Design for testability of gated-clock FSMs

M. Favalli, L. Benini* and G. De Micheli*

DEIS - University of Bologna - Viale Risorgimento, 2, 40136 Bologna, Italy
* CIS - Stanford University - Stanford CA-94305 USA

Abstract

Gated clocks allow significant power savings in synchronous systems, but are generally considered an unsafe design practice because they decrease testability. In this paper we present two methodologies that guarantee full single-stuck-at testability for gated-clock finite-state machines. The first technique, increased observability, can be used in conjunction with redundancy-removal techniques to obtain fully-testable gated clock FSMs with high performance. The second technique, increased observability and controllability, is applicable to large FSMs for which redundancy removal is not possible and produces fully-testable gated-clock FSMs with a moderate decrease in performance.

1 Introduction

Recently, the portable consumer electronics market has undergone a period of explosive growth. Constraints on battery operation times have forced designers to consider power dissipation as a major design parameter. Even for high-performance non-portable system, the increasing cost of packaging and cooling systems imposes stringent requirements on the maximum allowable power consumption.

Several techniques have been proposed for the reduction of the power consumption in digital systems, and good overviews can be found in [1, 3, 4]. Numerous approaches are based on the down-scaling of the power supply voltage. Unfortunately, the choice of the supply voltage is often not under the designer's control. In such cases the power consumption in CMOS circuits can be reduced by decreasing the switching activity and/or the switched capacitance.

In synchronous systems, a common way to reduce the circuit activity is to stop the local clock of inactive subsystems. The local clock must be *gated* with an activation signal. The dynamic power management schemes implemented in many state-of-the-art microprocessors [9, 10, 11] and signal-processing systems [1, 2] heavily rely on gated clocks to achieve low power consumption. Circuits with gated clocks can be designed by skilled designers or automatically synthesized [7, 5, 6].

Unfortunately, the extensive use of gated clocks creates major difficulties in testing and verification. The circuitry used to gate the clock is functionally redundant: it is not needed for correct functionality, but its failure can have disastrous effects. Intuitively, if the circuit performs correctly, the gating circuitry is transparent. In the presence of faults, some malfunctioning in the gating circuitry may be masked by the controlled sub-system or vice-versa. This behavior poses relevant testability problems.

The concerns about the diminished testability of circuits with gated clocks have been a major obstacle to the diffusion of dynamic power management schemes. In this work we describe a set of methodologies that allow the designer to produce gated-clock circuits with high testability. We show that the synthesis of testable circuits with gated clocks does not impose large overhead or performance losses with respect to the original implementation.

We also explore the subtle relationship between reduction in testability and logic redundancy introduced by gated clocks, and show that highly testable gated-clock circuits may have better performance in speed, area and power consumption than their low-testability counterparts.

Although we will base our analysis on the implementation style presented in [7, 8], our conclusions are general and can be extended to other dynamic power management schemes. Our work focuses on sequential systems described as FSMs. Pipelined circuits without feedback, such as those found in signal processing systems, can be treated as particular cases in our methodology.

2 Power management schemes

It is a well known fact that large sub-blocks of digital systems may be unused for a substantial fraction of the total operation time. If the result of the computation performed by a sub-block during a given clock cycle is not necessary for the correct functional behavior of the system, the sub-block is *idle*. During idle cycles power is wasted not only on the clock lines, but also in the logic that performs computation whose result is not needed.

In synchronous systems, the useless power dissipation can be eliminated by selectively stopping (gating) the clock in portions of the circuit where useful computation is not being performed. Gated clocks are commonly used in large power-constrained systems [9, 10, 11] as the core of dynamic power management schemes. Notice however that it is usually responsibility of the designer to find the conditions that disable the clock. Three approaches to the automatic synthesis of logic circuits that can be conditionally disabled by environmental signals have recently been reported [5, 6, 8]. The results obtained by these methods are promising: power reductions ranging between 10% and 70% have been reported on standard benchmark circuits.

In [5] Alidina et al. have described a *pre-computation-based approach,* where a block of redundant logic is added before the flip-flops that separate two successive stages of a sequential circuit. The function of the pre-computation logic is to disable the flip-flops at some of the inputs of the second stage if the result of operations that it must perform in the next clock cycle can be computed using only part of the inputs.

The pre-computation approach has been extended to deal with combinational circuits: in [6] Tiwari et al. showed that it is possible to selectively disable parts of a combinational logic network without being restricted to stop the computation only at latch boundaries. This extended pre-computation strategy has been called *guarded evaluation.*

In [7, 8] the authors have presented a method for the automatic synthesis of FSMs with gated clocks. Our approach is based on the observation that self-loops on the STG of a Moore machine are idle conditions, because the state and the output do not change when the machine is in a self-loop. A block of redundant logic can then be synthesized whose function is to stop the clock of the machine when a self-loop is detected observing the input and state lines.

Despite some important differences, all dynamic power management schemes based on gated clocks rely on the presence of a redundant logic block whose purpose is to stop the clock and to guarantee functional equivalence with the original system. The size, delay and power of the clock-stopping circuitry must be much smaller than the size of the controlled system.

In this work we will adopt the implementation style

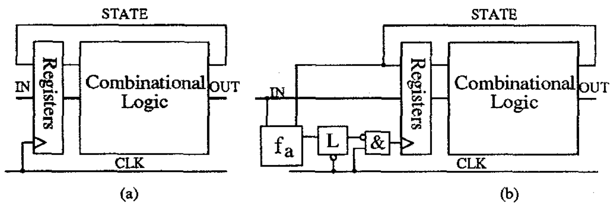

Figure 1: **(a) Single clock, flip-flop based finite-state machine. (b) Gated clock version.**

presented in [8]. The model of a FSM with gated clock is shown in Figure 1 together with the original FSM without gating on the clock. The block labeled "L" in Figure 1 represents a level-sensitive latch, transparent when the global clock signal is low. Its presence is needed to avoid malfunctioning due to glitches on the output of f_a [7].

The combinational function of input and state variables that stops the clock when the FSM is idle is called *activation function.* The activation function f_a detects when the machine is in a self-loop. Since we want to keep the size of the activation function's implementation as small as possible, we may choose only a subset of all self-loops to be included in the ON-set of f_a. The chosen self-loops must have high probability, in order for the activation function to stop the clock with maximum efficiency. A probabilistic algorithm that select optimal activation functions is described in [8]. In this work we focus on the testability properties of a FSM with gated clock. Our starting point will be the gated-clock FSM implementation including the activation function, the flip-flops and the combinational logic block.

Example 1 *The FSM in Figure 2(a) is implemented by the synchronous network shown in Figure 2(b). The FSM has two self-loops. Clocking the machine when it is in a self-loop is useless, because the output and the state do not change (in this case output and state are the same). Notice that even if the output and state do not change, clocking the machine dissipates power on the clock line, the flip-flops and possibly on the internal capacitances of the combinational logic.*

Assume that the algorithms for the synthesis of the activation function decides to stop the clock only in the self-loop leaving state 1 (shaded in Figure 2(a)). The resulting gated-clock FSM is shown in Figure 2(c).

It is important to observe that when $f_a = 1$ the inputs of the combinational logic of the FSM do not change. Consequently, all input-state configurations in the ON-set of f_a are not propagated to the input of the combinational logic of the FSM. In other words, the ON-set of f_a becomes an additional *controllability don't care*

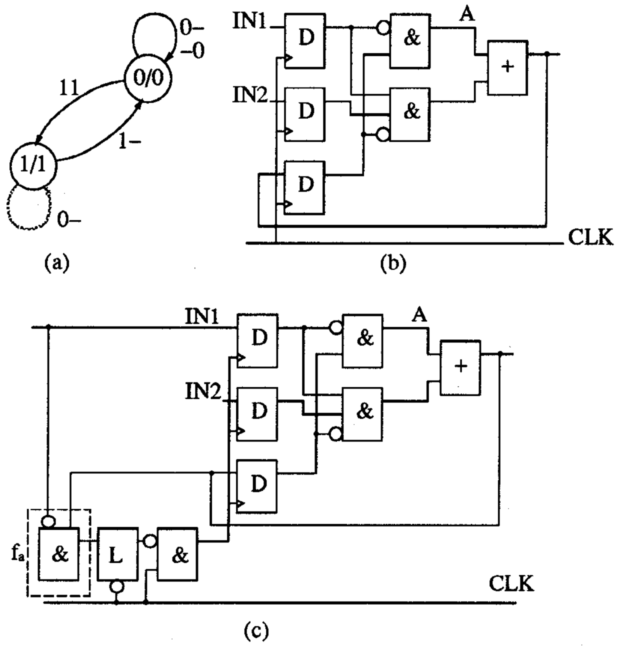

Figure 2: **(a) STG of a simple two-state FSM. (b) Implementation of the FSM. (c) Gated-clock implementation.**

set for the FSM logic, and can be used to simplify its implementation. This is an important property from a practical point of view, because it allows the designer to recover some of the area used for the activation function, obtaining a smaller circuit that will likely have diminished power dissipation.

Notice however that this optimization is an optional step in the synthesis procedure of gated-clock FSMs. The *don't care set* represents a degree of freedom in the implementation of a logic function. The designer may choose to avoid using it for many reasons such as reuse of optimized macro blocks or insufficient computational resources. The use of f_a as *don't care* set has important implications when testability is considered, as we will discuss in following sections.

3 Testable design of Gated-Clock FSMs

In the case of gated-clock FSMs, the clock is stopped only when the outputs of the FSM do not change, therefore there is complete input-output equivalence between the original FSM and its gated-clock counterpart.

We adopt the *stuck-at* logical fault model to represent physical faults. Moreover, we focus on static single-fault testability, disregarding transient, intermittent and multiple faults. In this work, a *fully testable* FSM is one in which there exist a test sequence that reveals any given single *stuck-at* fault.

The automatic synthesis of fully testable FSMs is considerably more difficult than the synthesis of testable combinational logic. Nevertheless effective tools have been developed and are commonly used in industrial and academic [15, 16] environments.

In many practical cases, full testability for single *stuck-at* faults is considered a minimum safety requirement. We show that the addition of clock-gating circuitry makes the FSM not fully testable. Assume that we have modified our original FSM implementation by adding the clock-gating circuitry. Let us consider a *s-a-0* fault on the output wire of the activation function. In this case, the clock will always be enabled, and the FSMs will have the same behavior as in the original implementation. Observing the outputs, it is not possible to detect the fault. In general, we cannot reveal a fault ϕ that transforms the activation function f_a into a faulty function f_a^ϕ with a smaller ON-set.

$$f_a^\phi \subseteq f_a \rightarrow \phi \textit{ is untestable} \qquad (1)$$

This property implies the existence of at least one untestable fault in the gated-clock FSM (the *s-a-0* on the output of f_a). The gated-clock FSM is *never* fully testable.

While the existence of untestable faults in the activation function logic is quite intuitive, it may be possible to overlook that the insertion of clock-gating circuitry can decrease the testability of the combinational logic of the FSM as well. The activation function is active (one) in a sub-set of the self-loops. If a fault in the original FSM can be tested only with input sequences that imply traversing a self-loop in the ON-set of the activation function, the fault becomes untestable in the gated-clock FSM. This statement can be clarified through an example.

Example 2 *In the FSM of Figure 2(b), consider a s-a-0 fault on wire A. In the original FSM, the fault is testable. Assuming that the initial (reset) state is 0, an input sequence that reveals the fault is* (11, 01).

The s-a-0 fault on A is untestable in the gated-clock version of the FSM shown in Figure 2(c). The input value required at the AND gate to activate the fault is 01, *but this value never appears on the output of the flip-flops. This is because the activation function is high when IN1 and the state are both one. When the activation function is high the clock is stopped and the value required for the activation of the fault is not propagated to the output of the flip-flops.*

We call I the set of possible input values for the combinational logic of the FSM. The presence of the activation function reduces the size of set I. In other words, the controllability *don't care* set for the combinational logic of the FSM is increased, and we can exploit only a subset of the STG arcs to generate test vectors.

In application where testability is a primary requirement, the untestable faults generated by the clock-gating logic are not acceptable. We will show that it is possible to generate testable gated-clock FSMs with minimum overhead and no loss in performance. We will assume that the original FSM is fully testable, because in the following discussion we want to focus only on untestable faults that are created by the insertion of the activation function.

3.1 Increasing observability

We first address the problem of the *s-a-0* untestable fault on the output of f_a. Since we have shown that it is impossible to propagate the effect of such fault to the output, the only way to solve the problem is to increase the observability. If we make the output of the activation function observable, the *s-a-0* fault becomes trivially testable: any input-state configuration in the ON-set of the activation function is a valid test vector.

The penalty of increasing the observability is due to additional wiring needed to route the output of f_a to the closest observation point. This penalty is not excessive, especially for large FSM that are often idle (for which the activation function allows substantial power savings). We call *observability increase* the addition of one observable output in the FSM. Notice that the combinational logic of the FSM and the activation function are not modified. We now discuss the effect of the observability increase on the activation function logic.

The observability of f_a makes the generation of tests for faults in f_a not harder than the test generation for faults in the combinational logic of the original FSM. This can be intuitively understood observing that the inputs of the activation function are exactly the same as those of the combinational logic of the original FSM (anticipated by one clock cycle). Notice that the inputs of the activation function do not come from the output of conditionally enabled flip-flops, therefore they can assume any value in the original set of allowed input values. If the activation function is synthesized without internal redundancies, the presence of the additional observation point is sufficient to guarantee its full testability.

Once the activation function has been synthesized, we want to generate a test set for the gated-clock FSM with observability increase. This cannot be done using standard ATPG programs, because they assume a standard synchronous implementation, where clock-gating is not allowed. We propose here a *multiplexed model* of the gated-clock FSM: it is a sequential circuit with standard clock distribution that is equivalent for test generation purposes to the gated-clock FSM.

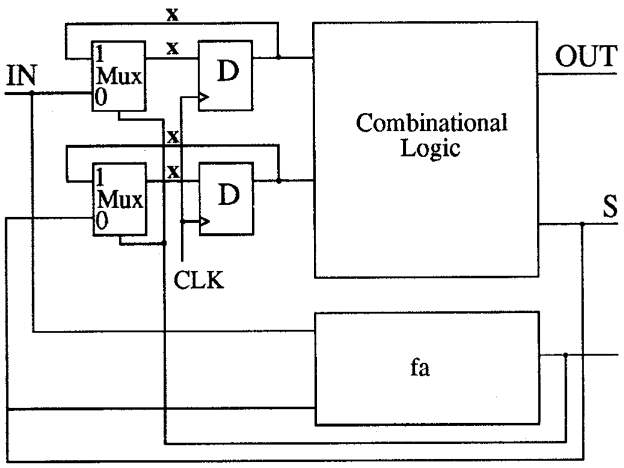

Figure 3: **Multiplexed model of a gated-clock FSM.**

The multiplexed model of a gated-clock FSMs can be obtained explicitly accounting for the effect of f_a in the equations of the flip-flops:

$$y^{n+1} = D^n f_a + y^n f_a' \tag{2}$$

that is the equation of a multiplexer with f_a as control variable. We can then obtain a synchronous network without gated-clock by eliminating the circuitry at the output of f_a, placing the multiplexers at the inputs of the flip-flops and clocking the flip-flips with the global clock (CLK). The structure of the multiplexed model for the FSM with increased observability is shown in Figure 3. Notice that the insertion of the multiplexers requires the addition of two additional wires for each state and input variable, marked with "X" in Figure 3. Faults on these wires do not correspond to any actual fault in the gated-clock FSM.

The main advantage of the multiplexed model for gated-clock FSMs is that standard sequential ATPG programs can be used for test generation. The multiplexed model can easily be automatically generated from the gated-clock FSM implementation, then fed to the sequential ATPG program with no need of dedicated tools or error-prone hand-crafted procedures.

3.2 Eliminating redundancy

While the use of an additional observation point solves the problem of testing the activation function, it still does not guarantee full testability of the gated-clock FSM, as we can see in the following example.

Example 3 *Consider the gated-clock FSM of Figure 2(c). Requiring the observability of f_a guarantees the testability of the s-a-0 fault on its output. Unfortunately, the s-a-0 fault on line A is still untestable. The only input-state configuration that reveals it is never observed by the inputs of the combinational logic of the FSM, because the activation function freezes the clock when it occurs at flip-flop inputs.*

The gated-clock FSM with increased observability is not fully testable because even if the additional observation point makes the activation function irredundant, it does not improve the testability of the combinational logic in the FSM. The activation function reduces the set I of allowed input values to a set $I' \subset I$. We assumed full testability of the original FSM when the full I can be used for test generation, but nothing can be said on testability with respect to I'. Insertion of more observability points is not a viable solution, because the untestable faults are caused by the lack of fault activation conditions.

Fortunately, we can solve this problem employing the same tools used for the synthesis of the original fully testable FSM. Synthesis for testability of FSMs can be performed using standard *redundancy removal* techniques. Given a synchronous network, the redundancy removal algorithm attempts test generation for all faults in the network, using a collection of ATPG algorithms, with increasing computational requirements. In this approach, faults that are not detectable after the application of a technique are inserted into the fault list for the next more powerful ATPG algorithm.

As a last resource *exact* state enumeration techniques [16] are used. If the fault is proven untestable, the wire with the untestable fault is replaced by a constant corresponding to the type of fault. The constant is then propagated if possible, to simplify the transitive fan-out network and eliminate other untestable faults.

The approach above outlined can be applied to the multiplexed model associated with the gated-clock FSM (with increased observability). The exact algorithm will locate and remove all redundancies in the FSM's combinational logic, making the gated-clock FSM fully testable. Notice that redundancy removal is always beneficial for performance: if the constant value is propagated, we may be able to reduce the number or the complexity of gates in the transitive fan-out of the wire replaced by a constant. Moreover if the wire replaced by a constant is the only fan-out stem of a gate, the gate can be eliminated without changing the functional behavior.

Example 4 *In the gated-clock FSM of Figure 2(c) the s-a-0 fault on wire A is untestable. The redundancy removal algorithm detects it ad replaces wire A with a connection to the constant value 0. The constant can then be propagated to further simplify the circuit. The OR gate on the output can then be eliminated, and the and gate whose output has been blocked at 0 can be removed as well.*

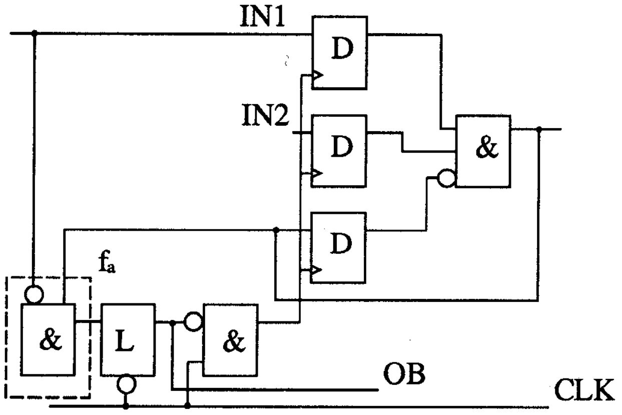

Figure 4: **Optimized and fully-testable gated-clock FSM with increased observability**

The final optimized circuit is shown in Figure 4. It is not only fully testable, but has also better performance that the original gated-clock FSM, in terms of area, delay and power dissipation.

It is important to notice that the multiplexed model contains faults not corresponding to actual faults in the gated-clock implementation. The multiplexers shown in Figure 3 on the inputs and state variables are inserted only for the purpose of modeling the effect of the activation function. The fault at the outputs of the multiplexers and at the inputs that are selected when the activation function is one should not be inserted in the initial fault list, because they do not correspond to any actual wire in the gated-clock implementation. We call these faults *external faults*. The exclusion of external faults from the initial fault list is not necessary for the correctness of our procedure, but it is important for efficiency reasons.

It is also important to notice that the redundancy removal algorithm can be applied to the multiplexed model only if the activation function is made observable. If this is not the case, the activation function becomes redundant, and it will most likely be removed.

In the previous section we mentioned the possibility of using the activation function as additional *don't care* set for the synthesis of the FSM. Conceptually, this operation is similar to redundancy removal: instead of removing logic in a post-processing step, the synthesis algorithm exploits the *don't care* information to avoid

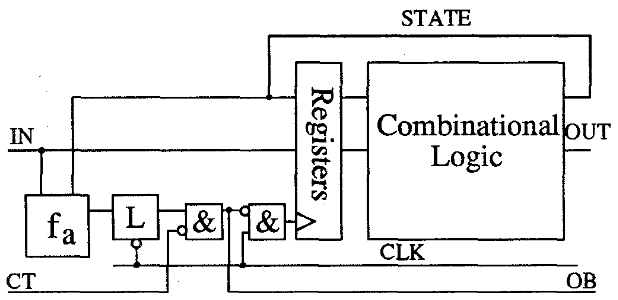

Figure 5: **Gated-clock FSM with increased controllability and observability.**

the synthesis of useless logic.

Practically, however there is an important difference. Since *don't care-based* optimization is computationally expensive, reduced *don't care* sets are often employed [13]. Generally, *don't care-based* optimizations improve the testability of the gated-clock FSM reducing logic redundancies, but do not guarantee full testability, unless the full *don't care* set is applied on every gate of the final mapped network. In this case *don't care* optimization becomes equivalent to redundancy removal, but it is generally much less efficient [14].

3.3 Increasing controllability

Eliminating redundancy from a gated-clock FSM with increased observability is an effective way to obtain fully testable implementation with improved performance. Unfortunately, there are two important cases in which this procedure cannot be applied.

First, in many applications the combinational logic of the machine consists of blocks that cannot be modified. This is often the case when gated-clock synthesis is applied to data-path circuits: the combinational logic may consist of adders, multipliers, comparators or other standard components implemented by highly optimized cells that the designer is not allowed to modify. Faults inside the standard components may become untestable and they cannot be removed.

Second, the redundancy removal process may become very computationally expensive. Even if symbolic techniques for FSM traversal are effective for small and medium sized circuits, they still are impractical for large circuits or for classes of moderate sized circuit for which the BDD representation is exponentially large. If the redundancy removal algorithm fails, we do not have guarantees on the testability of our implementation. This is an unacceptable limitation for a general design methodology.

This problem can be avoided by adding an extra controllability input CT that inhibits the effects of the clock gating, as shown in Figure 5. When such input is set at logic 1, the combinational part of the FSM can be tested without worrying about the effects of the activation function. When the activation function becomes active, the clock is not stopped if CT $= 1$, and the flip-flops of the FSM sample the input and state value. Thus, the allowed input set I is exactly the same as the one of the original FSM. The availability of the complete I guarantees that no untestable faults exist in the combinational logic (under the assumption of full testability for the original FSM). If the activation function is observable (through the observability output OB), we can also guarantee the full testability of f_a, as discussed in the previous section.

The insertion of an additional controllability input has two advantages. First, since we do not need to modify the combinational logic of the FSM in the gated-clock version, the test set developed for the original FSM can be used to test the gated-clock FSM as well. More test vectors can just be appended to fully test the activation function. Second, while testing the gated-clock FSM without added controllability may be substantially harder than testing the original FSM, the test generation process in the added controllability case is generally very efficient. The reason for this difference is that a large amount of time is spent in the first case to prove the untestability of redundant fault, and to subsequently remove the redundant wires, while the FSM with added controllability does not have redundant faults (if the original FSM is fully testable).

The price paid when adding controllability is obviously in increased number of inputs and in the need to access the controllability input during testing. Notice also that during testing the power dissipated by the gated-clock FSM is larger than the power consumed in the original FSM, because the clock is not gated and the activation function dissipates additional power. Another disadvantage is that one more gate is on the critical path in the activation function.

In summary, adding controllability to the gated-clock FSM (with increased observability) makes the test generation task not harder than in the original FSM, while still allowing substantial power savings during normal operation. The performance penalty for this implementation style is quite small, while the main disadvantage is in the increased number of inputs and outputs (one additional input and output).

4 Experimental results

The methodology for the design of gated-clock FSMs described in section 2 has been applied to the MCNC logic synthesis sequential benchmarks. We report here the results on a subset of the benchmark suite that rep-

benchmark	test. no	CPU
bbtas	41	0.6
bbara	178	10.6
bbsse	177	72
lion9	104	3.4
keyb	281	257
styr	146	168
s420	218	108
test	157	7.6
scf	934	693

Table 1: **Test length and CPU time for the original (non-gated) FSMs.**

resents different kinds of FSMs. Benchmarks such as **bbsse** and **bbara** are *reactive* FSMs with many self-loops and activation functions with large ON-set. We included larger FSMs such as **scf** and **styr** to give a flavor of the problems encountered in the test generation process when the circuit complexity increases.

The combinational logic of the FSM and the activation function of each benchmark have been mapped on a CMOS library. We have then automatically generated the multiplexed model of the gated-clock FSM needed in the ATPG step.

We first generated complete test sets for the FSMs implementations without gated clock using the sequential ATPG algorithms [15] included in SIS [12]. The test vectors have been generated assuming the availability of a reset state for each flip-flop, as required by the ATPG algorithms provided by SIS. The length of the test set and the CPU time in seconds (on a SUN-station IPX) are reported in Table 4. The redundancy removal algorithm provided by SIS was applied after test generation to obtain full testability (100% fault coverage).

We have then generated the test vectors for the gated-clock FSMs. The multiplexed models used for test generation in the case of added observability have multiplexers whose operation is functionally equivalent to the gating of clock. Notice that these multiplexer are not present in the real circuit, and must not be accounted for in the fault coverage of the synchronous network. Even if the multiplexers are not used in the actual circuits, the current implementation of SIS ATPG does not allow to exclude faults from the initial fault list.

The last three columns of Table 4 show the achieved results. First, the computation time needed to generate the test sets is substantially increased, because of the presence of untestable faults in the combinational logic of the FSMs. The fault coverage, evaluated without taking into account the faults in the multiplexers, is not 100%, showing that in all cases the activation function actually creates redundancies. If we have the freedom to modify the combinational logic of the FSM, we can employ the redundancy elimination procedure provided by SIS to obtain fully testable circuits. The performance improvement obtained by eliminating redundancy is strongly dependent on the number of untestable faults. The high fault coverage obtained (because don't cares are partially exploited during synthesis) shows that only a small number of redundancies can be eliminated, and the performance improvement is generally a second-order effect.

The second and third columns of Table 4 show the size of the test sets and the test generation times for the gated-clock FSMs with increased controllability and observability. These FSMs are fully testable by construction, because they are generated starting from fully testable implementations without clock-gating. The number of test vectors required to fully test these FSMs is larger than the test length requited for the original implementation. This increase is caused by the need of generating tests for the activation function logic. The computation times show clearly that the test generation process for the gated-clock FSMs with increased controllability and observability is not harder than in the original implementation.

5 Conclusions

This work introduced two transformations that increase the testability of gated-clock circuits. The first transformation, namely *increased observability*, allows the synthesis of fully testable gated-clock circuits with improved performance. An observation point is added to make the activation function logic fully testable. A redundancy removal step follows, whose purpose is to eliminate untestable faults in the combinational logic of the FSM. The test generation and redundancy removal steps for *increased observability* circuits are computationally expensive and should be applied in aggressive implementations.

The second transformation, namely *increased controllability and observability*, guarantees the full testability of the gated-clock circuit if the initial implementation was fully testable. This transformation is slightly more expensive than the previous one but does not require any modification of the logic in the gated-clock subsystem. Moreover, the test generation process is does not require substantially higher computational effort compared to the original implementation.

These transformations allow the designer to safely apply power-saving techniques based on clock-gating

Benchmark	Cells no.	Obs. and Contr.		Observability		
		Tests no.	CPU (s)	Tests no.	CPU (s)	Coverage
bbtas	28	28	0.6	49	1.2	98.1
bbara	63	109	2.9	187	10.8	99.0
bbsse	129	187	70	241	815	98.7
lion9	41	102	1.9	133	2.3	98.3
keyb	127	302	39	315	403	100
styr	491	818	1228	882	34128	98.0
s420	90	288	50	310	1213	98.7
test	11	111	3.4	197	8.4	99.2
scf	491	1020	514	1125	4008	98.2

Table 2: **Comparison between the testability characteristics of the gated clock version with increased controllability and observability vs. the version with increased observability only. The first column shows the circuit size in number of library cells.**

without renouncing to the high testability requirements typical of industrial designs.

References

[1] A. Chandrakasan *Low-Power digital CMOS design.* Kluwer, 1995.

[2] T. Meng, B. Gordon, et al., "Portable Video-on-Demand in wireless Communication," *Proceedings of the IEEE*, no. 4, pp. 659–680, April 1995.

[3] S. Devadas and S. Malik, "A survey of optimization techniques targeting Low Power VLSI circuits", in *DAC, Proceedings of the Design Automation Conference*, pp. 242-247, June 1995.

[4] D. Singh, J.M. Rabaey, et al., "Power conscious CAD Tools and methodologies: a perspective," *Proceedings of the IEEE*, no. 4, pp. 570–594, April 1995.

[5] M. Alidina, J. Monteiro, et al., "Precomputation-based sequential logic optimization for low power," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 426–436, Jan. 1995.

[6] V. Tiwari, S. Malik and P. Ashar, "Guarded evaluation: pushing power management to logic synthesis/design," *International Symposium on Low Power Design*, pp. 221–226, April 1995.

[7] L. Benini, P. Siegel and G. De Micheli, "Automatic synthesis of gated clocks for power reduction in sequential circuits" *IEEE Design and Test of Computers*, pp. 32–40, Dic. 1994.

[8] L. Benini and G. De Micheli, "Transformation and synthesis of FSMs for low power gated clock implementation " *International Symposium on Low Power Design*, pp. 21–26, April 1995.

[9] N. Yeung, et al., "The design of a 55SPECin92 RISC processor under 2W," in *IEEE International Solid-State Circuits Conference*, pp. 206–207, Feb. 1994.

[10] B. Suessmith and G. Paap III, "PowerPC 603 microprocessor power management," *Communications of the ACM*, no. 6, pp. 43–46, June 1994.

[11] J. Schutz, "A 3.3V 0.6μm BiCMOS superscalar microprocessor," in *IEEE International Solid-State Circuits Conference*, pp. 202–203, Feb. 1994.

[12] E. Sentovich, et al., "Sequential circuit design using synthesis and optimization," in *ICCD, Proceedings of the International Conference on Computer Design*, pp. 328–333, Oct. 1992.

[13] G. De Micheli. *Synthesis and optimization of digital circuits.* McGraw-Hill, 1994.

[14] S. Devadas, A. Ghosh and K. Keuzer, *Logic Synthesis.* McGraw-Hill, 1994.

[15] S. Devadas and K. Keutzer, "A unified approach to the synthesis of fully testable sequential machines," *IEEE Transactions on Computer-Aided Design*, vol. 10, no. 4, pp. 39–51, Jan. 1991.

[16] H. Cho, G. Hachtel and F. Somenzi, "Redundancy Identification/Removal and Test Generation for Sequential Circuits Using Implicit State Enumeration," *IEEE Transactions on Computer-Aided Design*, vol. 12, no. 7, pp. 935–45, July 1993.