ACM/IEEE International Conference on CAD 94 (ICCAD-94), San Jose, Ca., November 1994

An Efficient Procedure for the Synthesis of Fast Self-Testable Controller Structures

Sybille Hellebrand, Hans-Joachim Wunderlich
Institute of Computer Structures
University of Siegen
Germany

Abstract

The BIST implementation of a conventionally synthesized controller in most cases requires the integration of an additional register only for test purposes. This leads to some serious drawbacks concerning the fault coverage, the system speed and the area overhead. A synthesis technique is presented which uses the additional test register also to implement the system function by supporting self-testable pipeline-like controller structures. It will be shown, that if the need of two different registers in the final structure is already taken into account during synthesis, then the overall number of flipflops can be reduced, and the fault coverage and system speed can be enhanced. The presented algorithm constructs realizations of a given finite state machine specification which can be trivially implemented by a self-testable structure. The efficiency of the procedure is ensured by a very precise characterization of the space of suitable realizations, which avoids the computational overhead of previously published algorithms.

1 Introduction

The increasing demand for highly reliable micro-electronic systems in various safety-critical applications prompts for extremely high quality standards, which have to be guaranteed by refined testing techniques. Built-in self-test (BIST) is of particular importance, because it allows an efficient production testing and the capabilities for pattern generation and test response evaluation on chip can also be used for periodic maintenance tests [1].

Often the BIST is implemented by so-called multi-functional test registers like the well-known BILBO which are able to work as a system register, to generate test patterns and to compress the test responses by signature analysis. Such registers have been developed for random, deterministic, pseudo-exhaustive, and weighted random pattern testing [4, 9, 17, 20, 26]. However, in general it is not possible to use one multi-functional register for test pattern generation and test response evaluation concurrently, since this way the required properties of the test patterns can only be ensured in some special cases [12, 14, 19]. In most cases two different registers are necessary to generate the patterns and to compress the test responses. Conventional synthesis procedures for controllers and even most of the advanced synthesis techniques for sequentially irredundant and easily testable controllers do not take into account this fact [2, 5, 6, 7, 10, 11, 13, 22, 23, 24]. They usually provide a circuit structure as shown in figure 1, which has to be complemented by an extra test register for BIST (see figure 2).

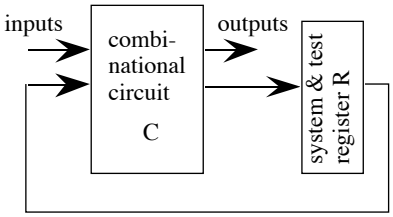

Figure 1: Conventionally synthesized controller structure.

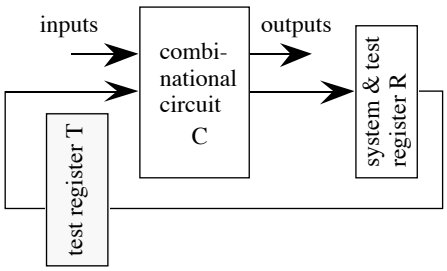

Figure 2: Required modifications for BIST.

During test mode the register T is used as pattern generator and the multi-functional register R is configured as signature analyzer, during system mode T must be transparent. This configuration with an extra register only for test purposes has some serious drawbacks:

1) The number of flipflops must be doubled.

This work was supported by EP 7107 ARCHIMEDES.

2) In system mode the test register T must be transparent or bypassed. This prolongs the critical path and may slow down the system speed of the controller.
3) There are faults on the feedback lines from R to the inputs of C which are not detected, as these lines are not completely exercised during self-test. This holds, even if the connections between R and T are tested in an additional step.

The above mentioned drawbacks can be avoided if the required additional test register T is also used to implement the system function. This results in a pipeline-like structure as shown in figure 3 and described in detail in [18].

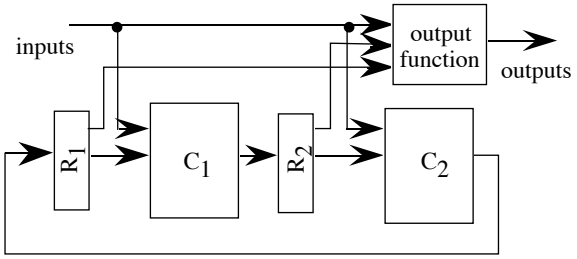

Figure 3: Controller target structure.

A state of the controller is represented by the contents of both registers R_1 and R_2, and the state transition function is implemented by two independent combinational circuits C_1 and C_2. Clearly none of the registers needs to be transparent during system mode and there is no additional delay imposed this way. The self-test can be performed in two sessions by alternatively using one of the registers for pattern generation and the other for signature analysis. Moreover, as there is no transparency mode or bypassing a complete fault coverage is possible. In general the registers R and T of figure 1b are wider than the registers R_1 and R_2 of the proposed target structure, hence the structure of figure 3 needs less flipflops then the self-testable structure of figure 1b. Furthermore, also the combinational circuits C_1 and C_2 are smaller than the original circuit. As a consequence the critical path in C_1, C_2 and the output function is shorter than in the circuit C of figure 1. This allows higher clock rates and thus leads to a higher system performance.

It is important to note that this structure is different from structures provided by decomposition techniques where the resulting submachines contain internal feedback loops [16, 3, 15]. In [18] a computationally expensive search procedure was used for a feasibility study of the approach. In this paper a computationally efficient synthesis procedure for target structures as shown in figure 3 is presented, which makes the approach generally applicable. In contrast to known approaches trying to reduce dependencies between state variables by appropriate state coding the presented procedure relies on algebraic structure theory to address the problem already at the finite state machine level [25, 8]. The theoretical basis of the work is a very precise characterization of the search space.

The organization of the paper is as follows: In section 2 the main results of [18] are briefly summarized. Subsequently in section 3 the main theorem for a precise characterization of the search space is proven and an efficient search procedure is developed. Section 4 provides experimental results achieved for a collection of finite state machine benchmarks [McEl 93]. Conclusions are given in section 5.

2 Partition Pairs and Self-Testable Realizations

The problem of synthesizing controller structures compatible with BIST can be reduced to the problem of constructing suitable realizations of the original finite state machine specifications.

Throughout this work it is assumed that a controller is fully specified as a mealy-type finite state machine $M = (S, I, O, \delta, \lambda)$ with a finite non-empty set of states S, a finite non-empty set of inputs I, a finite non-empty set of outputs O, a next state function $\delta: S \times I \rightarrow S$ and an output function $\lambda: S \times I \rightarrow O$.

To guarantee that a finite state machine can be implemented by a self-testable structure it is necessary to require some additional properties.

Definition 1: Let $M = (S, I, O, \delta, \lambda)$ be a finite state machine. M is called a *finite state machine supporting a self-testable structure*, if and only if there are sets S_1 and S_2 and functions $\delta_1: S_1 \times I \rightarrow S_2$, $\delta_2: S_2 \times I \rightarrow S_1$, such that $S = S_1 \times S_2$ and $\delta((s_1, s_2), i) = (\delta_2(s_2, i), \delta_1(s_1, i))$ holds for all $s = (s_1, s_2) \in S$ and $i \in I$.

Obviously the straightforward implementation of such a finite state machine with logic blocks for δ_1, δ_2 and λ and registers for S_1 and S_2 provides a self-testable structure as shown in figure 3.

Self-testable controllers can therefore be synthesized from finite state machine specifications in two steps. First a finite state machine which realizes the specification and which supports a self-testable structure is constructed and then state coding and logic minimization algorithms are applied to this realization. A precise definition of the term realization can be found in [16, 18].

In [18] it has been shown that self-testable realizations for a finite state machine $M = (S, I, O, \delta, \lambda)$ correspond to specific pairs of equivalence relations on the set of states S. In the following equivalence relations on S will always be considered as subsets $\vartheta \subset S \times S$. This way the set theoretic operators „$\cap$“ (intersection) and „$\cup$“ (union) are defined for equivalence relations and there is a partial ordering on the set of equivalence relations given by „$\subset$“ (sub-

set). The intersection of two equivalence relations is again an equivalence relation, but the union need not be transitive. Therefore an operator „+“ is defined for equivalence relations by $\pi_1 + \pi_2 := (\pi_1 \cup \pi_2)^t$, where π^t denotes the transitive closure of a relation π. For an equivalence relation $\pi \subset S \times S$ and an element $s \in S$ the corresponding equivalence class is denoted by $[s]_\pi$. The set S/π of equivalence classes completely specifies π, and for convenience we describe π mostly by S/π and not by enumerating all the pairs.

Definition 2: Let $M = (S, I, O, \delta, \lambda)$ be a finite state machine, and let $\pi, \sigma \subset S \times S$ be equivalence relations on S. (π, σ) is called a *partition pair* for M, if and only if $\big((s, t) \in \pi \Rightarrow \forall\, i \in I: (\delta(s, i), \delta(t, i)) \in \sigma\big)$ holds. If (σ, π) is a partition pair, too, then (π, σ) is called a *symmetric partition pair*.

For the construction of self-testable realizations symmetric partition pairs are of special interest. The following theorem has been shown in [18]:

Theorem 1: Let $M = (S, I, O, \delta, \lambda)$ be a finite state machine. Let ε denote the equivalence of states and let (π, σ) be a symmetric partition pair for M satisfying $\pi \cap \sigma \subset \varepsilon$. Let $M^* = (S^*, I^*, O^*, \delta^*, \lambda^*)$ be defined by

(i) $S^* := S/\pi \times S/\sigma$, $I^* := I$, $O^* := O$,

(ii) $\delta^*((s_1, s_2), i)) := (\delta_2(s_2, i), \delta_1(s_1, i))$ with
$\delta_1(([s]_\pi, i)) := [\delta(s, i)]_\sigma$ and
$\delta_2([s]_\sigma, i)) := [\delta(s, i)]_\pi$, and

(iii) $\lambda^*((s_1, s_2), i) :=$
$$\begin{cases} \lambda(s, i) & \text{if } s_1 \cap s_2 \neq \emptyset \text{ and } s \in s_1 \cap s_2 \\ o^* & \text{else} \end{cases},$$
where $o^* \in O$ is an arbitrary output value.

Then M^* is a finite state machine supporting a self-testable structure which realizes M.

Theorem 1 is illustrated by the following example.

Example 1: Figure 4 shows the next state table of a small finite state machine. By $S/\pi = \{\{1, 2\}, \{3, 4\}\}$ and $S/\sigma = \{\{1, 4\}, \{2, 3\}\}$ a symmetric partition pair (π, σ) with $\pi \cap \sigma \subset \varepsilon$ is defined.

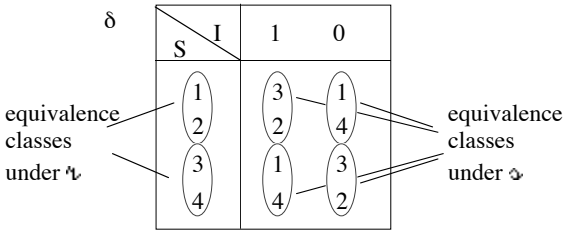

Figure 4: Effect of a partition pair (π, σ) on the next state table of a finite state machine.

The resulting mappings δ_1: $S/\pi \times I \rightarrow S/\sigma$ and δ_2: $S/\sigma \times I \rightarrow S/\pi$, which provide the state transition function δ^*, are shown in figure 5.

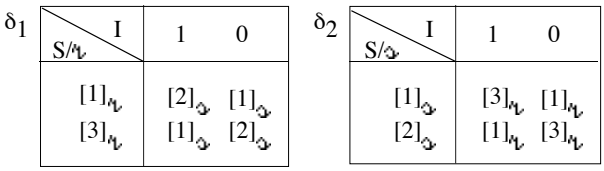

Figure 5: Tables for δ_1: $S/\pi \times I \rightarrow S/\sigma$ and δ_2: $S/\sigma \times I \rightarrow S/\pi$.

If $[1]_\pi$ and $[1]_\sigma$ are both encoded by 1 and $[3]_\pi$ and $[2]_\sigma$ are encoded by 0, then the constructed finite state machine $M^* = (S/\pi \times S/\sigma, I, O, \delta^*, \lambda^*)$ can be implemented by the structure shown in figure 6. ❑

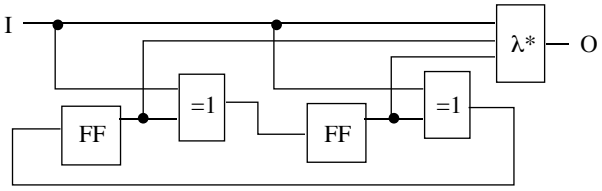

Figure 6: Structure of M*.

Theorem 1 has two consequences for the synthesis of self-testable controllers. Firstly, there is always a trivial self-testable realization for a given finite state machine $M = (S, I, O, \delta, \lambda)$, since the identity relation $id \subset S \times S$ provides a symmetric partition pair (id, id) with $id \cap id \subset \varepsilon$. The resulting finite state machine $M^* = (S/id \times S/id, I, O, \delta^*, \lambda^*)$ corresponds to simply „doubling“ the original machine. Secondly, the problem of finding an optimal self-testable realization with small registers of about equal size can be formulated as follows:

Problem OSTR (**O**ptimal **S**elf-**T**estable **R**ealization): Let $M = (S, I, O, \delta, \lambda)$ be a finite state machine. Find a symmetric partition pair (π, σ) with $\pi \cap \sigma \subset \varepsilon$, such that

(i) $\lceil \log_2 |S/\pi| \rceil + \lceil \log_2 |S/\sigma| \rceil$ is minimal, and

(ii) $\left| \frac{|S/\pi|}{|S/\sigma|} - 1 \right|$ is minimal

for all pairs satisfying (i).

In the next section an algorithm for OSTR will be developed, which is based on an efficient enumeration procedure for symmetric partition pairs.

3 An Efficient Algorithm for OSTR

In the previous section it has been shown that the problem of synthesizing self-testable controllers can be reduced to finding suitable symmetric partition pairs. The algorithm to be developed in this paragraph rigorously exploits the symmetry requirement in order to restrict the search to a small number of candidate pairs. It is based on the notion of „mm-pairs“ which will be introduced in the sequel.

Definition 3: Let M = (S, I, O, δ, λ) be a finite state machine and let $\mathfrak{r}, \mathfrak{s} \subset S \times S$ be equivalence relations. Then $m(\mathfrak{r})$ denotes the $\subset$-minimal equivalence relation, such that $(\mathfrak{r}, m(\mathfrak{r}))$ is a partition pair, and $M(\mathfrak{s})$ denotes the $\subset$-maximal equivalence relation, such that $(M(\mathfrak{s}), \mathfrak{s})$ is a partition pair. $(\mathfrak{r}, \mathfrak{s})$ is called an *Mm-pair*, if both $M(\mathfrak{s}) = \mathfrak{r}$ and $m(\mathfrak{r}) = \mathfrak{s}$ hold. If $m(\mathfrak{s}) = \mathfrak{r}$ and $m(\mathfrak{r}) = \mathfrak{s}$ are both true, then $(\mathfrak{r}, \mathfrak{s})$ is called an *mm-pair*.

The Mm-pairs for a finite state machine can be regarded as the skeleton for the set of all partition pairs [16], and the basic procedure for OSTR described in [18] used an enumeration procedure for the set of Mm-pairs. Since Mm-pairs need not be symmetric, this resulted in an unnecessary computational overhead. In the following it will be shown that the set of symmetric partition pairs can be directly characterized by mm-pairs. By definition of the m-operator an mm-pair is a symmetric partition pair. To show that it is sufficient to concentrate on mm-pairs some more theoretical effort is required. The first result derived from the symmetry property concerns the iterative construction of an mm-pair from an initial relation $\mathfrak{p}$. With $m^i(\mathfrak{p})$ denoting the relation $m(m^{i-1}(\mathfrak{p}))$, where $m^0(\mathfrak{p}) := \mathfrak{p}$, the following lemma can be shown.

Lemma 1: Let M = (S, I, O, δ, λ) be a finite state machine, and let $\mathfrak{p} \subset S \times S$ be an equivalence relation. Then the relations

$$m^{even}(\mathfrak{p}) := \sum_{i=0}^{\infty} m^{2i}(\mathfrak{p}) \text{ and } m^{odd}(\mathfrak{p}) := \sum_{i=0}^{\infty} m^{2i+1}(\mathfrak{p})$$

constitute an mm-pair $\left(m^{even}(\mathfrak{p}), m^{odd}(\mathfrak{p})\right)$. Furthermore, if $(\mathfrak{r}, \mathfrak{s})$ is any symmetric partition pair with $\mathfrak{p} \subset \mathfrak{r}$, then $m^{even}(\mathfrak{p}) \subset \mathfrak{r}$ and $m^{odd}(\mathfrak{p}) \subset \mathfrak{s}$.

Some useful properties of the operators m^{even} and m^{odd} can inductively be derived from the corresponding properties of the m-operator proven in [16].

Observation 1: Let M = (S, I, O, δ, λ) be a finite state machine, and let $\mathfrak{r}, \mathfrak{s} \subset S \times S$ be equivalence relations. Then $m^{even}(\mathfrak{r} + \mathfrak{s}) = m^{even}(\mathfrak{r}) + m^{even}(\mathfrak{s})$ and $m^{odd}(\mathfrak{r} + \mathfrak{s}) = m^{odd}(\mathfrak{r}) + m^{odd}(\mathfrak{s})$.

Observation 2: Let M = (S, I, O, δ, λ) be a finite state machine, and let $\mathfrak{r} \subset \mathfrak{s} \subset S \times S$ be equivalence relations. Then $m^{even}(\mathfrak{r}) \subset m^{even}(\mathfrak{s})$ and $m^{odd}(\mathfrak{r}) \subset m^{odd}(\mathfrak{s})$.

The following characterization of symmetric partition pairs is an immediate consequence of lemma 1.

Theorem 2: Let M = (S, I, O, δ, λ) be a finite state machine, and let $\mathfrak{r}, \mathfrak{s} \subset S \times S$ be equivalence relations. The pair $(\mathfrak{r}, \mathfrak{s})$ is a symmetric partition pair, if and only if there is an mm-pair $(\mathfrak{r}^*, \mathfrak{s}^*)$ with $\mathfrak{r}^* = \mathfrak{r}$ and $\mathfrak{s}^* \subset \mathfrak{s} \subset M(\mathfrak{r}^*)$.

With respect to problem OSTR it is important to note, that an mm-pair $(\mathfrak{r}^*, \mathfrak{s}^*)$ has the minimal intersection of all pairs in $\{(\mathfrak{r}, \mathfrak{s}) \mid \mathfrak{r} = \mathfrak{r}^* \text{ and } \mathfrak{s}^* \subset \mathfrak{s} \subset M(\mathfrak{r}^*)\}$, i.e. if $\mathfrak{r}^* \cap \mathfrak{s}^* \not\subset \mathfrak{e}$, then $\mathfrak{r} \cap \mathfrak{s} \not\subset \mathfrak{e}$ for all pairs in $\{(\mathfrak{r}, \mathfrak{s}) \mid \mathfrak{r} = \mathfrak{r}^* \text{ and } \mathfrak{s}^* \subset \mathfrak{s} \subset M(\mathfrak{r}^*)\}$. Consequently, if the set of mm-pairs does not provide a better solution than $(\mathfrak{id}, \mathfrak{id})$ for OSTR, then there is no better solution at all. Furthermore, using theorem 2 it is possible to derive all solutions from the solutions found in the set of mm-pairs. Therefore the search space can be restricted to mm-pairs, and the algorithm for OSTR will be based on an efficient procedure to enumerate the set of mm-pairs.

It is possible to construct the set of mm-pairs from the base relations $\mathfrak{B}(S) := \{\mathfrak{p}_{s,t} \mid s,t \in S\}$, where $\mathfrak{p}_{s,t} := \mathfrak{id} \cup \{(s, t), (t, s)\}$ is the equivalence relation identifying the states s and t in S and distinguishing all other states.

By lemma 1 an mm-pair $(\mathfrak{r}, \mathfrak{s})$ is characterized by $m^{even}(\mathfrak{r}) = \mathfrak{r}$ and $m^{odd}(\mathfrak{r}) = \mathfrak{s}$. Thus the set of all mm-pairs for a finite state machine M = (S, I, O, δ, λ) is described by $\mathfrak{R}_{mm}(M) := \{ (m^{even}(\mathfrak{r}), m^{odd}(\mathfrak{r})) \mid \mathfrak{r} \subset S \times S$ is an equivalence relation$\}$. Obviously, $\mathfrak{r} = \sum_{(s,t)\in\mathfrak{r}} \mathfrak{p}_{s,t}$ is true for any equivalence relation, and $\mathfrak{R}_{mm}(M) = \{ (m^{even}(\mathfrak{r}), m^{odd}(\mathfrak{r})) \mid \mathfrak{r} = \sum_{\mathfrak{p}\in\mathcal{P}} \mathfrak{p}, \mathcal{P} \subset \mathfrak{B}(S) \}$.

Therefore $\mathfrak{R}_{mm}(M)$ can be constructed by enumerating all possible sums of relations in $\mathfrak{B}(S)$ and calculating the corresponding mm-pair with the help of the m^{even}- and the m^{odd}-operator. However, this would require a computational effort of $O\left(2^{|S|^2}\right)$. In fact, only those relations in $\mathfrak{B}(S)$ have to considered which lead to different mm-pairs. To make this more precise a partial ordering „≤“ on equivalence relations is introduced as follows:

Definition 4: Let M = (S, I, O, δ, λ) be a finite state machine, and let $\mathfrak{r}, \mathfrak{s} \subset S \times S$ be equivalence relations. Then $\mathfrak{s}$ is said to *dominate* $\mathfrak{r}$ ($\mathfrak{r} \le \mathfrak{s}$), if and only if $m^{even}(\mathfrak{r}) \subset m^{even}(\mathfrak{s})$. $\mathfrak{r}$ and $\mathfrak{s}$ are called *mm-equivalent* ($\mathfrak{r} \sim \mathfrak{s}$), if both $\mathfrak{r} \le \mathfrak{s}$ and $\mathfrak{s} \le \mathfrak{r}$ are true.

It is easily verified that ~ is in fact in equivalence relation, and observations 1 and 2 provide the following observation:

Observation 3: Let M = (S, I, O, δ, λ) be a finite state machine, and let $\mathfrak{r}, \mathfrak{s} \subset S \times S$ be equivalence relations. Then $\mathfrak{r} \le \mathfrak{s}$ implies $m^{even}(\mathfrak{s}) = m^{even}(\mathfrak{r} + \mathfrak{s})$.

For mm-equivalent relations $\mathfrak{r} \sim \mathfrak{s}$ observation 3 yields $m^{even}(\mathfrak{r}) = m^{even}(\mathfrak{r} + \mathfrak{s}) = m^{even}(\mathfrak{s})$ and therefore the same mm-pair is obtained from $\mathfrak{r}$ and $\mathfrak{s}$. To construct the set of all mm-pairs it is sufficient to consider the quotient space $\mathfrak{B}(S)/\sim$.

Theorem 3: Let M = (S, I, O, δ, λ) be a finite state machine, and let ~ denote the mm-equivalence of relations.

Then the set of mm-pairs for M is characterized by $\mathcal{R}_{mm}(M) =$

$$\{(m^{even}(\mathfrak{r}), m^{odd}(\mathfrak{r})) \mid \mathfrak{r} = \sum_{\mathfrak{p}\in\mathcal{P}} \mathfrak{p}\ ,\ \mathcal{P} \subset \mathfrak{B}(S)/\sim \}.$$

Making use of theorem 3 and observation 1 the following basic search tree (V, E) can be constructed to enumerate all mm-pairs. First the quotient $\mathfrak{B}(S)/\sim$ is computed. This computation also provides the set $\mathfrak{m}^{even} := \{ m^{even}(\mathfrak{r}) \mid \mathfrak{r} \in \mathfrak{B}(S)/\sim \}$. With $\mathfrak{m}^{even} = \{\mathfrak{m}_1, \mathfrak{m}_2, \dots\}$ ordered arbitrarily (V, E) is defined by:

$V := \mathcal{P}(\mathfrak{m}^{even})$

$E := \{ (\mathcal{M}_1, \mathcal{M}_2) \in V \times V \mid \mathcal{M}_2 = \mathcal{M}_1 \cup \{\mathfrak{m}_k\}$ with $k > \max\{i \mid \mathfrak{m}_i \in \mathcal{M}_1\} \}$

The root of the search tree is $\emptyset$. Each vertex in this tree corresponds to a subset $\mathcal{M} \subset \mathfrak{m}^{even}$ and provides an mm-pair $(\mathfrak{r}, m(\mathfrak{r}))$ with $\mathfrak{r} := \sum_{\mathfrak{m}_i\in\mathcal{M}} \mathfrak{m}_i$. The tree has $O(2^{|\mathfrak{m}^{even}|})$ vertices and in general $|\mathfrak{m}^{even}|$ is much smaller than $|S|^2$, which already leads to an enormous reduction of the computational effort compared to the straightforward approach. In addition to that, observation 3 shows that edges $(\mathcal{M}, \mathcal{M} \cup \{\mathfrak{m}\})$ in the search tree with $\mathfrak{m} \le \mathfrak{r} := \sum_{\mathfrak{m}_i\in\mathcal{M}} \mathfrak{m}_i$ can be omitted, since in this case $m^{even}(\mathfrak{r} + \mathfrak{m}) = m^{even}(\mathfrak{r})$. Therefore a reduced tree (V*, E*) with

$E^* := \{ (\mathcal{M}_1, \mathcal{M}_2) \in V \times V \mid \mathcal{M}_2 = \mathcal{M}_1 \cup \{\mathfrak{m}_k\}$ with $k > \max\{i \mid \mathfrak{m}_i \in \mathcal{M}_1\}$ and $\mathfrak{m}_k \not\le \sum_{\mathfrak{m}_i\in\mathcal{M}_1} \}$

is sufficient to get all mm-pairs.

To solve problem OSTR the tree (V*, E*) can be traversed using a breadthfirst or depthfirst strategy. For each vertex $\mathcal{M}$ the relation $\mathfrak{r} := \sum_{\mathfrak{m}_i\in\mathcal{M}} \mathfrak{m}_i$ is calculated. If $m(\mathfrak{r}) \cap \mathfrak{r} \subset \mathfrak{e}$ is true, then the mm-pair $(\mathfrak{r}, m(\mathfrak{r}))$ is a solution for OSTR. The costs

$$\lceil \log_2 |S/m(\mathfrak{r})| \rceil + \lceil \log_2 |S/\mathfrak{r}| \rceil \text{ and } \left| \frac{|S/m(\mathfrak{r})|}{|S/\mathfrak{r}|} - 1 \right|$$

are calculated for this pair and compared to the lowest costs obtained so far. Finally the solution with minimal costs is selected to realize the specification.

The specific requirements of problem OSTR provide another criterion to prune the search tree and make the basic procedure computationally more efficient.

Lemma 2: Let M = (S, I, O, δ, λ) be a finite state machine, and let (V*, E*) be the search tree defined above. For $(\mathcal{M}_1, \mathcal{M}_2) \in E$ let $\mathfrak{r}_1 := \sum_{\mathfrak{m}_i\in\mathcal{M}_1} \mathfrak{m}_i$ and $\mathfrak{r}_2 := \sum_{\mathfrak{m}_i\in\mathcal{M}_2} \mathfrak{m}_i$. If $m(\mathfrak{r}_1) \cap \mathfrak{r}_1 \not\subset \mathfrak{e}$, then $m(\mathfrak{r}_2) \cap \mathfrak{r}_2 \not\subset \mathfrak{e}$.

As a consequence of lemma 2, once a node $\mathcal{M}$ in the searchtree with $m(\mathfrak{r}) \cap \mathfrak{r} \not\subset \mathfrak{e}$ is reached, all of its successors have this property and the subtree rooted at $\mathcal{M}$ can be discarded.

4 Experimental results

The algorithm for problem OSTR described in section 3 has been implemented as a depthfirst procedure and has been applied to the finite state machine benchmarks distributed for the International Workshop on Logic Synthesis '93 [21]. For incompletely specified finite state machines don't care transition were fixed to transitions with next state = present state and output don't cares were set to zero. For 23 examples a nontrivial solution of OSTR (i.e. a solution different from $(\mathfrak{id}, \mathfrak{id})$) could be found. Table 1 shows the results in more detail for some of these examples.

Name	|S|	|S_1|	|S_2|	# FFs (conventional)	# FFs (OSTR)
bbara	10	7	7	8	6
bbsse*	16	8	8	8	6
beecount*	7	4	6	6	5
dk16	27	21	23	10	10
dk512	15	11	14	8	8
ex1*	20	17	19	10	10
mark1*	15	11	6	8	7
planet*	48	44	45	12	12
s1488	48	44	45	12	12
s1494	48	44	45	12	12
s208	18	10	10	8	8
shiftreg	8	2	4	6	3
sse*	16	8	8	8	6
tav	4	2	2	4	2
tbk	32	16	16	10	8

Table 1: Results of depthfirst search procedure for problem OSTR (incompletely specified examples are marked with an asterisk).

Column 2 contains the number of states of the specification. Columns 3 and 4 show the number of states in the factors S_1 and S_2 of the best realization found, and

columns 5 and 6 list the required number of flipflops for a conventional BIST and for an optimized BIST structure based on the presented approach. The results show that for eight examples the number of flipflops required for the optimized structure is less than for a conventional BIST. For shiftreg and tav even the lower bound $|S_1| \cdot |S_2| = |S|$ is achieved and the number of flipflops is reduced to 50%. In terms of hardware costs the gain is even higher, because the transparent register for the conventional solution is more costly to implement than the registers used for the presented approach.

Since the overall hardware costs for both alternatives can only be compared after state coding and logic minimization, a state coding algorithm is currently being developed which takes advantage of the self-testable decomposition. A prototype version of this algorithm followed by ESPRESSO was used to determine the required PLA area for the circuits C_1, C_2 and the output function λ [5]. The PLA area for the circuit C of a conventional implementation was determined by applying NOVA to the finite state machine specifications [24]. In both cases the PLA area was estimated by $(2 \cdot i + o) \cdot p$, with i, o, p denoting the number of inputs, the number of outputs and the number of product terms, respectively. Table 2 shows the results.

Name	area for C_1	area for C_2	area for λ	total area Σ	area for C	$\frac{\Sigma}{C}$
bbara	320	320	108	748	550	1.36
bbsse*	351	252	546	1149	990	1.16
beecount*	126	90	8	224	228	0.98
dk16	1045	931	432	2408	1584	1.52
dk512	224	182	114	520	323	1.61
ex1*	1053	1161	1815	4029	2652	1.52
planet*	2272	2304	3420	7996	4539	1.76
s1488	2738	2701	3355	8794	7473	1.18
s1494	2808	2736	3528	9072	7897	1.15
s208	176	176	210	562	780	0.72
shiftreg	3	12	3	18	96	0.19
sse*	351	252	546	1149	990	1.16
tav	3	3	128	134	198	0.68

Table 2: Comparison of PLA areas for conventional implementations and optimized self-testable realizations.

The results show that for all examples the PLA area for the circuits C_1, C_2 and the output function is smaller than the PLA area for C. Therefore it can be assumed that in a multi-level implementation the critical path in each combinational logic block of the self-testable implementation is shorter than in the logic block of a conventional implementation. Higher clock rates and a higher performance can thus be expected. For four examples even the total PLA area for the circuits C_1, C_2 and the output function together is less than the PLA area for the conventional implementation. Moreover, for all examples the total PLA area is smaller than the area obtained by simply doubling the original network C. This is true even for planet, s1488 and s1494 were the specification with 48 states could only be decomposed into factors with 44 and 45 states, respectively.

Table 3 compares the computational effort for the presented algorithm and the basic algorithm used in [18]. For both procedures the size |V| of the search tree and the number of the nodes that have to be investigated to find the optimal solution are listed.

Name	$\|S\|$	$\log_2 \|V\|$	# nodes investigated	$\log_2 \|V\|$ [18]	# nodes investigated [18]
bbara	10	7	19	43	815
bbsse*	16	43	timeout	-	-
beecount*	7	15	85	-	-
dk16	27	20	15437	206	337041
dk512	15	15	221	56	343853
ex1*	20	10	53	-	-
mark1*	15	45	timeout	-	-
planet*	48	9	75	-	-
s1488	48	9	75	-	-
s1494	48	9	75	-	-
s208	18	28	255	-	-
shiftreg	8	17	811	7	49
sse*	16	43	timeout	-	-
tav	4	5	21	7	47
tbk	32	11	55	-	-

Table 3: Computational effort for exhaustive search.

The results confirm that the presented approach is able to characterize the search space significantly better than the procedure used in [18]. For example for dk16 the search tree can be reduced from 2^{206} to 2^{20} nodes. Moreover, the impact of lemma 2 on the computational effort can be seen clearly. For almost all examples the optimal solution could be found by investigating only a small number of nodes. The limit of 500 000 nodes (timeout) was reached only by bbssee, mark1 and sse. But as table 1 indicates, for these examples a large number of possible solutions could be found before the limit was reached. All examples which did not provide a non-trivial solution could be identified quickly.

5 Conclusions

Pipeline-like controller structures implement the states of the specification by two different multi-functional system registers. A self-test can be performed in two sessions without any extra hardware by alternatingly using the registers for test pattern generation and signature analysis. This architecture increases the fault coverage as well as the system performance.

An efficient algorithm has been presented to generate minimal pipelined realizations from state transition diagrams. The proposed algorithm relies on algebraic structure theory to construct realizations which can be trivially implemented by a self-testable structure. The efficiency of the procedure is ensured by a very precise characterization of the space of suitable realizations, the theoretical basis of which is the newly introduced concept of mm-pairs.

The experimental results show that in general a shorter critical path and thus a higher performance can be expected. In many cases the number of flipflops is less than the respective number for a conventional BIST, and in some cases even the total PLA area is reduced. This confirms that not only higher speed and fault coverage can be obtained this way, but also area can be saved.

References

1 V. D. Agrawal, C. R. Kime, K. K. Saluja: A Tutorial on Built-In Self-Test, Part 1: Principles, IEEE Design & Test of Computers, Vol. 10, No. 1, March 1993, pp. 73-82

2 P. Ashar, S. Devadas: Irredundant Interacting Sequential Machines Via Optimal Logic Synthesis, IEEE Trans. on CAD, Vol. 10, No. 3, March 1991, pp. 311-325

3 P. Ashar, S. Devadas, A. R. Newton: A Unified Approach to the Decomposition and Re-decomposition of Sequential Machines, Proc. 27th ACM/IEEE Int. Design Automation Conf., 1990, pp. 601-606

4 Z. Barzilai, D. Coppersmith, A. L. Rosenberg: Exhaustive Generation of Bit Patterns with Applications to VLSI Self-Testing, IEEE Trans. on Computers, Vol. c-32, No. 2, February 1983, pp. 190 - 194

5 R. K. Brayton, G. D. Hachtel, C. T. McMullen: Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Publishers, Boston, The Hague, Dordrecht, Lancaster, 1984

6 R. K. Brayton et al.: MIS: A Multiple-Level Logic Optimization System, IEEE Trans. on CAD, Vol. CAD-6, No. 6, 1987, pp. 1062-1081

7 V. D. Agrawal, K.-T. Cheng: Finite State Machine Synthesis with Embedded Test Function, Journal of Electronic Testing Theory and Applications, Vol. 1, No. 3, October 1990, pp. 221-228

8 K.-T. Cheng, V. D. Agrawal: State Assignment for Testable Design, Int. Journal of Computer Aided VLSI Design, Vol. 3., March 1991

9 W. Daehn, J. Mucha: Hardware Test Pattern Generation for Built-In Test, Proc. IEEE Int. Test Conf., Philadelphia, 1981, pp. 100 - 113

10 S. Devadas, K. Keutzer: A Unified Approach to the Synthesis of Fully Testable Sequential Machines, IEEE Trans. on CAD, Vol. 10, No. 1, January 1991, pp. 39-50

11 S. Devadas et al.: MUSTANG: State Assignment of Finite State Machines Targeting Mulitlevel Logic Implementations, IEEE Trans. on CAD, Vol. 7, No. 12, Dec. 1988, pp. 1290-1300

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

13 B. Eschermann, H.-J. Wunderlich: Optimized Synthesis Techniques for Testable Sequential Circuits, IEEE Trans. on CAD, Vol. 11, No. 3, March 1992, pp. 301-312

14 R. Gage: Structured CBIST in ASICS; Proc. IEEE Int. Test Conf., Baltimore, Maryland, 1993, pp. 332-338

15 Martin Geiger, Thomas Müller-Wipperfürth: FSM Decomposition Revisited: Algebraic Structure Theory Applied to MCNC Benchmark FSMs, Proc. 28th ACM/IEEE Design Automation Conf., San Francisco, 1991, pp. 182-185

16 J. Hartmanis, R. E. Stearns: Algebraic Structure Theory of Sequential Machines, Prentice Hall, Englewood Cliffs, 1966

17 S. Hellebrand, H.-J. Wunderlich, O. Haberl: Generating Pseudo-Exhaustive Vectors for External Testing; Proc. IEEE Int. Test Conf., Washingtion, D. C., 1990, pp. 670-679

18 S. Hellebrand, H.-J. Wunderlich: Synthesis of Self-Testable Controllers, in: Proc. EDAC/ETC/EuroAsic '94, Paris, Feb. 1994, pp. 580-585

19 K. Kim, D. Ha, J. Tront: On Using Signature Registers as Pseudorandom Pattern Generators in Built-in Self Testing, IEEE Trans. on CAD, Vol. 7, 1988, pp. 919-928

20 B. Koenemann, J. Mucha, G. Zwiehoff: Built-in Logic Block Observation Techniques, Proc. IEEE Int. Test Conf., Cherry Hill, N. J., 1979, pp. 37 - 41

21 K. McElvain: IWLS'93 Benchmark Set: Version 4.0, distributed as part of the IWLS'93 benchmark distribution

22 I. Pomeranz, S. M. Reddy: Design and Synthesis for Testability of Synchronous Sequential Circuits Based on Strong-Connectivity, Proc. IEEE 23rd Int. Symp. on Fault-Tolerant Computing, FTCS-23, Toulouse, June 1993, pp. 492-501

23 G. Saucier, M. C. De Paulet, P. Sicard: ASYL: A Rule-Based System for Controller Synthesis, IEEE Trans. on CAD, Vol. CAD-6, No. 6, Nov. 1987, pp. 1088-1097

24 T. Villa, A. Sangiovanni-Vincentelli: NOVA: State Assignment of Finite State Machines for Optimal Two-Level Logic Implementations, Proc. 26th ACM/IEEE Design Automation Conf., Las Vegas, 1989, pp. 327-332

25 P. Weiner, E. J. Smith: Optimization of Reduced Dependencies for Synchronous Sequential Machines, IEEE Trans. on Electronic Computers, Vol. EC-16, No. 6, Dec. 1967, pp. 835-847

26 H.-J. Wunderlich: Self Test Using Unequiprobable Random Patterns, Proc. IEEE 17th Int. Symp. on Fault-Tolerant Computing, FTCS-17, Pittsburgh, 1987, pp. 258-263