A Unified Approach for the Synthesis of Self-Testable Finite State Machines

Bernhard Eschermann, Hans-Joachim Wunderlich
Institut für Rechnerentwurf und Fehlertoleranz, Universität Karlsruhe, Germany

Abstract – Conventionally self-test hardware is added after synthesis is completed. For highly sequential circuits like controllers this design method either leads to high hardware overheads or compromises fault coverage. In this paper we outline a unified approach for considering self-test hardware like pattern generators and signature registers during synthesis. Three novel target structures are presented, and a method for designing parallel self-testable circuits is discussed in more detail. For a collection of benchmark circuits we show that hardware overheads for self-testable circuits can be significantly reduced this way without sacrificing testability.

1 Introduction

Two categories of sequential circuits can be distinguished, namely data paths and control paths. Circuits of the first type consist of regular modules with a simple interconnection structure and relatively few feedbacks, whereas the interconnection structure of circuits of the second type is complex and irregular with many feedback lines, leading to a high sequential depth in spite of the relatively small number of storage elements. Testing and particularly implementing a built-in self-test (BIST) is more difficult and requires more effort for such highly sequential circuits. In spite of this, incorporating BIST circuitry into highly sequential portions of the chip is necessary, if a chip is to be made completely self-testable, e. g. to allow testing it with a "RUNBIST" instruction via its boundary scan interface [IEEE 90] after placing it on a board.

The *behavior* of a highly sequential circuit is commonly described by a finite state machine (FSM) model and its *structure* by an interconnection of combinational logic and storage elements (Fig. 1). FSM synthesis, the automatic generation of structural from behavioral descriptions, has been thoroughly investigated. For implementations with PLA´s or random combinational logic, state assignment and logic minimization are known to have a strong impact on the quality of the resulting designs. Recent research tries to consider testability during synthesis. State assignment and logic minimization can avoid redundancies [DMNS 90], controllability and observability can be increased by adding special state transitions to the FSM description [AgCh 90]. These techniques support external testing; in this paper we deal with the problem of synthesizing self-testable circuits.

Self-test hardware, i. e. pattern generators and signature registers for response analysis are conventionally added after the synthesis is finished. We show that for self-testable circuits a proper choice of the self-test strategy has a major impact on the quality of the resulting design if the self-test hardware is accounted for while synthesizing the circuit. We develop circuit structures and optimization procedures targeted towards self-testable circuits which decrease the area or speed penalties of such circuits. They can also increase the testability of dynamic faults and reduce the number of test control signals. Criteria are given to decide about the best self-test structure based on the major design goals – area, speed, design effort, test length and fault coverage.

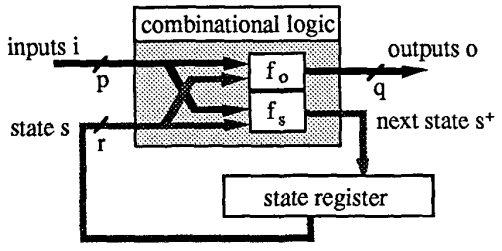

Fig. 1: Basic structure of sequential circuits.

The paper is organized as follows: Section 2 presents two conventional and three novel BIST structures for highly sequential circuits and discusses their relative merits. In Section 3 a general framework for the required synthesis for testability procedures is established. Optimization procedures are discussed in more detail for a new parallel self-testable circuit structure, which offers significant advantages in terms of area and testability. The results are validated with a collection of FSM synthesis benchmarks in Section 4.

2 BIST Structures for Finite State Machines

2.1 Conventional BIST Structures

If the state register in Fig. 1 is replaced by a single multifunctional self-test register, e. g. a BILBO [KöMZ 79], the direct feedback lines imply that the signatures of the test responses would have to be used as test patterns for the state variables. In [WaMc 87] the direct feedback path from storage elements to storage elements via the combinational logic is broken by doubling the number of flipflops and adding an additional self-test register solely responsible for compacting the test responses. The state register itself is reconfigured as a pure pattern generator in self-test mode (see Fig. 2a)[1]. Another possibility would be to incorporate the MISR (multiple input signature register) functionality into the state register and to provide a separate pattern generator (see Fig. 2b). These solutions are feasible, but they may result in significant hardware overheads. In the sequel we designate these structures as *DFF*, since in system mode the state registers are only used as D-flipflops.

[1] Pattern generation and response analysis for primary inputs and outputs are not shown, since they are identical for all self-test structures.

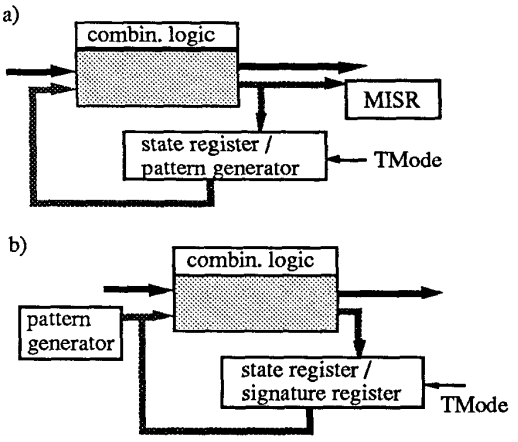

Fig. 2: Conventional BIST structures for FSM's (DFF).

2.2 *Motivation for Alternative Target Structures*

The state registers of Fig. 2 are not only D-flipflops but they have the additional functionality of a pattern generator or a signature register. The following simple example from [EsWu 90] shows how the ability of a linear feedback shift register (LFSR) to generate patterns can be utilized for the implementation of the system logic.

Example: The state diagram of the FSM to be implemented is shown in Fig. 3a. The three states of the FSM are already encoded. For test pattern generation the LFSR with the feedback polynomial $1+x+x^2$ is used. Its autonomous state transitions are shown in Fig. 3b. It is easily seen that the LFSR function covers a part of the system function. There is no need to implement these state transitions in the system logic if in the synthesized circuit structure it is possible to switch the state register between D-flipflop and LFSR mode. To be useful, the savings from not having to implement these state transitions of course have to be larger than the cost for the additional mode control signal.

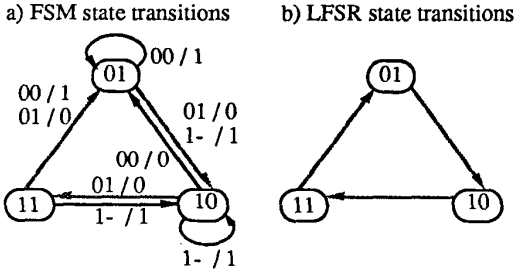

Fig. 3: Utilizing pattern generation capabilities of the state register.

In the following sections we present the general target structures for using the signature analysis or pattern generation capabilities of state registers during system mode.

2.3 *BIST Structure Utilizing Pattern Generator Functions*

Pattern generators for self-testable designs in autonomous mode cycle through a fixed sequence of states to stimulate the circuit. This property can also be used in system mode, if the encodings of the present and the next state are consecutive elements in this cycle. Whenever the next state code in Fig. 2a is produced by the pattern generation register, which has to be implemented for testing purposes anyway, it is not necessary to generate it in the next state logic. Replacing the next state entries with don't cares for all such transitions, increases the potential for logic optimization of the combinational logic. Fig. 4 illustrates a possible realization of this idea [EsWu 90]. An additional output signal "Mode" determines, whether the state machine flipflops behave like ordinary D-flipflops or function in pattern generation mode. In this mode the state register generates the next state on its own ("smart state register"), the next state signals asserted by the combinational logic can be set to arbitrary values. Since in this structure pattern generation is integrated into system mode, we refer to it as *PAT*. The additional MISR may be saved, if observing the state is possible in another way (cf. e. g. [Ghee 89]).

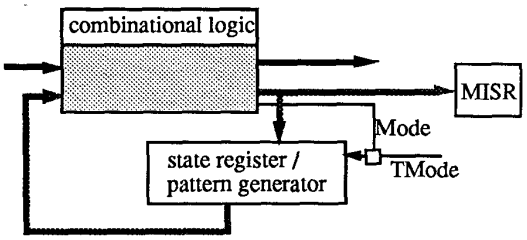

Fig. 4: BIST structure with integrated pattern generator (PAT).

2.4 *BIST Structures with Integrated Signature Register*

In a "parallel self-test" signature register outputs are used as test patterns [KiHT 88]. Empirical results for structures without direct feedbacks indicate that for certain examples this does not cause a significant loss of fault coverage; similar results were published for the circular self-test path approach [KrPi 89]. However, it cannot *guarantee* a high enough fault coverage and requires extensive fault simulation; in structures with direct feedbacks it might be completely impossible to set the next state lines to all the values needed to detect certain faults [ChGu 89]. The problem is caused by dividing the register functionality into a system mode and a self-test mode. In self-test mode additional XOR-gates are in the data path, whereas in system mode these gates are disabled by some form of mode control logic, for which the control signals are provided externally. Since the excitation function of the flipflops is changed, the state diagram in self-test mode is different from the state diagram in system mode. Only in special cases we have a modified state transition graph in self-test mode which is strongly connected, such that all system states stay reachable from all other system states.

Contrary to these solutions, the structure of Fig. 5 does not contain a control signal for switching between MISR and D-flipflop mode. Such a structure becomes possible, if the system functionality is implemented by using the MISR in its signature analysis mode as state register.

Let M(s) be the next state of a MISR in autonomous mode, m(s) the feedback function of the MISR, $f_s(i, s)$ the next state function of the system logic and $f_y(i, s)$ the excitation function of the state register. Because of the linearity of the operations involved, the necessary excitation variable y to produce a state transition from state s to state s^+ can be computed easily and is[2]

[2] The variables denote bit vectors, $\oplus$ denotes a bitwise XOR-operation on these vectors.

$$y = s^+ \oplus M(s) = f_s(i, s) \oplus M(s) = f_y(i, s)$$

compared with $y = f_s(i, s)$ for D-flipflops. This is similar to T-flipflops where we have $y = f_s(i, s) \oplus s = f_y(i, s)$. By implementing a pertinent next state function $f_y(i, s)$, arbitrary FSM's can be realized with MISR's as state registers, which makes it unnecessary to provide a special system mode.

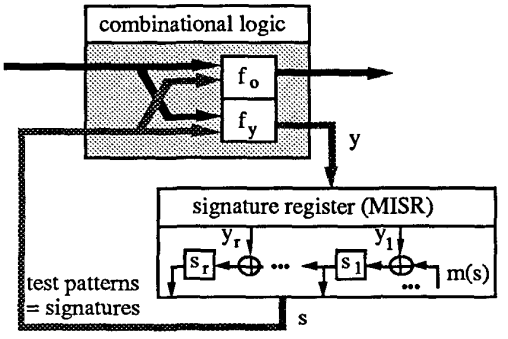

Fig. 5: Parallel BIST structure with MISR state register (PST).

The circuit structure for a parallel self-test without disjoint system and test modes (called *PST* in the sequel) in many cases has advantages with respect to area and testability. No flipflop duplication is required, the area of the self-test register is reduced by eliminating the D-flipflop mode. Besides signature analysis the only other mode needed is a scan mode to initialize the flipflops and to shift out the resulting signature; hence the number of control signals is decreased. The cause of the controllability problem mentioned earlier is also removed, as a self-test mode with modified state transitions is avoided. Since the functionality in self-test mode is identical to the system functionality, all states reachable in system mode stay reachable during self-test. As there is no reconfiguration of the flipflops in self-test mode, a test at the full clock frequency can be performed in order to detect dynamic faults relevant to system operation, e. g. delay faults, if only the test patterns for the primary inputs are supplied fast enough, for example with a random pattern generator (cf. the analysis in [EsWu 91]). By targeting the state assignment algorithm towards MISR state registers, which secure the observability of the memory elements, the combinational logic needed to implement the system function can be optimized in the same way a controller with D-flipflops can be optimized.

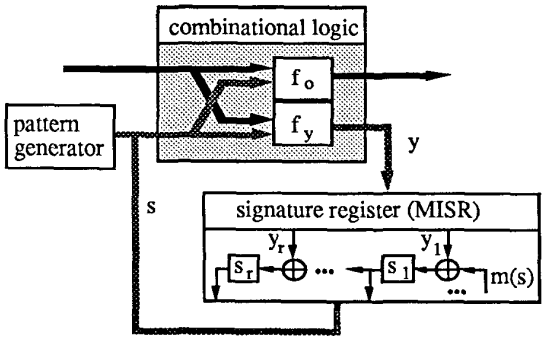

Fig. 6: BIST structure with integrated signature analysis (SIG).

Some of these advantages can also be obtained without using a parallel self-test by integrating the signature register into the structure of Fig. 2b as shown in Fig. 5. The resulting circuit structure is illustrated in Fig. 6 and will be called *SIG*.

2.5 *Comparison*

In Table 1 the main characteristics of the BIST structures are compared.

In conventional self-test structures (DFF) the area needed for storage elements is high, since a signature register has to be added only for testing purposes. The speed of the circuit is decreased by the additional control logic and XOR-gates in the data path. Two control signals are needed to operate the state register (scan path/initialization mode, pattern generation mode, system mode). It may not be possible to detect all the dynamic faults relevant to the system mode in the next state logic, since the next state variables are monitored in a separate register not used in system mode. The structure with integrated pattern generator (PAT) decreases the necessary amount of combinational logic, but apart from that is identical with the conventional structure.

By using the structure SIG and avoiding disjoint signature analysis and system modes, the control logic for the state register can be simplified; one control signal is enough now. The combinational logic can become simpler or more complicated depending on the FSM under consideration. Since the next state signals are captured in the same register, where they are needed in system mode, dynamic faults can be more easily detected.

structure	DFF	PAT	SIG	PST
area				
• combin. logic	o	++	+/–	+/–
• storage elements	–	–	o	+
speed	o	–	o	++
test length	+	+	+	o/–
test control effort	–	–	o	+
dyn. fault detection	–	–	o	+

Table 1: Comparison of different BIST structures.

By integrating the signature register in such a way, it is also possible to use the signatures as test patterns without running into problems with unreachable states. Thus the separate pattern generator may be saved, a parallel self-test becomes possible (structure PST). Now there is no more difference between system and test mode with respect to the production and capture of the next state signals. Therefore all dynamic faults occuring in system mode can be detected during self-test. The test length and the effort to produce pertinent test patterns for the primary inputs may, however, increase.

A detailed examination of the testability aspects (input stimulation, fault model, fault masking probability, fault coverage) of the structure PST may be found in [EsWu 91]. For typical examples an increase of 30 % in the number of weighted random test patterns was computed to obtain the same test confidence (probability of detecting *all* faults of a given fault set, in this case we wanted to achieve a single stuck-at test confidence of 99.9 %) as for a conventional self-test. For some sequential circuits several different weight distributions might be needed to obtain reasonable test lengths.

In summary there is no single self-test structure that is preferable in all cases, as the importance of the criteria listed in Table 1 depends on the application. If automatic synthesis procedures are available for all the self-test structures, it is possible to try alternative designs and then decide about the actual implementation of the circuit.

3 Synthesis and Optimization Procedures

The BIST structures presented indicate that self-testable circuits can be optimized by considering the self-test hardware (pattern generators and signature registers) during synthesis. To make the best use of this minimization potential in practice, it is however necessary to find optimization procedures targeted towards the BIST structures presented. Conventional synthesis procedures cannot take the functionality of self-test registers into account. Therefore they can only be used for the self-test structures in Fig. 2, where the system mode is completely independent of the self-test hardware.

3.1 Synthesis Framework

The main steps necessary for synthesizing a self-testable circuit from an FSM description are illustrated in Fig. 7. After choosing a BIST structure, the symbolic states of the FSM description have to be assigned binary code words. Afterwards the excitation functions for forcing the memory elements into the correct next states have to be derived. At that point a truth table for a multi-output boolean function is obtained, which can then be minimized using standard programs.

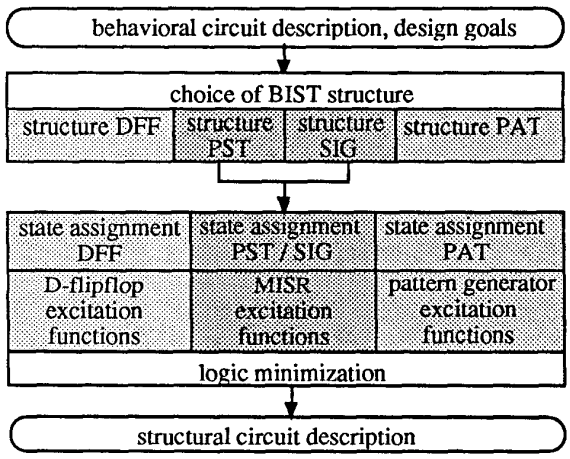

Fig. 7: Synthesis process for BIST FSM's.

In section 3.2 we first show how the excitation functions for the state flipflops are derived for the different BIST structures. Since the excitation functions strongly depend on the BIST structure chosen, the cost function representing the complexity of the resulting combinational logic should reflect this influence. Procedures to encode the states in such a way that this cost function is minimized are treated in section 3.3.

3.2 Excitation Functions

The symbolic output and next state functions of an FSM are

$$o = F_o(i, S) \text{ and } S^+ = F_s(i, S),$$

where $i \in \{0,1\}^p$ is the vector of input variables, $o \in \{0,1\}^q$ is the vector of output variables and S, $S^+ \in \mathcal{S}$ are symbolic states. The state assignment ψ: $\mathcal{S} \rightarrow \{0,1\}^r$, $\psi(S) = s = (s_1, \ldots s_r)$ is an injective mapping from the state set $\mathcal{S}$ into the set of state codes $\{0,1\}^r$, where $r \geq r_0 = \lceil \log_2 |\mathcal{S}| \rceil$. The resulting binary output and next state functions are

$$o = f_o(i, s) \text{ and } s^+ = f_s(i, s),$$

with $s = \psi(S)$ and $s^+ = \psi(S^+)$. Let $\tau(s, s^+)$ be a function with which the excitation variables

$$y = \tau(s, s^+) = f_y(i, s)$$

for a certain type of state register can be obtained. The complexity $C(\tau, \psi)$ of the necessary combinational logic to implement the FSM depends on the output functions f_o and the excitation functions f_y, which in turn depend on the type of state register τ chosen and the state assignment ψ:

(DFF) For a state register with D-flipflops, we simply have

$$\tau(s, s^+) = s^+, \quad y = (y_1 \ldots y_r) = (s_1^+ \ldots s_r^+).$$

(PST / SIG) The situation is a bit more complicated for a MISR state register (cf. Fig. 5), where

$$\tau(s, s^+) = s^+ \oplus M(s),$$

$$y = (y_1 \ldots y_r) = (s_1^+ \oplus m(s), s_2^+ \oplus s_1 \ldots s_r^+ \oplus s_{r-1}).$$

(PAT) For the case of "smart" state registers (cf. Fig. 4) the excitation vector contains an additional element "Mode", which controls, whether the register loads the excitation variables $(y_1 \ldots y_r)$ or just goes to the next state M(s) in autonomous mode

$$\tau(s, s^+) = \begin{cases} \text{don't care for Mode} = 0 \\ s^+ \text{ for Mode} = 1 \end{cases},$$

$$y = (y_1 \ldots y_r, \text{Mode}) = \begin{cases} (-\ldots -, 0) \text{ for } s^+ = M(s) \\ (s_1^+ \ldots s_r^+, 1) \text{ else} \end{cases}.$$

Once the BIST structure is chosen, τ is fixed and the complexity of the combinational logic for a given FSM $C_\tau(\psi)$ mainly depends on the state assignment ψ.

3.3 State Assignment

The task of the state assignment procedure is to find an injective mapping ψ with minimal cost $C_\tau(\psi)$. Since the cost to increase the width of a self-test register is quite high, it is generally preferable to use the minimal number of state variables r_0. One possibility to obtain an optimal assignment would be to enumerate all possible functions ψ, to minimize the resulting output and excitation functions and to keep track of the assignment with minimal $C_\tau(\psi)$. Unfortunately this is only feasible for small FSM's [McUn 59, WeSm 67]. State assignment is actually an NP-hard problem [WoKA 88], therefore heuristics have to be used. In the sequel we will only consider the state assignment problem PST / SIG. The DFF structures can be synthesized using state assignment algorithms for D-flipflops (e.g. [DMNS 88, ViSa 90]), a state assignment algorithm for the problem PAT has been described in [EsWu 90].

3.3.1 Necessity of a special state assignment procedure. If a conventional state assignment procedure is used, the combinational logic is optimized such that $y = s^+$ is easily minimizable. It is easy to validate that the same assignment procedures are not effective for minimizing the function $y = s^+ \oplus M(s)$. With a state assignment targeted to make $y = s^+ \oplus M(s)$ easily minimizable, the combinational logic of the PST/SIG solutions can be implemented much more efficiently.

The sequence of code bits influences the combinational logic for MISR state registers because of the direct dependence of excitation variables on the contents of other flipflops in the MISR. For r state variables and n states the number of non-equivalent state assignments is therefore

$$SA(r, n) = \frac{2^r!}{(2^r - n)!}$$

and exceeds the number relevant to D-Flipflops by a factor of r!. Conventional state assignment algorithms cannot cope with the complex dependences in MISR state registers. Algorithms for T- and JK-flipflops [WeDo 69, TuBr 74]

have been published, but do not help since in these cases the value of the i-th excitation variable y_i only depends on the contents s_i of the i-th flipflop. Consequently it is necessary to develop a new state assignment algorithm for this application.

3.3.2 A PST / SIG state assignment procedure. The goal is to devise a state assignment strategy, which follows the structural dependences in a MISR (cf. Fig. 5). This can be achieved by encoding the states state variable by state variable. In the resulting divide-and-conquer algorithm the set of states is recursively partitioned into two sets, one encoded with a code bit 0, the other with a 1. When only one state is left in a partition, its encoding is different from the encoding of all other states. The partitioning and assignment is done such that a cost function reflecting the complexity of realizing the next state and output logic is minimized. To obtain such a cost function, however, is more difficult than in the D-flipflop case, because the values of the excitation variables depend on other state variables. The idea used is that once an encoding for one state variable s_{i-1} is fixed, the excitation variable y_i can be derived from s_{i-1} and the code of the next state variable s_i^+,

$$y_i = s_i^+ \oplus s_{i-1} \qquad i = 2, \ldots r$$

(cf. section 3.2). Alternatively, s_i^+ can be chosen such that y_i becomes as simple to implement as possible, so that at any point in the state assignment process the cost of the next assignment can be estimated. The process of assigning code bits state variable by state variable and computing the excitation variables from the already known state variable values is illustrated in Fig. 8.

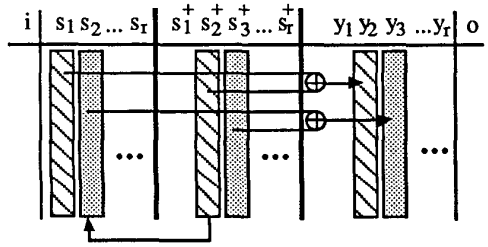

Fig. 8: FSM transition/excitation table with dependences.

The cost function is computed as follows: First the output function $o = f_o(i, S)$ is symbolically minimized [DeMi 86]. The resulting number of symbolic implicants is a lower bound for the number of product terms needed in a PLA implementation. By fixing a coding column, the number of implicants required to represent f_o and the partial excitation function f_y^i up to the current column i may increase because of two effects:

- Groups of symbolic present states can no longer be encoded in a subspace of $\{0,1\}^r$ not containing other symbolic states [DeMi 86] and have to be split (input incompatibility).
- The resulting excitation variable in the current column, which could not be considered during symbolic minimization, is different for state transitions summarized in the same symbolic implicant (output incompatibility).

The cost function reflects the increase in the number of necessary implicants. Several partitions of the state set into 0- and 1-encoded states with small cost values are generated for each coding column and are explored using a branch-and-bound algorithm. The tradeoff between runtime and the quality of the resulting solution can be controlled by restricting the number of partitions considered for each column.

Only the determination of the first state variable s_1 remains problematic, as the implementation effort for y_1 cannot be estimated before all the other coding columns are known. However, it is possible to estimate its effect on the complexity of the output function $f_o(i,s)$ and to use this information to choose s_1. After state assignment, the MISR feedback function m(s) can be chosen in such a way that $y_1 = s_1^+ \oplus m(s)$ is easily realizable. Even if for testability reasons a primitive feedback polynomial is required, a large number of choices for m(s) remains. The synthesis process for PST / SIG structures is summarized in Fig. 9. When k is the number of assignment partitions explored for each state variable, the worst case size of the search space is $O(k^r)$, but typically only a small percentage of all branches have to be enumerated explicitly.

```
procedure MISR_state_assignment
  read FSM description;
  for a set of feasible encodings of variable s1:
      estimate the cost for implementing fo(i, s);
  choose the encoding s1opt with least cost;
  for all state variables si, i = 2, ... r:
      for a set of feasible encodings of variable si:
          compute yi = si-1 ⊕ si+;
          estimate the cost for implementing fo(i, s) and y2, ... yi;
      choose the encoding siopt with least cost;
  for all primitive MISR feedback functions m(s):
      compute y1 = m(s) ⊕ s1+;
      estimate the cost for implementing fo(i, s) and fy(i, s);
  choose the function mopt(s) with least cost;
  minimize fo(i, s) and fy(i, s) with logic synthesis program;
  return the optimized FSM implementation;
  end;
```

Fig. 9: Synthesis process for controllers with MISR state registers.

4 Results

The algorithm was programmed in C and evaluated with the examples from the MCNC benchmark set [MCNC 88]. The resulting numbers of product terms for the largest benchmarks are summarized in Table 2. The branch-and-bound algorithm was parameterized such that the run time for state assignment was in the range of minutes on a SUN 4/60. As no state assignment algorithm for signature registers was published until now and it is not feasible to compute an optimal solution for these circuits, the results are compared with the best of 50 randomly selected encodings. It can be seen that the heuristic algorithm is preferable to the costly trial-and-error method in all cases.

example	average of ... ... 50 random encodings	best of ...	heuristic solution
dk16	91.7	87	76
dk512	25.5	23	19
donfile	73.5	65	42
ex1	73.8	69	64
ex4	20.6	18	18
kirkman	122.1	94	67
mark1	26.0	25	23
modulo12	17.4	15	13
planet	103.9	102	94
sand	116.3	111	107
scf	168.0	156	138
styr	143.5	132	128
tbk	261.9	224	159

Table 2: Number of product terms for PST/SIG state assignment.

Table 3 compares synthesis results[3] after two-level and multi-level logic minimization for the three approaches presented in this paper. The PST / SIG structures have advantages with respect to testing speed, fault coverage and test control compared with conventional DFF solutions [EsWu 91]; the table shows that these advantages are obtained without having to pay for them with a significant increase of hardware over the DFF solution. Some examples even lead to a lower combinational logic complexity when implemented with a MISR state register, while others require more area for the next state logic than the DFF solution. It should be noted that Table 3 is biased in favor of the conventional DFF solution, because the self-test registers required for pattern generation and to secure the observability of state variables are not taken into account. The PAT structure can decrease the amount of combinational logic by 10-20 % compared with the DFF solution; these results resemble those given in [AmEB 88] for FSM's with loadable counters used as state registers.

example	number of product terms			number of literals		
	PST/SIG	DFF	PAT	PST/SIG	DFF	PAT
dk16	76	59	57	289	270	241
dk512	19	18	17	67	70	48
donfile	42	29	28	121	160	74
ex1	64	48	44	288	280	253
ex4	18	19	16	65	77	70
kirkman	67	64	54	153	176	146
mark1	23	20	17	119	108	94
modulo12	13	13	9	39	35	29
planet	94	91	83	545	578	569
sand	107	97	97	566	570	547
scf	138	146	136	714	822	773
styr	128	94	93	629	594	512
tbk	159	149	59	421	547	496

Table 3: Comparison of PST/SIG, DFF and PAT results.

5 Conclusions

Conventional self-test approaches for highly sequential circuits either require large hardware overheads or have to compromise testability. We presented a unified framework to synthesize BIST structures overcoming that problem. Several synthesis procedures targeted towards self-testable FSM's were implemented. Depending on the major design goal, area, speed, test length, test control effort or the detectability of dynamic faults can be optimized compared to conventional solutions. The approach is well-suited to sequential circuits typically described by an FSM model (e. g. controllers); in its current form it is, however, not applicable to the synthesis of sequential circuits with a large number of states like data paths.

Acknowledgements

The authors would like to thank Mr. A. Rothacker for his help in implementing the algorithms. We would also like to acknowledge Dr. R. Kumar for reading the manuscript.

References

AgCh 90 V. D. Agrawal, K.-T. Cheng: An Architecture for Synthesis of Finite State Machines; Proc. 1st European Design Automation Conference, pp. 612-616, 1990.

AmEB 88 R. Amann, B. Eschermann, U. G. Baitinger: PLA-Based Finite State Machines Using Counters as State Memories; Proc. Int. Conf. on Computer Design, pp. 267-270, 1988.

ChGu 89 C. Chuang, A. Gupta: The Analysis of Parallel BIST by the Combined Markov Chain (CMC) Model; Proc. Int. Test Conference, pp. 337-343, 1989.

DeMi 86 G. DeMicheli: Symbolic Design of Combinational and Sequential Logic Circuits Implemented by Two-Level Logic Macros; IEEE Trans. on Computer-Aided Design, vol. CAD-5, pp. 597-616, 1986.

DMNS 88 S. Devadas, H.-K. Ma, A. R. Newton, A. Sangiovanni-V.: MUSTANG: State Assignment of Finite State Machines Targeting Multilevel Logic Implementations; IEEE Trans. on Computer-Aided Design, vol. CAD-7, pp. 1290-1300, 1988.

DMNS 90 S. Devadas, H.-K. T. Ma, A. R. Newton, A. Sangiovanni-Vincentelli: Irredundant Sequential Machines Via Optimal Logic Synthesis; IEEE Trans. on Computer-Aided Design, vol. CAD-9, pp. 8-18, 1990.

EsWu 90 B. Eschermann, H.-J. Wunderlich: Optimized Synthesis of Self-Testable Finite State Machines; Proc. 20th Int. Symp. Fault-Tolerant Computing, pp. 390-397, 1990.

EsWu 91 B. Eschermann, H.-J. Wunderlich: Parallel Self-Test and the Synthesis of Control Units; Proc. 2nd European Test Conf., Munich, April 1991.

Ghee 89 T. Gheewala: CrossCheck: A Cell Based VLSI Testability Solution; Proc. 26th Design Automation Conference, pp. 706-709, 1989.

IEEE 90 IEEE Std. P1149.1: Standard Test Access Port and Boundary-Scan Architecture, 1990.

KiHT 88 K. Kim, D. Ha, J. Tront: On Using Signature Registers as Pseudorandom Pattern Generators in Built-in Self Testing; IEEE Trans. on CAD, vol. 8, pp. 919-928, 1988.

KöMZ 79 B. Könemann, J. Mucha, G. Zwiehoff: Built-In Logic Block Observation Techniques; Proc. Int. Test Conference, pp. 37-41, 1979

KrPi 89 A. Krasniewski, S. Pilarski: Circular Self-Test Path: A Low-Cost BIST Technique for VLSI Circuits; IEEE Trans. on CAD, vol. 8, no. 1, pp. 46-55, 1989.

MCNC 88 R. Lisanke: Logic Synthesis and Optimization Benchmarks, Version 2.0; Microelectronics Center of North Carolina, 1988.

McUn 59 E. McCluskey, S. Unger: A note on the Number of Internal Variable Assignments for Sequential Switching Circuits; IRE Trans. on Electronic Computers, vol. EC-8, pp. 439-440, 1959.

TuBr 74 G. Tumbush, J. Brandeberry: A State Assignment Technique for Sequential Machines Using J-K Flip-Flops; IEEE Trans. on Comp., vol. 23, pp. 85-86, 1974.

ViSa 90 T. Villa, A. Sangiovanni-V.: NOVA: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementation; IEEE Trans. on Computer-Aided Design, vol. CAD-9, pp. 905-924, 1990.

WaMc 87 L. Wang, E. McCluskey: Built-in Self-Test for Sequential Machines; Proc. Int. Test Conference, pp. 334-341, 1987.

WeDo 69 P. Weiner, T. Dolotta: Mixed Memory Realizations of Sequential Machines; IEEE Trans. on Comp., vol. 18, pp. 272-277, 1969.

WeSm 67 P. Weiner, E. Smith: On the Number of Distinct State Assignments for Synchronous Sequential Machines; IEEE Trans. on Electronic Computers, vol. EC-16, pp. 220-221, 1967.

WoKA 88 W. Wolf, K. Keutzer, J. Akella: A Kernel-Finding State Assignment Algorithm for Multi-Level Logic; Proc. 25th Design Automation Conference, pp. 433-438, 1988.

[3] The DFF solutions were obtained using the programs *nova* (-ihybrid) [ViSa 90] for 2-level logic and *mustang* (best results of fanin and fanout algorithm) [DMNS 88] for multi-level logic.