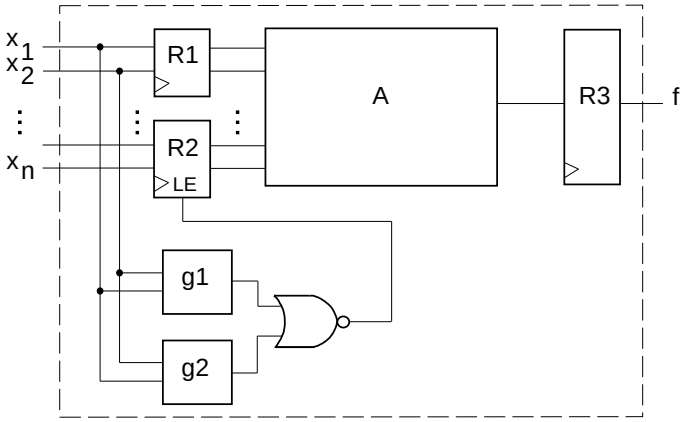

Figure 2 Precomputation architecture.

4 ATPG TECHNIQUES FOR POWER MANAGED CIRCUITS

We address the problem of generating test patterns to detect faults in circuits using data-dependent power management techniques. These techniques, overviewed in Section 2, all use mechanisms to prevent transitions in some logic signals from propagating to combinational logic circuits, thus reducing power consumption. As a consequence, some input combinations to the combinational logic block may no longer be possible, hence dramatically reducing the effectiveness of ATPG programs.

4.1 Definition of the Problem

We will assume that we have full controllability of the inputs and observability of the outputs of the original circuit, i.e., the circuit before the power management techniques have been applied. Our objective is to measure how the testability of the circuit before (Figure 1) and after (Figure 2) power management compare.

Suppose the combinational logic block A in Figure 1 can have a total of m stuck-at faults. Since we are assuming full controllability and observability of the inputs and outputs respectively, standard ATPG techniques can be used for test generation.

After we apply power management, extra logic has been added to the circuit. In Figure 2, this corresponds to functions g_1 and g_2 and to the NOR gate. If the possible number of faults in this extra logic is k, the total number of faults in the precomputed circuit will be $m+k$. However, to be effective, g_1 and g_2 are necessarily much simpler than A, therefore $m + k$ is not much larger than m. Thus, it is not the extra number of faults that makes test generation significantly more difficult.

The major test generation problems arise from the fact that the precomputation logic will prevent some input combinations to the logic block from happening. For this reason, in order to fully test the power managed circuit, we may need two input vectors. We describe this approach in Section 4.3.

Still, even using two input vectors for testing, the precomputation logic introduces

many redundant faults. Redundant faults can create difficulties for most ATPG programs as they try to prove that the fault is indeed redundant. In Section 5, we describe an ATPG algorithm that can efficiently handle all the redundant faults introduced by the precomputation logic.

4.2 Testing using Scan Techniques

Before we describe the our testing approach based on a two input vector sequence, it is worth mentioning the case for circuits using scan test techniques. With scan techniques, the registers in the circuit are connected in series. During the testing process, these registers can be directly loaded with the desired values and their contents can also be read. We therefore have full observability and controllability of the inputs and outputs of the combinational blocks in the circuit.

For circuits with scan techniques, precomputation does not create any significant additional testing problem as all registers can be directly loaded, thus circumventing the precomputation logic. The only extra concern is the test of the precomputation logic, which as previously stated should be a very small fraction of the total logic.

However, there is some overhead associated with scan techniques. This overhead is generally too expensive for all registers in the circuit to be included in the scan chain. In practice, partial scan is used, where only a fraction of the register are in the scan chain. Under partial scan, a sequence of input vectors may have to be generated in order to set the output of registers not in the scan chain to some desired value. Hence, the motivation for our two input vector based testing approach that we present next.

4.3 Testing using a Two Input Vector Sequence

As described before, whenever power management is asserted, some input transitions are prevented from reaching a portion of some combinational logic block. This may impede certain input combinations at this logic block from happening. We propose to solve this controllability problem by using a two input vector sequence. As observability of the outputs is assumed, the result of the test can be verified after the second input vector.

There are two situations to consider. First, consider a fault that can be detected using some input combination that disables the precomputation logic. Then, the first input vector is of no relevance since the second input vector will be loaded into the input registers and thus we are able to set the inputs of the combinational logic to any value that is required to detect the fault.

Now, consider that some fault can only be detected by input combinations that assert the precomputation logic. In this case, we build the first input vector such that precomputation is disabled and load the desired values to the registers disabled by precomputation. Next, the second input vector needs only have the correct values for the remaining registers since the first registers will already have the correct values and will not be disturbed since the precomputation logic will be active.

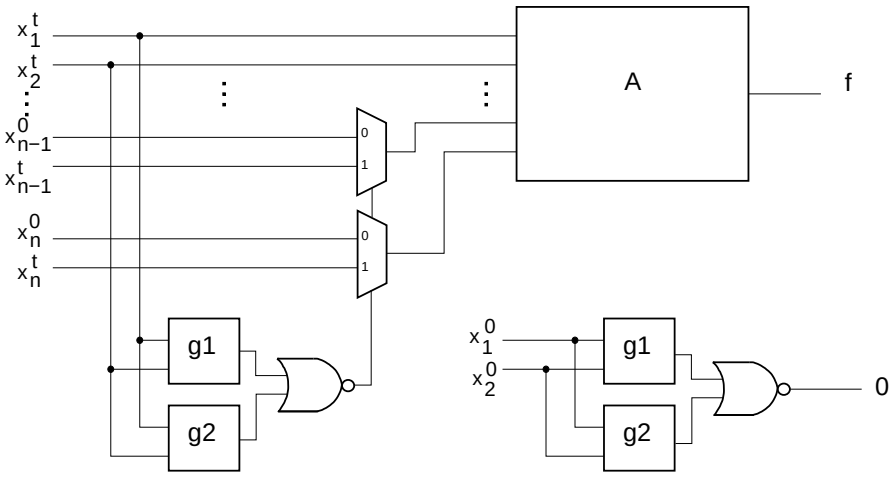

Figure 3 Circuit transformation.

4.4 Generating the Two Input Vector Sequence

We present a circuit transformation technique for the automatic generation of the two input vector sequence used in the testing of power managed circuits. This will allow us to use combinational ATPG techniques, thus avoiding the more computationally expensive sequential test generators (Abramovici, Breuer & Friedman 1990).

The proposed transformation is shown in Figure 3, obtained from the precomputed circuit of Figure 2. We have duplicated the inputs x of the original circuit. x^0 corresponds to the first input vector and x^t to the second.

The subset of inputs used in the precomputation logic (in the case of Figure 2, x_1 and x_2) will always be present at the input of the combinational logic, therefore this subset of x^t goes directly to block A after transformation.

The remaining inputs of the second input vector x^t go to the precomputation logic and the output of this logic will decide if these values of x^t (no precomputation for x^t) or those of the first input vector x^0 (circuit being precomputed for x^t) are the inputs to A.

Note that the values x^0 corresponding to the inputs used in the precomputation logic are not defined by the transformation. Yet, for the transformation to make sense, we have to make sure that they disable the precomputation logic so that x^0 can be loaded to the registers. This condition is also shown in Figure 3.

We can now run standard ATPG tools on the circuit after transformation to obtain values for x^0 and x^t that detect any fault in the original combinational logic block A and precomputation logic g_1 and g_2. However, we have duplicated the number of inputs, increasing dramatically the input search space of the ATPG tool. Additionally, the circuit after transformation will have a significant amount of redundancy, further complicating the problem for the ATPG tool. In the next section we describe an ATPG tool that can efficiently handle this problem.

5 ALGORITHMS FOR ATPG

As described in the previous sections, data-dependent power management based on precomputation can introduce a large number of redundant faults. In addition, the solution of using two input vector sequences for detecting faults in the resulting circuit potentially duplicates the search space. As a result, circuits using data-dependent power management are expected to be significantly harder for test pattern generation tools. This is indeed the case and traditional ATPG algorithms are likely to be unable to yield acceptable fault coverages for circuits using data-dependent power management. In particular, this is the case with the D-algorithm, PODEM, FAN and SOCRATES (Abramovici et al. 1990) and with recent implementations of these algorithms (Lee & Ha 1993).

Nevertheless, given the relationship between Propositional Satisfiability (SAT) and ATPG, recent work on efficient search algorithms for SAT (Silva & Sakallah 1996) can potentially enable the development of ATPG algorithms specifically suited for circuits with many hard-to-detect faults.

5.1 Satisfiability-Based ATPG

It is well-known that fault detection problems can be cast as instances of SAT (Larrabee 1992, Stephan, Brayton & Sangiovanni-Vincentelli 1996). Basically, the valid assignments to the nodes of a circuit can be represented by a Conjunctive Normal Form Formula (CNF). For ATPG, we just need to consider two replicas of a given circuit, one denoting the *good* circuit and the other denoting the *faulty* circuit (on which the given fault must be activated). By creating the OR of the XOR's of the primary outputs of the two circuits, and by requiring the output of the OR gate to assume value 1, we create a satisfiability problem whose solution is a test pattern for the given fault (Larrabee 1992). In general, additional information is added to the CNF representation in order to prune the amount of search.

It is generally accepted that SAT-based ATPG algorithms have a few significant drawbacks. First, representing each fault detection problem as an instance of SAT is extremely time-consuming. Indeed, known experimental results indicate that CNF formula creation can take as much as 75% of the total testing time (Stephan et al. 1996). Second, since all clauses in a CNF formula must be satisfied, test patterns may become over specified. Consequently, SAT-based ATPG algorithms may yield test sets larger than necessary.

Despite these drawbacks, SAT-based ATPG algorithms are particularly versatile in that improvements to SAT algorithms can be readily applied and extended for ATPG.

5.2 The GRASP SAT Algorithm

The GRASP SAT algorithm is detailed in (Silva & Sakallah 1996), and it is basically a backtrack search algorithm. However, GRASP is able to analyze the causes of conflicts, i.e. situations of the search in which one or more clauses have all literals set to 0. Analysis of the causes of conflicts can be used for implementing several powerful pruning techniques:

- By analyzing the causes of conflicts we can backtrack directly to the cause of each conflict. Hence, GRASP implements *non-chronological backtracking*.
- Given that the causes of conflicts are identified, they can be *recorded* as new clauses, and so these new clauses can be used to augment the original CNF formula. Hence, we can prevent known conflicts from being identified again during the search.
- Careful analysis of the structure of conflicts permits identifying variable assignments which are deemed necessary for a solution to be found. Since GRASP identifies more necessary assignments due to the causes of conflicts, the search is further pruned.

As illustrated in (Silva & Sakallah 1996), new pruning techniques can be easily incorporated into GRASP. In addition, preliminary experimental results strongly suggest that GRASP is one of the most efficient SAT algorithms for highly structured instances of SAT.

Moreover, instances of SAT obtained from fault detection problems for stuck-at or bridging faults are highly structured, and GRASP performs particularly well on these benchmarks. As a result, circuits having a significant number of hard-to-detect faults are potentially amenable for a ATPG tool based on GRASP. The experimental results given in Section 6 strongly support this motivation.

The proposed ATPG algorithm, named TG-GRASP, basically encodes fault detection problems using the approach given in (Stephan et al. 1996) and uses GRASP as the back-end search engine. In addition, a few additional features have been incorporated into TG-GRASP:

- To prevent the over-specification of test patterns, TG-GRASP implements *syntactic* satisfiability, which permits early identification of sufficient conditions for satisfiability before satisfying all clauses in the CNF formula. This technique can be viewed as equivalent to restricted forms of dynamic head line identification that can be identified by circuit-based ATPG tools (Silva & Sakallah 1994).
- Because GRASP records new clauses during the search as by-product of conflict analysis, some of these clauses, in particular the ones solely associated with variables in the good circuit, can be re-used for subsequent faults, thus potentially pruning the amount of search for subsequent fault detection problems. We refer to these clauses as *pervasive* clauses.

Syntactic satisfiability reduces computation time for detectable faults, whereas pervasive clauses, because they add additional constraints to the search, potentially reduce the computation time for all faults.

6 EXPERIMENTAL RESULTS

In this section we compare the testability of a subset of the circuits in the MCNC'91 combinational benchmark set before and after adding data-dependent power management. In Table 1 we present the statistics for the circuits used. Under the column *Original* we give the number of primary inputs (PI), the number of primary out-

Table 1 Statistics of the circuits and power reduction through precomputation.

Circuit	Original					Precomp.		%
Name	PI	PO	Gates	Lits	Power	Lits	Power	Red.
9symml	9	1	157	327	1837.1	40	1487.4	19.0
apex2	39	3	195	390	2322.7	4	1201.2	48.3
comp	32	3	105	188	1712.6	13	720.9	57.9
comp16	35	3	221	413	2597.8	17	907.9	65.1
cps	24	102	1001	1979	4518.8	26	2879.5	36.3
dalu	75	16	827	1611	7014.2	20	3638.4	48.1
duke2	22	29	351	716	2243.7	20	1527.6	31.9
e64	65	65	251	379	2554.7	5	573.7	77.5
i2	201	1	156	369	7748.5	22	2520.0	67.5
misex3	14	14	533	1111	3473.1	2	2503.9	27.9
seq	41	35	1246	2590	7494.2	0	3923.1	47.7
too_large	38	3	234	471	2396.4	1	1561.1	34.9

puts (PO), the number of gates (Gates) and literals (Lits) in the circuit and its power dissipation (Power).

Also in Table 1, we show the power savings obtained after precomputation is applied to each circuit. Under *Precomp.*, we give the number of literals in the precomputation logic and the power of the power managed circuit. We can see that the size of the precomputation logic is in general much smaller than the original circuit. The last column of Table 1 indicates the percentage savings in power obtained through precomputation. As we can observe, power reductions of upto 77% are possible.

To measure the testability of the circuits, we have used two SAT-based ATPG tools, TEGUS (Stephan et al. 1996) and TG-GRASP, described in Section 5. The CPU times reported are for a Sun 5/85 machine with 64 MByte of physical memory. All tools were compiled with the same optimization options.

In Table 2 the ATPG results for the original MCNC'91 benchmark circuits are shown. The transformation described in Section 4.4 was applied to the precomputed circuits and the ATPG programs were run on the modified circuit. The ATPG results after precomputation are given in Table 3 In each table **F**, **D**, **R**, **A** and **CPU** denote, respectively, the total number of faults, the number of detected faults, the number of redundant faults, the number of aborted faults and the CPU time for each tool. For each benchmark *all* faults are targeted. This solution permits a larger set of faults to be studied and guarantees that each ATPG tool is presented with exactly the same set of faults.

Table 2 ATPG results for the MCNC original benchmark circuits.

Circuit Name	**F**	TEGUS				TG-GRASP			
		D	**R**	**A**	**CPU**	**D**	**R**	**A**	**CPU**
9symml	752	750	2	0	15.9	750	2	0	18.3
apex2	948	945	3	0	77.1	945	3	0	25.3
comp	480	479	1	0	6.2	479	1	0	6.6
comp16	960	960	0	0	15.7	960	0	0	17.2
cps	4642	4640	2	0	146.6	4640	2	0	199.5
dalu	3742	3740	1	1	313.8	3740	2	0	201.0
duke2	1708	1708	0	0	41.7	1708	0	0	53.7
e64	1142	1136	0	0	27.3	1142	0	0	34.6
i2	762	760	2	0	17.1	760	2	0	18.7
misex3	2590	2583	7	0	105.8	2583	7	0	130.2
seq	5912	5908	4	0	1375.4	5908	4	0	378.6
too_large	1132	1117	15	0	27.5	1117	15	0	35.0

For both the original and precomputed circuits, TEGUS and TG-GRASP have comparable and acceptable performance in all circuits. Nevertheless, for a few circuits the pruning techniques used in TG-GRASP make the difference and lead to reasonably smaller CPU times. Note that for benchmark *dalu*, TEGUS aborts one fault. These results, and the fact that TG-GRASP aborts no faults, illustrate the *robustness* of the TG-GRASP algorithmic solution. Even though precomputation introduces a large number of redundant faults, there exist ATPG tools which can test the resulting circuits with 100% fault coverage.

REFERENCES

Abramovici, M., Breuer, M. & Friedman, A. (1990). *Digital Systems Testing and Testable Design*, IEEE Press.

Alidina, M., Monteiro, J., Devadas, S., Ghosh, A. & Papaefthymiou, M. (1994). Precomputation-Based Sequential Logic Optimization for Low Power, *IEEE Trans. on VLSI Systems* **2**(4): 426–436.

Benini, L., Siegel, P. & Micheli, G. D. (1996). Automatic Synthesis of Low-Power Gated-Clock FSMs, *IEEE Trans. on Computer-Aided Design* **15**(6): 630–643.

Chandrakasan, A., Sheng, T. & Brodersen, R. (1992). Low-Power CMOS Digital Design, *IEEE Journal of Solid-State Circuits* **27**(4): 473–484.

Larrabee, T. (1992). Test Pattern Generation Using Boolean Satisfiability, *IEEE Trans. on*

Table 3 ATPG results for the MCNC benchmark circuits with precomputation.

Circuit Name	**F**	TEGUS				TG-GRASP			
		D	**R**	**A**	**CPU**	**D**	**R**	**A**	**CPU**
9symml	932	890	42	0	160.2	890	42	0	65.6
apex2	1492	1243	249	0	142.1	1243	249	0	79.5
comp	956	749	207	0	28.1	749	207	0	26.0
comp16	1448	1228	220	0	173.8	1228	220	0	53.4
cps	4996	4844	152	0	457.9	4844	152	0	351.2
dalu	4794	4274	519	1	513.1	4274	520	0	468.1
duke2	1970	1847	123	0	85.1	1847	123	0	104.9
e64	2122	1686	430	0	128.0	1692	430	0	139.0
i2	3696	2371	1325	0	425.1	2371	1325	0	357.7
misex3	2792	2696	96	0	145.9	2696	96	0	170.1
seq	6536	6259	277	0	838.7	6259	277	0	593.4
too_large	1706	1430	276	0	74.9	1430	276	0	92.5

Computer-Aided Design **11**(1): 4–15.

Lee, H. & Ha, D. (1993). On the Generation of Test Patterns for Combinational Circuits, *Technical Report 12/93*, Department of Electrical Eng., Virginia Polytechnic Institute and State University.

Monteiro, J., Devadas, S., Ashar, P. & Mauskar, A. (1996). Scheduling Techniques to Enable Power Management, *Proceedings of the* 33^{rd} *Design Automation Conference*, pp. 349–352.

Monteiro, J., Rinderknecht, J., Devadas, S. & Ghosh, A. (1995). Optimization of Combinational and Sequential Logic Circuits for Low Power Using Precomputation, *Proceedings of the 1995 Chapel Hill Conference on Advanced Research on VLSI*, pp. 430–444.

Najm, F. (1993). Transition Density: A New Measure of Activity in Digital Circuits, *IEEE Trans. on Computer-Aided Design* **12**(2): 310–323.

Silva, J. & Sakallah, K. (1994). Dynamic Search-Space Pruning Techniques in Path Sensitization, *Proceedings of the* 31^{st} *Design Automation Conference*, pp. 705–711.

Silva, J. & Sakallah, K. (1996). GRASP—A New Search Algorithm for Propositional Satisfiability, *Proceedings of the International Conference on Computer-Aided Design*, pp. 220–227.

Stephan, P., Brayton, R. & Sangiovanni-Vincentelli, A. (1996). Combinational Test Generation Using Satisfiability, *IEEE Trans. on Computer-Aided Design* **15**(9): 1167–1176.

Tiwari, V., Ashar, P. & Malik, S. (1995). Guarded Evaluation: Pushing Power Management to Logic Synthesis/Design, *Proceedings of the International Symposium on Low Power Design*, pp. 221–226.