Center for Reliable Computing

TECHNICAL REPORT

Synthesis-for-Scan and Scan Path Ordering

By Robert B. Norwood and Edward J. McCluskey

97-3

(CSL TR # 97-740)

November 1997

Center for Reliable Computing
Gates 236
Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305

Abstract:

Designing a scannable circuit is typically a two-step process. The circuit is first designed to meet the functional specifications, without taking the scan path into consideration. The circuit is then analyzed, and the scan path is inserted based on this analysis. When the scan path is considered during the synthesis of a circuit rather than after the synthesis, the overhead due to the scan path can be reduced. We present beneficial scan, a synthesis-for-scan technique that orders the scan path(s) during logic synthesis to maximize the amount of sharing that can take place between the functional and test logic and thereby minimize the area and performance overhead due to the scan path. We also present modifications to the state assignment algorithms to consider beneficial scan path insertion during this step.

Funding:

This research was supported in part by the Advanced Research Projects Agency under Contract No. DABT63-94-C-0045, and in part by the National Science Foundation under Grant No. MIP-9107760.

1 Introduction

Scan paths are a commonly used design-for-test technique that improve the testability of sequential designs. A scan path provides direct access to the sequential elements, greatly improving the controllability and observability of the circuit. With a scan path inserted it is not necessary to generate test patterns for a sequential circuit, a difficult and time consuming task, since the circuit can be treated as a combinational circuit during testing.

There are a variety of scan techniques in use, each with its positive and negative aspects [McCluskey 86]. Scan designs, in general, have some costs associated with them. These costs are:

1. Increased area due to the additional circuitry and interconnect required to construct the scan path.
2. Possible performance penalty due to the increased propagation delay in the scannable bistables or due to the loading of the scan path interconnect.
3. Additional pins for the test signals.
4. Increased testing time due to serialization of the test patterns.

Various techniques have been proposed to reduce the costs associated with scan designs. Specially designed scan elements [Schultz 85] [Zasio 85] [Bhavsar 86] [Giles 91] [Mukund 91] can reduce the additional area or interconnect loading. Careful ordering of the scan path elements can reduce the interconnect or testing time. The long test application time due to the serial shifting of the test data can be reduced by ordering the scan path so that the frequently accessed elements are closer to the scan-in/scan-out pins, thereby making those elements more easily accessed [Gupta 91] [Narayanan 92] [Narayanan 93]. This ordering is done after the circuits have been designed, and the additional interconnect overhead can be high. The test application time can also be reduced by exploiting the inherent sequential nature of the circuit and applying multiple

system clock cycles to the circuit for each test vector that is scanned in [Pradhan 92] instead of applying a single system clock cycle as is standard, but the test generation time is increased due to the use of sequential test patterns. By giving up some of the controllability and observability of a fully scanned design, partial-scan designs attempt to reduce the overhead by making only a subset of the system bistables scannable. Since fewer bistables need to be modified, the area overhead of partial scan is typically less than that of full scan [Agrawal 88] [Chickermane 90] [Gupta 90] [Lee 90] [Chakradhar 94], but the circuit is no longer strictly combinational during test, and sequential test pattern generation is necessary. Cost-free scan [Lin 95] attempts to choose a primary input vector to sensitize paths through the combinational logic to form portions of the scan path. The scan path overhead is reduced since fewer bistables must be modified to form the scan path. However, the overhead reduction is constrained by the limited number of cost-free situations in many circuits. Section 3.1.5 shows how cost-free scan can be considered as a subset of beneficial scan. These techniques generally approach the problem of reducing the test overhead after the circuit has been designed.

Other work has considered scan path insertion during the synthesis of the circuit. Testability constraints can be used during synthesis to embed scan paths in a circuit [Cox 94] [Cox 95] Algorithms similar to those of automatic test pattern generation can transform the circuit so that a scan path is embedded and any rule violations corresponding to the constraints are repaired. Instead of modifying the circuit structure directly, the finite state machine (FSM) description of the circuit can be augmented to embed a shift register into the state machine, and the circuit can then be synthesized [Vinnakota 92] [Kanjilal 93]. In this case, a FSM description of the circuit must be available, and it is not obvious how a particular modification to the FSM description will affect the size of the final synthesized circuit. Bistable selection for partial scan can also be performed during synthesis [Bhatia 93].

We present a technique, called *beneficial scan*, that combines circuit synthesis and scan path insertion into one step. Knowledge of the circuit functions is used to order the scan path elements in such a fashion that the functional logic and the test logic are shared, reducing the cost of the scan path. The scannable bistables are assumed to be implemented with MD flip-flops for this discussion. An *MD flip-flop* [McCluskey 86], shown in Fig. 1, is formed by placing a multiplexer at the data input of the flip-flop to allow the selection of two different data inputs — either d_0 for normal system operation or d_1 for test mode — based on the test select, *T*.

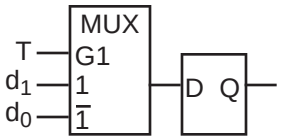

Figure 1. MD flip-flop

Earlier work [Norwood 96] introduced beneficial scan and described how a beneficial scan path can be inserted into a circuit during synthesis. Here we provide some more details on this procedure and discuss how state assignment of symbolic states can be targeted to beneficial scan.

This work is based on previous work done at the Center for Reliable Computing looking at reducing the overhead of a built-in self test (BIST) design [Avra 93] [Avra 94] by ordering the scan path so that the functional logic and the test logic — a multiple input signature register (MISR) — can be shared. Beneficial scan uses the same concept of ordering the scan path so that the functional and the test logic can be shared but focuses on reducing the overhead due to the scan path itself.

2 Overview

This synthesis-for-scan method takes the scan path implementation into account during the synthesis of the circuit. A scan path order is found that maximizes the sharing of the functional and test logic, reducing the overhead due to the scan path. The techniques described here assume a full-scan design.

Every flip-flop input in a sequential circuit has some relationship with every other flip-flop output, as well as with every primary input. Some of these relationships allow some, or all, of the test logic to be shared with the functional logic. When functional and test logic can be shared, it is called a *beneficial relationship.* Other, *non-beneficial*, relationships do not allow the logic to be shared and a multiplexer, or multiplexer equivalent, must be inserted to make a flip-flop scannable during test operations. The various relationships, both beneficial and non-beneficial, are described in detail in Sec. 3. Each pair of flip-flops in the design is classified according to these relationships, and a *relationship graph*, described in Sec. 4, is constructed.

Once each flip-flop is classified based on the relationships between it and every other flip-flop, as well as on the relationships between it and every primary input, an order for the scan path is determined using these relationships. Each type of relationship has a cost associated with it, where the cost reflects the logic overhead required to make the flip-flop scannable. The goal is to obtain an ordering that minimizes the cost and thereby maximizes the logic sharing.

After the order is determined, the circuit is synthesized. Synthesis occurs after the scan path ordering so that the equations for the flip-flop inputs may be factored in such a way as to take advantage of the sharing of the test and functional logic. The factoring is based on the Shannon expansion, as explained in Sec. 3. The final scan path obtained in this manner may contain inversions between flip-flops, but these inversions may be compensated for during test pattern generation by inverting appropriate data bits in the test vector. Forbidding inversions may result in a less optimal ordering and a reduction in the amount of logic shared.

We illustrate our synthesis technique with a simple example. The circuit in Fig. 2 has a traditional scan path where each flip-flop has a multiplexer added to make the flip-flop scannable. These additional multiplexers are highlighted in the figure. Examination of the circuit based on the criteria to be discussed in Sec. 3 shows that the input of *flip-flop 1*

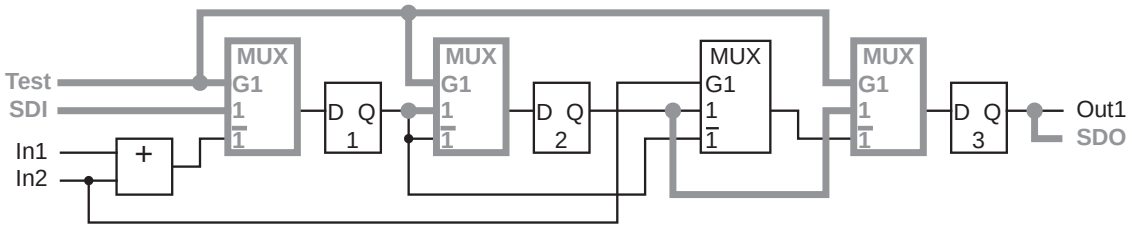

Figure 2. Example circuit with a traditional scan path

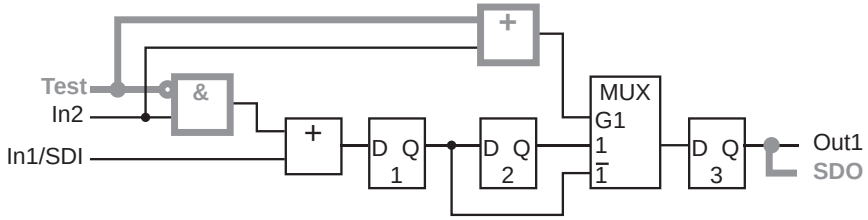

Figure 3. Example circuit with beneficial scan

Table 1. Flip-flop relationships for example circuit in Fig. 2

Output of	Input of flip-flop 1	Input of flip-flop 2	Input of flip-flop 3
flip-flop 1	—	Beneficial	Beneficial
flip-flop 2	Non-Beneficial	—	Beneficial
flip-flop 3	Non-Beneficial	Non-Beneficial	—
input *In1*	Beneficial	Non-Beneficial	Non-Beneficial
input *In2*	Beneficial	Non-Beneficial	Beneficial

has a beneficial relationship with primary inputs *In1* and *In2*. The input of *flip-flop 2* has a beneficial relationship with *flip-flop 1*. The input of *flip-flop 3* has a beneficial relationship with *flip-flop 1*, *flip-flop 2*, and primary input *In2*. These relationships are summarized in Table 1.

Figure 3 shows the same circuit, but with a beneficial scan path inserted taking advantage of the beneficial relationships. The scan path takes advantage of three beneficial relationships — replacing three multiplexers with one AND gate and one OR gate and reducing the interconnect between flip-flops — resulting in a circuit with less area and probably less delay.

Section 3 defines the various beneficial and non-beneficial relationships used and discusses the cost associated with each case. Section 4 describes our algorithm for determining the scan path order. Section 5 discusses the implementation of the technique

and contains results for some benchmark circuits. Section 6 introduces a state assignment technique that targets beneficial scan, and Sec. 7 presents some results for this technique. Section 8 summarizes this work.

3 Scan element classifications

The input equation for each flip-flop in a circuit can be expressed as a function of flip-flop outputs and primary inputs. Based on these input equations each flip-flop in a synchronous circuit has some relationship with every other flip-flop and primary input in the circuit. This relationship may be trivial, as in the case when a flip-flop input expression is vacuous in another flip-flop output. This relationship could be a simple classification whether one input equation includes the output of another flip-flop, or it could be based on more complex input function characteristics.

Some of these relationships allow some, or all, of the test logic to be shared with the functional logic. These are *beneficial relationships*. Other relationships do not allow the logic to be shared, and a multiplexer, or multiplexer equivalent, must be inserted to make a flip-flop scannable. These are *non-beneficial relationships*.

For our beneficial scan algorithm, we base the majority of the relationships between flip-flops on Shannon's expansion theorem. The Shannon expansion transforms a function, $f(x_1, x_2,\ldots, x_n)$, based on residues. The x_i-residue, also called the positive-phase cofactor of f with respect to x_i, is defined as

$$f_{x_i}(x_1, x_2,\ldots, x_n) = f(x_1, x_2,\ldots, x_i = 1,\ldots, x_n)$$

and the x'_i-residue, or negative-phase cofactor of f with respect to x_i, as

$$f_{x'_i}(x_1, x_2,\ldots, x_n) = f(x_1, x_2,\ldots, x_i = 0,\ldots, x_n).$$

The Shannon expansion is defined as

$$f(x_1, x_2,\ldots, x_n) = x_i\, f_{x_i} + x'_i\, f_{x'_i}.$$

Based on the Shannon expansion, each flip-flop j may be classified with respect to every other flip-flop i. The classifications are summarized in Table 2, and a more detailed description of each case is given in this section. Notation: D_j is the input of flip-

Table 2. Flip-flop classifications with beneficial cases in bold

Constants are '0' or '1'.

Q_i* indicates that the flip-flop output may be inverted.

Class	Q_i-residue	Q'_i-residue	Equation Form	New Scan Function	Ovhd
case B	constant	constant	$D_j=Q_i^*$	$D_j=Q_i^*$	none
case A	constant not constant	not constant constant	$D_j=Q_i^*+f_{Q'_i}$ $D_j=Q_i^*f_{Q_i}$	$D_j=Q_i^*+T'f_{Q'_i}$ $D_j=Q_i^*(T+f_{Q_i})$	AND OR
case X	$(f_{Q'_i})'$	$f_{Q'_i}$	$D_j=Q_i^*\oplus f_{Q'_i}$	$D_j=Q_i^*\oplus T'f_{Q'_i}$	AND
case M	(s + a)	(s' a)	$D_j=sQ_i^*+s'a$	$D_j=(s+T)Q_i^*+(s+T)'a$	OR
case N	f	f	$D_j=f$	$D_j=T'f+TQ_i$	MUX
case O	not constant	not constant	$D_j=Q_if_{Q_i}+Q'_if_{Q'_i}$	$D_j=T'f+TQ_i$	MUX

flop *j*. Q_i is the output of flip-flop *i*. *T* is the scan test select signal — '1' for scan mode, '0' for normal system operation. *f* is the function for normal system operation, that is, the input equation before scan is inserted. *g*, *s*, and *a* are arbitrary functions.

Some of these classifications (case B, case A, case X, and case M) are beneficial relationships and allow the test logic to be shared with the functional logic. The other classifications (case N and case O) are non-beneficial, and no logic may be shared. These four beneficial classifications include all situations where the sharing of the test logic and the functional logic causes a single AND or OR gate (or no gate) to be added to a flip-flop input to make the flip-flop scannable. The assumption is made that a two-to-one multiplexer has less area than two AND or two OR gates. Therefore, we do not consider situations in which more than one logic gate must be added since this would result in greater overhead than the simple addition of a multiplexer. This assumption is validated by examining several commercial technologies, such as the LSI G10p technology [LSI Logic 96]. Other relationships also exist that may be used to classify the flip-flops, and one, cost-free scan (a beneficial relationship), is described in Sec. 3.1.5.

3.1 Beneficial cases

The various beneficial classifications are further described in this section.

3.1.1 Case B

A logical buffer or inverter between two flip-flops, $D_j = Q_i$ or $D_j = Q'_i$, leads to a case B (*case B*uffer) classification. This is a beneficial relationship since no modifications need to be made to form a scan path between the two flip-flops, as shown in Fig. 4. In this case both the Q_i- and the Q'_i-residues are constant-value functions, either '0' or '1'.

Figure 4. Case B flip-flop before and after test logic is inserted

3.1.2 Case A

Flip-flop input functions of the form $D_j = Q_i f_{Q_i}$ or $D_j = Q'_i f_{Q'_i}$ are classified as case A (*case A*ND since there is an AND gate before flip-flop$_i$). This is a beneficial relationship since only an OR gate is required to form a scan path between flip-flop *i* and flip-flop *j*, as shown in Fig. 5. The Q'_i-residue is a constant '0', and the Q_i-residue is not a constant, or vice versa.

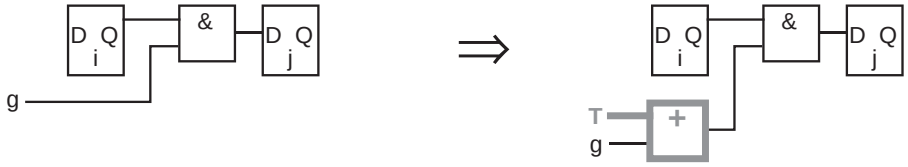

Figure 5. Case A flip-flop before and after test logic is inserted

The case A dual, also beneficial, has $D_j = Q_i + f_{Q'_i}$ or $D_j = Q'_i + f_{Q_i}$. In this case the Q'_i-residue is not a constant and the Q_i-residue is a constant '1', or vice versa. An AND gate is needed to form a scan path between the two flip-flops, as shown if Fig. 6.

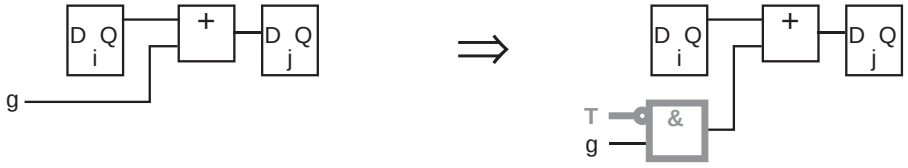

Figure 6. Dual case A flip-flop before and after test logic is inserted

3.1.3 Case X

Case X (*case X*OR because of the exclusive-OR before the flip-flop) relationships have flip-flop input functions of the form $D_j = Q_i\, f'_{Q'_i} + Q'_i\, f_{Q'_i}$ or $Q'_i f'_{Q'_i} + Q_i\, f_{Q'_i}$. This is equivalent to $D_j = Q_i \oplus f_{Q'_i}$ or $D_j = Q_i \oplus f'_{Q'_i}$. The Q_i -residue is not a constant and is equal to the complement of the Q'_i-residue. If flip-flop j follows flip-flop i in the scan path, then only a single AND (or OR) gate is needed to form the scan path, as shown in Fig. 7.

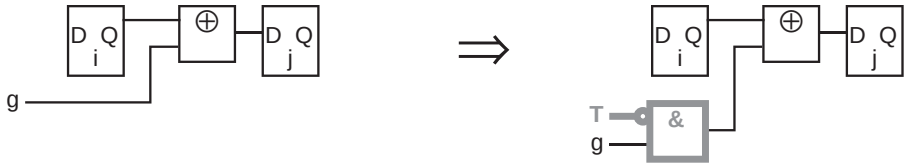

Figure 7. Case X flip-flop before and after test logic is inserted

3.1.4 Case M

Flip-flop input equations of the form $D_j = s\, Q_i + s'\, a$ or $D_j = s\, Q'_i + s'\, a$, where s and a are arbitrary equations, are classified as case M (*case M*ultiplexer) relationships. Case M relationships are beneficial and require the addition of an OR gate to form the scan path, as shown in Fig. 8. This situation essentially corresponds to a multiplexer in the functional logic, and the multiplexer select line is modified to achieve the scan function.

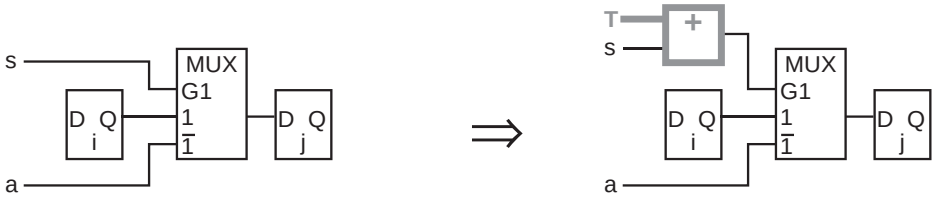

Figure 8. Case M flip-flop before and after test logic is inserted

3.1.5 Cost-free scan

As previously mentioned, the relationships between pairs of flip-flops can be based on something other than Shannon expansions. One such additional classification is “cost-free scan”. *Cost-free scan* [Lin 95] is a technique that allows the reuse of combinational logic for scan by exploiting the controllability of primary inputs. The circuit is analyzed

and a primary input vector, called the enabling vector, is selected to enable the maximum number of scan paths that can be sensitized between flip-flops. Only one enabling vector may be used, and because of this, the overhead reduction achievable with cost-free scan alone is limited as compared to beneficial scan where cost-free scan can be included as a subset of the relationships. Cost-free scan is a beneficial relationship since no additional logic is needed to form the scan path. Fig. 9 illustrates the technique. When input *A* is held at '1' during test, a scan path is formed from flip-flop *i* to flip-flop *j*.

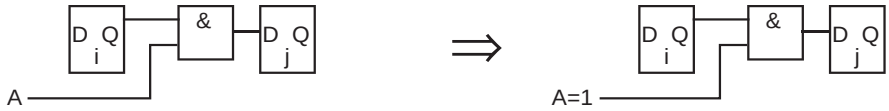

Figure 9. Cost-free scan flip-flop before and after test logic is inserted

3.2 Non-beneficial cases

There are also two non-beneficial cases.

3.2.1 Case N

A case N (*case N*one because there are no interesting relationships) relationship occurs when D_j is vacuous in Q_i — D_j has no dependence on Q_i. Case N relationships are not beneficial. If flip-flop *j* follows flip-flop *i* in the scan path, then a multiplexer, or multiplexer equivalent, must be added to the input of flip-flop *j* to make the flip-flop scannable, as shown in Fig. 10.

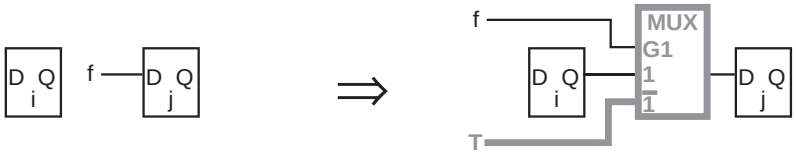

Figure 10. Case N or case O flip-flop before and after test logic is inserted

3.2.2 Case O

A case O (*case O*ther) relationship is also non-beneficial, but D_j is a function of Q_i, $D_j = Q_i f_{Q_i} + Q'_i f_{Q'_i}$. However, there is no simple relationship between the flip-flops that results in less overhead than occurs with the addition of a multiplexer. In these cases, a multiplexer is used to form the scan path, as in Fig. 10. This case is the default for scan

path insertion when not taking advantage of beneficial relationships to order the scan path.

4 Scan path ordering

4.1 Single scan paths

Once the flip-flops have been classified, a *relationship graph* is constructed. A relationship graph is a weighted, directed graph with a node for each flip-flop and each primary input and edges representing the relationships between the flip-flops and inputs. The edges follow the direction of the shift during the scan operation; therefore, the head of an edge is the node for flip-flop *i*, or input *i*, and the tail is the node for flip-flop *j*. Each flip-flop node has a weighted edge to and from every other flip-flop node, as well as one from every primary input node. An example relationship graph is shown in Fig. 11 with a possible scan path highlighted.

The edges are weighted to show the cost of adding test logic between the two flip-flops. Case B and cost-free flip-flops need no additional logic to perform the scan function. Case A, case X and case M flip-flops require the addition of a single gate to add scan capability. Case N and case O flip-flops must have an entire multiplexer added. Based on these overheads, case N and case O edges have a large weight, case B and cost-free edges have a very small weight, and the other cases are weighted in-between.

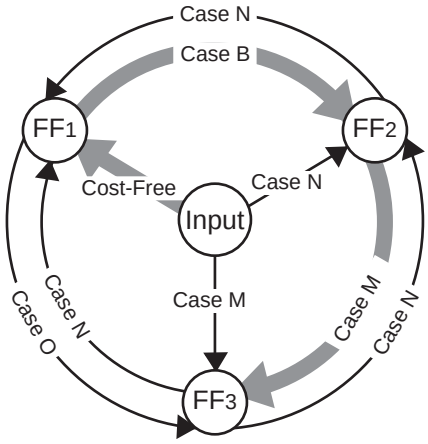

Figure 11. Relationship graph

The goal is to find a minimal-weight path through the graph that starts at an input node and covers all the flip-flop nodes. For a circuit with *p* primary inputs and *n* flip-flops, there are $p \bullet n!$ possible scan path orderings since there are $n!$ ways to order the flip-flops and *p* different inputs to use to scan in data. An exact solution is computationally expensive, so a heuristic is required.

Certain characteristics of the relationship graph allow heuristics to perform very well.

- Any ordering of the flip-flops is a valid order for the scan path.
- The cost of choosing any particular flip-flop has little dependence on previous decisions, i.e., a poorly chosen flip-flop will only affect the cost of one other segment of the scan path.

We have implemented a greedy approach, which is described as follows:

1. Find a lowest-weight edge from a primary input to a flip-flop. Make this primary input the scan data input and make the flip-flop the first in the scan path.
2. Find a lowest-weight edge from the last flip-flop in the scan path to another flip-flop not already in the scan path. Make this new flip-flop the last flip-flop in the scan path.
3. Repeat step 2 until all flip-flops are added to the scan path.

This basic greedy algorithm can be modified to make better choices at each step, such as giving preference to flip-flops with only one beneficial relationship over flip-flops with multiple beneficial relationships in order to leave more options open for later or making sure that a selected flip-flop is not the only beneficial case for another flip-flop. Modifications such as these help guide the algorithm to a better result.

The algorithm attempts to reduce the number of additional test pins needed by using one of the existing primary inputs as the scan-data-in pin. If there is a need to have an explicit scan-data-in pin, a new primary input node may be added to the relationship

graph while the other primary input nodes are removed. This new input node will have a case N edge to every flip-flop node.

4.2 Multiple scan paths

The approach just described will give an ordering for a single scan path, with one scan data input and one scan data output Multiple scan paths, with multiple scan data inputs and outputs, can also be formed. Multiple scan path orderings take advantage of an additional characteristic of the relationship graph:

- Beneficial relationships with a primary input tend to be more prevalent than beneficial relationships with other flip-flops.

Even if a flip-flop has no beneficial relationships with other flip-flops, examination of benchmark circuits has shown that it is likely that there will be a beneficial relationship with one or more primary inputs. By allowing multiple scan paths, a lower cost scan path ordering can be obtained since more beneficial relationships with primary inputs will be included.

Our algorithm for multiple scan paths is the same as that for single scan paths, with a slight modification to step 2:

1. Find a lowest-weight edge from a primary input to a flip-flop. Make this primary input the scan data input, and make the flip-flop the first in the scan path.
2. Find a lowest-weight edge from the last flip-flop in any scan path to another flip-flop not already in any scan path. Find a lowest-weight edge from an unused primary input to a flip-flop not already in any scan path. If the weight of the flip-flop to flip-flop edge is less than the weight of the primary input to flip-flop edge, then make this new flip-flop the last flip-flop in the appropriate scan path. Otherwise, make the primary input the scan data input of a new scan path and make the flip-flop the first flip-flop in this new scan path.
3. Repeat step 2 until all flip-flops are added to the scan path(s).

As with the single scan paths, the algorithm tries to use existing primary inputs as the scan-data-in pins. Any additional logic sharing achieved by using multiple scan paths comes from this sharing of the inputs. Explicit scan-data-in pins can be forced as described above.

The algorithm only adds another scan path if there is some additional logic sharing gain. A specific number of scan paths may be forced with some degradation in the results. The algorithm also tries to balance the paths without degrading the results. If there is more than one equally attractive choice during step 2, the flip-flop is chosen to balance the existing scan paths. Completely balanced paths may be forced, but the resulting order will not be optimal.

5 Implementation and results

We have implemented the beneficial scan algorithm in *sis* [Sentovich 92] to order flip-flops for single, or multiple, scan paths. We have used our algorithms to analyze the circuits and order the flip-flops in the scan paths. Existing *sis* functions are used to perform logic minimization and technology map the resulting circuits. The circuits are mapped to the Alliance technology library [Greiner 92].

Table 3 shows the results of the scan path ordering for some of the IWLS91 sequential, multi-level benchmark circuits [Yang 91]. Both single and multiple scan paths are shown unless the two results are the same and there is no benefit in having multiple scan paths for that particular circuit. For each benchmark circuit the number of scan paths is shown along with the breakdown of the flip-flop classifications used in the final scan path implementation.

Table 4 shows the area results of the circuits after placement and routing. Area and percentage overhead are shown for three different scan path implementations: 1) the circuit with a single scan path implemented entirely with MD flip-flops, 2) the circuit with a single, beneficially ordered scan path, and 3) the circuit with multiple, beneficially

Table 3. Number of beneficial relationships in final scan path(s)

Circuit	Number of Scan Chains	Frequency of Various Relationships in Final Scan Path(s)							Total # Flip-flops
		Beneficial Cases					Non-Beneficial Cases		
		1	2	3	4S	FS	4	0	
dsip	1	0	0	0	1	222	0	1	224
mm4a	1	0	4	0	0	0	6	2	12
mult16b	1	0	1	14	15	0	0	0	30
mult32b	1	0	0	29	30	1	0	2	62
s1488	1	0	1	0	5	0	0	0	6
s1494	1	0	1	0	0	0	5	0	6
s1494	6	0	1	0	5	0	0	0	6
s208.1	1	0	0	1	0	0	6	1	8
s27	1	0	2	0	1	0	0	0	3
s298	1	0	2	0	5	0	1	6	14
s344	1	0	1	0	6	0	3	5	15
s344	6	0	1	0	11	0	2	1	15
s349	1	0	1	0	6	0	3	5	15
s349	6	0	1	0	11	0	2	1	15
s382	1	0	4	0	8	0	3	6	21
s386	1	0	4	0	2	0	0	0	6
s400	1	0	4	0	8	0	3	6	21
s420.1	1	0	0	1	0	0	12	3	16
s444	1	0	4	0	8	0	3	6	21
s510	1	0	0	0	2	0	4	0	6
s526	1	0	2	0	7	0	6	6	21
s641	1	0	2	0	6	6	0	5	19
s713	1	0	2	0	6	6	0	5	19
s820	1	0	1	0	0	0	4	0	5
s820	5	0	1	0	4	0	0	0	5
s832	1	0	1	0	0	0	4	0	5
s832	5	0	1	0	4	0	0	0	5
s838.1	1	0	0	1	0	0	24	7	32

ordered scan paths. The area is obtained after performing placement and routing with the Alliance CAD tools [Greiner 92] and is shown in λ^2. The overhead is calculated as

$$\%\text{Overhead} = \frac{\text{Area}_{\text{Scan}} - \text{Area}_{\text{NoScan}}}{\text{Area}_{\text{NoScan}}} \times 100,$$

where $\text{Area}_{\text{Scan}}$ is the area of the circuit with scan and $\text{Area}_{\text{NoScan}}$ is the area of the circuit without scan.

Table 4. Area of benchmark circuits after routing shown in λ^2

Circuit	No Scan Path	Traditional Scan		Beneficial Scan			
		Single Scan Path		Single Scan Path		Multiple Scan Path	
	Area	Area	%Ov	Area	%Ov	Area	%Ov
dsip	32,717,304	37,004,058	13.1	33,742,188	3.1	33,742,188	3.1
mm4a	2,330,190	4,578,012	96.5	3,970,374	70.4	3,970,374	70.4
mult16b	674,406	1,176,496	74.4	860,700	27.6	860,700	27.6
mult32b	1,549,104	2,766,250	78.6	2,057,130	32.8	2,057,130	32.8
s1488	1,483,506	1,813,320	22.2	1,703,394	14.8	1,703,394	14.8
s1494	1,482,210	1,751,442	18.2	1,553,904	4.8	893,592	-39.7
s208.1	191,142	236,016	23.5	215,424	12.7	215,424	12.7
s27	50,688	65,280	28.8	57,600	13.6	57,600	13.6
s298	353,280	474,606	34.3	494,592	40.0	494,592	40.0
s344	372,528	453,258	21.7	513,648	37.9	491,640	32.0
s349	382,080	449,988	17.8	495,012	29.6	490,032	28.3
s382	441,090	493,968	12.0	581,742	31.9	581,742	31.9
s386	318,240	361,122	13.5	331,266	4.1	331,266	4.1
s400	422,232	509,922	20.8	574,824	36.1	574,824	36.1
s420.1	392,496	576,972	47.0	483,678	23.2	483,678	23.2
s444	409,266	553,350	32.5	543,084	32.7	543,084	32.7
s510	573,120	624,024	8.9	394,896	-31.1	394,896	-31.1
s526	476,160	600,606	26.1	593,712	24.7	593,712	24.7
s641	710,814	784,080	10.3	782,316	10.1	782,316	10.1
s713	649,440	810,156	24.7	758,718	16.8	758,718	16.8
s820	823,584	853,632	3.6	818,928	-0.6	696,528	-15.4
s832	673,764	714,780	6.1	783,006	16.2	705,348	4.7
s838.1	813,960	1,455,696	78.8	1,181,808	45.2	1,181,808	45.2

Table 5 shows the percentage reduction in the overhead for the beneficial scan circuits, both single scan paths and multiple scan paths, compared to the traditional scan circuits. The percent reduction is calculated as

$$\%\text{Reduction} = \frac{\text{Overhead}_{\text{TradScan}} - \text{Overhead}_{\text{BeneScan}}}{\text{Overhead}_{\text{TradScan}}} \times 100,$$

where the overheads are calculated as shown previously. Negative overhead reductions are possible when the beneficial scan overhead is larger than the traditional scan overhead.

The beneficially ordered scan path circuits are significantly smaller than the MD flip-flop scan paths. The average area overhead for beneficially ordered scan paths, after

Table 5. Percent reduction in overhead with beneficial scan compared to traditional scan

Circuit	% Reduction in Overhead	
	Single Path	Multiple Path
dsip	76.1	76.1
mm4a	27.0	27.0
mult16b	62.9	62.9
mult32b	58.3	58.3
s1488	33.3	33.3
s1494	73.4	318.6
s208.1	45.9	45.9
s27	52.6	52.6
s298	-16.5	-16.5
s344	-74.8	-47.5
s349	-66.3	-59.0
s382	-166.0	-166.0
s386	69.6	69.6
s400	-74.0	-74.0
s420.1	50.6	50.6
s444	7.1	7.1
s510	450.1	450.1
s526	5.5	5.5
s641	2.4	2.4
s713	32.0	32.0
s820	115.5	522.8
s832	-166.3	23.0
s838.1	42.7	42.7

placement and routing, is about 22%, compared to about 31% for the circuits with traditional scan paths. These average overheads are both somewhat large due to the high ratio of flip-flops to logic in many of the benchmark circuits. However, the comparison is still interesting since the beneficially ordered scan paths have almost 30% less overhead than the circuits with traditional scan paths.

Traditional scan paths can be ordered based on preliminary layouts to reduce the length of the scan signal interconnect. Beneficially ordered scan paths have short scan interconnect as a result of the ordering since flip-flops that should be placed near each other in layout to produce short functional interconnect also should be placed near each other to produce short scan interconnect for beneficial relationships. The segments of the

beneficial scan path composed of non-beneficial relationships can be reordered based on layout information to further reduce the interconnect overhead. All of this results in low interconnect overhead for the beneficially ordered scan paths.

Five of the beneficial scan circuits, two single scan path circuit (*s510* and *s820*) and three multiple scan path circuits (*s1494*, *s510*, and *s820*), are smaller than the non-scanned versions of the circuits. Since the component area (shown in Table 6) of the beneficial scan circuit and the traditional scan circuit are similar, we conclude that this result is a phenomenon of the place-and-route tool. The non-scanned circuit could achieve the same (or better) area as the beneficially ordered circuit by using the same placement and routing layout, but for a given level of effort the place and route tool gives better results for the beneficially ordered circuit than for the non-scanned circuit. The beneficially ordered circuit is easier to place and route.

Comparison of beneficial scan with other techniques is complicated by the fact that much of the savings from beneficially ordering scan paths comes after the circuit has been placed and routed. Table 6 shows the area of the circuits before placement and routing. Comparing the results in Table 6 to the results in Table 4 shows that much of the savings comes from the reduction in the interconnect. Comparisons based on the literal counts show the same tendency. Since Shannon's expansion theorem is used to factor the logic equations for the beneficial scan circuits, the literal count, and therefore the area before placement and routing, is not always reduced from that of the traditional scan circuits, but the factoring enables the logic to be shared for the beneficial relationships and can result in smaller circuits overall once the circuits are placed and routed.

6 State assignment for beneficial scan

There are two steps involved in synthesizing finite state machines (FSM): state assignment and logic synthesis. The previous sections describe how beneficial scan can be used during logic synthesis to reduce the scan overhead by sharing the functional logic

Table 6. Area of benchmark circuits before routing shown in λ^2

Circuit	No Scan Path	Traditional Scan		Beneficial Scan			
		Single Scan Path		Single Scan Path		Multiple Scan Path	
	Area	Area	%Ovhd	Area	%Ovhd	Area	%Ovhd
dsip	2,352,420	2,634,660	12.0	2,338,812	-0.6	2,338,812	-0.6
mm4a	506,772	521,892	3.0	569,268	12.3	569,268	12.3
mult16b	250,488	288,288	15.1	302,652	20.8	302,652	20.8
mult32b	447,300	525,420	17.5	593,460	32.7	593,460	32.7
s1488	400,680	408,240	1.9	430,416	7.4	430,416	7.4
s1494	397,404	404,964	1.9	405,216	2.0	418,320	5.3
s208.1	86,184	96,264	11.7	91,476	6.1	91,476	6.1
s27	20,916	24,696	18.1	24,444	16.9	24,444	16.9
s298	145,152	162,792	12.2	169,344	16.7	169,344	16.7
s344	146,916	165,816	12.9	187,740	27.8	194,292	32.2
s349	146,916	165,816	12.9	187,740	27.8	194,292	32.2
s382	163,800	190,260	16.2	201,348	22.9	201,348	22.9
s386	117,936	125,496	6.4	134,064	13.7	134,064	13.7
s400	163,800	190,260	16.2	201,348	22.9	201,348	22.9
s420.1	166,320	186,480	12.1	181,692	9.2	181,692	9.2
s444	163,296	189,756	16.2	202,356	23.9	202,356	23.9
s510	181,944	189,504	4.2	204,372	12.3	204,372	12.3
s526	175,392	201,852	15.1	210,420	20.0	210,420	20.0
s641	226,044	249,984	10.6	257,544	13.9	257,544	13.9
s713	226,044	249,984	10.6	256,284	13.4	256,284	13.4
s820	255,276	261,576	2.5	261,072	2.3	232,344	-9.0
s832	214,200	220,500	2.9	229,572	7.2	222,012	3.6
s838.1	333,144	373,464	12.1	367,416	10.3	367,416	10.3

and the test logic; however, state assignment, which is performed before logic synthesis, was not discussed.

State assignment is the process of taking a symbolic FSM description, such as that in Fig. 12a, and assigning binary codes to the symbolic states to obtain a specified FSM description, such as that in Fig. 12b. The state assignment used directly affects the implementation of the final circuit. One state assignment may result in a circuit with little area while another state assignment may lead to a large circuit. The state assignment also affects the relationships between flip-flops since those relationships are really the relationships between bits of the state assignments. The state assignment can

Input	Current State	Next State
0	state0	state0
0	state1	state4
0	state2	state0
0	state3	state4
0	state4	state2
0	state5	state6
0	state6	state3
0	state7	state7
1	state0	state4
1	state1	state4
1	state2	state4
1	state3	state4
1	state4	state6
1	state5	state6
1	state6	state7
1	state7	state7

a)

Input	Current State $b_1\ b_2\ b_3$	Next State $b_1\ b_2\ b_3$
0	0 0 0	0 0 0
0	0 0 1	1 0 0
0	0 1 0	0 0 0
0	0 1 1	1 0 0
0	1 0 0	0 1 0
0	1 0 1	1 1 0
0	1 1 0	0 1 1
0	1 1 1	1 1 1
1	0 0 0	1 0 0
1	0 0 1	1 0 0
1	0 1 0	1 0 0
1	0 1 1	1 0 0
1	1 0 0	1 1 0
1	1 0 1	1 1 0
1	1 1 0	1 1 1
1	1 1 1	1 1 1

b)

Figure 12. FSM description: a) symbolic description; b) fully specified description

be targeted to increase the number of beneficial relationships and thereby reduce the size of the final scannable circuit by allowing more sharing of the functional and test logic.

Take, for example, the four-state FSM *dk15* from the IWLS91 logic synthesis benchmark suite [Yang 91]. A minimum of two state bits, giving four (2^2) possible code words, are needed to implement a four-state FSM. Since all four possible code words are needed to encode four states, there are three distinct state assignments (symmetries reduce the twenty-four possible state assignments down to three distinct state assignments). Out of these three state assignments for *dk15*, one assignment results in a circuit with one case N non-beneficial relationship and one case M beneficial relationship, one results in a circuit with two case M beneficial relationships, and one results in a circuit with one case M beneficial relationship and one cost-free scan relationship. As this simple FSM shows, the state assignment can directly affect the types of relationships between flip-flops in the final scan path.

Modifications can be made to the state assignment algorithm to bias the final state assignment such that there are many beneficial relationships. In this way, the final size of the scannable circuit can be reduced. We have modified the *jedi* state assignment tool

[Lin 89] to take beneficial scan into consideration. *Jedi* uses simulated annealing with various cost functions to assign the state codes. The cost function can be modified to account for the number of beneficial relationships and lead to a solution with a maximal number of beneficial relationships. The new cost function is $cost\ function_{beneficial\ scan}$ and is described below and shown in Fig. 13.

The basic approach to finding the cost for a particular state assignment involves finding the relationships between pairs of current state bits and next state bits. There are $b(b\text{-}1)$ pairs of current state bits and next state bits to examine, where b is the number of bits used to represent the state, since bits in the same position do not need to be considered. For example, in the FSM description in Fig. 12b there are three state bits for each state and six interesting pairs:

1) bit one of the current state and bit two of the next state
2) bit one of the current state and bit three of the next state
3) bit two of the current state and bit one of the next state
4) bit two of the current state and bit three of the next state
5) bit three of the current state and bit one of the next state
6) bit three of the current state and bit two of the next state,.

For each pair, the relationship between the bits is determined, as described in Fig. 14, and the cost for that type of relationship is returned, where beneficial relationships have a lower cost than non-beneficial relationships. The following notation is used: *i*.*state*[*state_bit*], where *i* indicates a specific transition between two states, *state* indicates

```
beneficial_cost_function()
    cost = 0
    for each current state bit position s
        for each next state bit position ns
            if s != ns
                relationship_cost = find_best_relationship_cost(s, ns)
                cost = cost + relationship_cost
    return cost
end beneficial_cost_function
```

Figure 13. Pseudo-code for *beneficial_cost_function()*

```
find_best_relationship_cost(s, ns)
    for each transition i
        if i.next_state[ns] always equals i.current_state[s] or
          i.next_state[ns] always equals inverse of i.current_state[s]
            case1 = true
        if i.current_state[s] = 1 always implies i.next_state[ns] = 1 or
          i.current_state[s] = 0 always implies i.next_state[ns] = 0
            case2 = true
        for each transition j
            if i.input implies j.input
                if i.next_state[ns] always equals i.current_state[s] or
                  i.next_state[ns] always equals inverse of i.current_state[s]
                    freescan = true
                if j.current_state equals i.current_state except in bit position s
                    if i.next_state[ns] always equals i.current_state[s] or
                      i.next_state[ns] always equals inverse of i.current_state[s]
                        other_beneficial_case = true
    if freescan or case1 or case2 or other_beneficial_case is true
        return beneficial_case_cost
    else
        return non_beneficial_case_cost
end find_best_relationship_cost
```

Figure 14. Pseudo-code for *find_best_relationship_cost()*

either the current state or the next state, and *state_bit* indicates the particular bit position of the *state*; *i.input*, which refers to the input values corresponding to transition *i*.

For example, in the FSM in Fig. 12b, the first bit of the current state and the second bit of the next state have a case B relationship since the first bit and the second bit are the same for all sixteen transitions. The criteria for case A and cost-free scan relationships are also met because of the characteristics of the case B relationship, but the case B relationship is a stricter classification.

The first bit of the current state and the third bit of the next state exhibit a case A relationship since whenever the first bit is a ‘0’ the third bit is also a ‘0’. There is not a case B relationship because the fourth transition has the third bit equal to the first bit, and the fifth transition has the third bit equal to the inverse of the first bit. There is not a cost-free scan relationship because transitions exist with the input equal to ‘0’ and equal to ‘1’ where the third bit is both equal to the first bit and equal to the inverse of the first bit.

Table 7. Relationships for FSM description in Fig. 12b

Current State	Next State		
	Bit Position 1	Bit Position 2	Bit Position 3
Bit Position 1	—	Case B	Case A
Bit Position 2	Non-Beneficial	—	Case A
Bit Position 3	Free Scan	Non-Beneficial	—

There is no beneficial relationship between the second bit of the current state and the first bit of the next state. The first transition has the first bit equal to the second, and the second transition has the first bit equal to the inverse of the second, therefore no case B. The second transition has a '0' implying a '1', and the third transition has a '1' implying a '0', therefore no case A. There is no input combination where the transition has the first bit always equal to the second bit or always equal to the inverse of the second bit, therefore no cost-free scan. There is no other type of beneficial scan since there is no pair of transitions with the same input and state bits that meets the criteria. Therefore, there is no beneficial relationship for this pair of bits.

The three other bit pairs can be analyzed in a similar manner. The relationships are summarized in Table 7.

Figure 15 shows a circuit implementing the FSM from Fig. 12b. Analyzing the circuit as described in Sec. 3 shows that the relationships between the flip-flops in the circuit correspond to the relationships determined by the analysis of the FSM description.

Once the relationships are determined for a particular state assignment, the relative cost can be calculated. If the cost function is based solely on the beneficial relationships, without taking into consideration other factors such as output dominance, the number of possible beneficial relationships in the final circuit does increase but so does the circuit

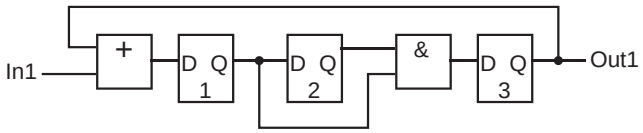

Figure 15. Circuit implementing FSM in Fig. 12b

Table 8. Number of beneficial relationships in final scan path with state assignment based solely on cost of beneficial relationships

Circuit	original *jedi*		modified *jedi*	
	Beneficial Cases	Non-Beneficial Cases	Beneficial Cases	Non-Beneficial Cases
bbara	3	1	4	0
bbsse	2	2	3	1
bbtas	2	1	2	1
cse	1	3	1	3
dk14	1	2	1	2
dk17	2	1	3	0
dk512	3	1	4	0
kirkman	3	2	3	2
lion9	3	1	3	1
mark1	4	0	2	2
mc	1	1	1	1
s208	5	0	5	0
s27	3	0	3	0
s510	1	5	1	5
shiftreg	3	0	3	0
sse	2	2	3	1
styr	1	4	1	4

size. Tables 8 and 9 show the results of modifying *jedi* so that the cost function is derived solely from the beneficial relationships. Table 8 shows the number of beneficial and non-beneficial relationships in the final scannable circuit for both the original jedi and the modified jedi. Table 9 shows the final circuit areas, in λ^2, and the percent difference calculated as

$$\%\text{Difference} = \frac{\text{Area}_{\text{jedi}_{\text{modified}}} - \text{Area}_{\text{jedi}_{\text{original}}}}{\text{Area}_{\text{jedi}_{\text{original}}}} \times 100.$$

As Table 8 shows, the number of beneficial relationships in the final circuit does tend to increase as compared to the circuit generated by the original *jedi* algorithm. The total number of possible beneficial relationships in the circuit (not shown) also tends to increase. However, as Table 9 shows, the area of the scannable circuit obtained with the modified *jedi* also tends to increase as compared to the scannable circuit obtained with

Table 9. Area of benchmark circuits after routing with state assignment based solely on cost of beneficial relationships

Circuit	Beneficial Scan		% Change from Original *jedi*
	Original *jedi* Area	Modified *jedi* Area	
bbara	199,332	233,604	17.2
bbsse	346,680	324,618	-6.4
bbtas	66,240	79,200	19.6
cse	522,918	551,430	5.5
dk14	208,104	276,576	32.9
dk17	167,400	154,944	-7.4
dk512	175,812	226,188	28.7
kirkman	453,264	541,296	19.4
lion9	64,320	97,650	51.8
mark1	227,556	318,432	39.9
mc	77,520	76,230	-1.7
s208	263,760	346,752	31.5
s27	57,660	72,150	25.1
s510	1,019,070	1,580,040	55.0
shiftreg	42,828	59,160	38.1
sse	346,104	333,840	-3.5
styr	1,385,670	1,656,000	19.5

the original *jedi*. In order to minimize area while still increasing the number of beneficial relationships, criteria other than just the beneficial relationships must be included in the cost function. The goal is to have the beneficial scan cost guide the state assignment to a solution with many beneficial relationships without unduly affecting the original cost. If the beneficial scan cost function is added to the original cost function, then the original cost function can be made to dominate while the new cost function still guides the solution toward many beneficial relationships. The new cost function is now

$$\text{cost function}_{\text{new}} = \text{cost function}_{\text{original}} + \text{cost function}_{\text{beneficial scan}},$$

where $\textit{cost function}_{\textit{original}}$ is the original *jedi* cost function and $\textit{cost function}_{\textit{beneficial scan}}$ is the cost function described in Fig. 13. Section 7 discusses results for the modified *jedi* using this new cost function.

7 Implementation and results of state assignment

We have modified the *jedi* cost function as described in Sec. 6. Some of the IWLS91 benchmark FSM circuits were then synthesized using the modified *jedi* and the modified *sis* described previously. *Jedi*'s default output dominant algorithm was used to obtain the results shown here. The input dominant algorithm, coupled dominant algorithm, and modified output dominant algorithm show similar results. Table 10 shows the number of beneficial and non-beneficial relationships in the final scan paths for the circuits for both the original *jedi* and the modified *jedi.* Table 11 shows the area results for the circuits. Area is shown in λ^2. Results are shown for a single scan path to keep the amount of data to a reasonable size. The traditional scan path area shown uses the smallest original circuit, whether that circuit was obtained with the original or the modified *jedi* for state assignment. The last column shows the percent difference between the beneficial scan circuit with the modified *jedi* and the beneficial scan circuit with the original *jedi* calculated as

$$\%\text{Difference} = \frac{\text{Area}_{\text{jedi}_{\text{modified}}} - \text{Area}_{\text{jedi}_{\text{original}}}}{\text{Area}_{\text{jedi}_{\text{original}}}} \times 100 .$$

State assignment with the modified *jedi* results in a smaller scannable circuit than state assignment with the original *jedi* for thirty-three of the forty-four circuits. The average area savings is about 9%. Modifying the state assignment program to take beneficial scan into account can result in smaller scannable circuits, but the savings is not tremendous. Other factors, such as output dominance, have a larger impact on the final circuit size. The number of beneficial relationships is a secondary consideration.

8 Conclusions

We have presented beneficial scan, a synthesis-for-scan technique that orders the scan path during synthesis so as to maximize the sharing of the functional and the test logic and thereby minimize the area and delay overhead due to the scan path. The results of

Table 10. Number of beneficial relationships in final scan path with the modified state assignment

Circuit	original *jedi*		modified *jedi*	
	Beneficial Cases	Non-Beneficial Cases	Beneficial Cases	Non-Beneficial Cases
bbara	3	1	4	0
bbsse	2	2	2	2
bbtas	2	1	2	1
cse	1	3	1	3
dk14	1	2	1	2
dk15	2	0	2	0
dk16	0	5	1	4
dk17	2	1	2	1
dk27	3	0	3	0
dk512	3	1	2	2
ex1	2	3	1	4
ex2	1	4	2	3
ex3	2	2	2	2
ex4	3	1	3	1
ex5	3	1	3	1
ex6	1	2	1	2
ex7	4	0	3	1
keyb	2	3	4	1
kirkman	3	2	3	2
lion	2	0	2	0
mark1	4	0	4	0
mc	1	1	1	1
opus	3	1	3	1
planet	1	5	1	5
planet1	1	5	1	5
s1	3	2	4	1
s1488	1	5	1	5
s1494	1	5	1	5
s1a	3	2	4	1
s208	5	0	4	1
s27	3	0	3	0
s420	5	0	4	1
s510	1	5	1	5
s820	1	4	1	4
s832	1	4	1	4
sand	1	4	1	4
scf	1	6	1	6
shiftreg	3	0	3	0
sse	2	2	2	2
styr	1	4	1	4
tav	1	1	1	1
tbk	1	4	1	4
train4	1	1	1	1

Table 11. Area of benchmark circuits after routing shown in λ^2 with the modified state assignment

Circuit	Traditional Scan	Beneficial Scan		% Diff with Original *jedi*
		Original *jedi*	Modified *jedi*	
	Area	Area	Area	
bbara	148,896	199,332	133,200	-33.2
bbsse	299,832	346,680	270,816	-21.9
bbtas	76,230	66,240	72,270	9.1
cse	476,280	522,918	495,720	-5.2
dk14	218,196	208,104	235,296	13.1
dk15	156,996	186,516	168,198	-9.8
dk16	643,284	692,208	659,940	-4.7
dk17	146,316	167,400	155,064	-7.4
dk27	60,480	73,440	55,680	-24.2
dk512	170,748	175,812	181,692	3.3
ex1	621,600	650,826	730,848	12.3
ex2	421,200	427,812	394,110	-7.9
ex3	147,312	202,176	182,988	-9.5
ex4	201,072	229,320	227,556	-0.8
ex5	174,264	133,380	160,062	20.0
ex6	211,584	226,044	208,080	-7.9
ex7	176,400	233,220	180,336	-22.7
keyb	353,328	518,280	218,994	-57.7
kirkman	440,496	453,264	462,840	2.1
lion	46,956	57,876	52,416	-9.4
mark1	244,416	227,556	218,868	-3.8
mc	72,270	77,520	75,840	-2.2
opus	195,942	228,888	228,888	0.0
planet	1,954,758	1,041,300	690,336	-33.7
planet1	2,113,500	1,059,570	701,520	-33.8
s1	479,682	451,800	226,632	-49.8
s1488	2,020,356	1,524,900	2,123,682	39.3
s1494	953,736	2,332,848	738,018	-68.4
s1a	493,290	607,698	392,964	-35.3
s208	296,208	263,760	283,404	7.4
s27	51,324	57,660	53,508	-7.2
s420	229,896	262,680	248,688	-5.3
s510	967,104	1,019,070	887,502	-12.9
s820	849,420	875,448	862,974	-1.4
s832	729,144	787,896	784,080	-0.5
sand	1,691,580	1,481,220	1,471,500	-0.7
scf	3,807,360	3,632,580	3,547,896	-2.3
shiftreg	44,376	42,828	42,282	-1.3
sse	310,992	346,104	295,320	-14.7
styr	1,535,760	1,385,670	1,404,000	1.3
tav	69,750	72,540	67,500	-6.9
tbk	619,512	611,856	609,552	-0.4
train4	57,600	57,600	53,940	-6.4

this work show that sharing the test logic and interconnect with the functional logic and interconnect can reduce the overhead due to the insertion of a scan path, and that taking the scan path implementation into account during state assignment can result in more sharing of the functional and test logic and further reduce the scan path overhead.

The average area overhead for beneficially ordered scan paths, after placement and routing, is about 22%, compared to about 31% for more traditional scan paths. This area reduction results from two factors: the sharing of the functional and test logic, and the reduction in the interconnect. The final circuit size of finite state machines can be further reduced by taking the beneficial scan relationships into account during the state assignment. Area savings of 9%, on average, can be achieved in this way.

Acknowledgments

This work was supported in part by the Advanced Research Projects Agency under Contract No. DABT63-94-C-0045, and in part by the National Science Foundation under Grant No. MIP-9107760.

References

[Agrawal 88] Agrawal, V., K. Cheng, and D. Johnson, "Designing Circuits with Partial Scan," *IEEE Design and Test* , vol. 5, no. 2, pp. 8-15, April, 1988.

[Avra 93] Avra, L.J., "Synthesizing for Scan Dependence in Built-In Self-Testable Designs," *Proc. Intl. Test Conf.*, Baltimore, MD, pp. 734-743, Oct. 17-21, 1993.

[Avra 94] Avra, L.J., "Synthesis Techniques for Built-In Self-Testable Designs," Center for Reliable Computing Technical Report 94-7, Computer Systems Laboratory, CSL TR 94-633, Stanford University, Stanford, CA, Aug. 1994.

[Bhatia 93] Bhatia, S., and N.K. Jha, "Synthesis of Sequential Circuits for Easy Testability Through Performance-Oriented Parallel Partial Scan," *Intl. Conf. VLSI Design*, India, pp. 151-154, Jan., 1993.

[Bhavsar 86] Bhavsar, D., "A New Economical Implementation for Scannable Flip-Flops in MOS," *IEEE Design and Test*, pp. 52-56, June, 1986.

[Chakradhar 94] Chakradhar, S.T., A. Balakrishnan, and V. Agrawal, "An Exact Algorithm for Selecting Partial Scan Flip-Flops," *Proc. Design Automation Conf.*, San Diego, CA, pp. 81-86, June 6-10, 1994.

[Chickermane 90] Chickermane, V., and J.H. Patel, "An Optimization Based Approach to the Partial Scan Design Problem," *Proc. Intl. Test Conf.*, Washington, DC, pp. 377-386, Sept. 10-14, 1990.

[Cox 94] Cox, H., "On Synthesizing Circuits with Implicit Testability Constraints," *Proc. Intl. Test Conf.*, Washington, DC, pp. 989-998, 0ct. 2-6, 1994.

[Cox 95] Cox, H., "Synthesizing Circuits with Implicit Testability Constraints," *IEEE Design and Test*, vol. 12, no. 2, pp. 16-23, Summer 1995.

[Giles 91] Giles, G.L., and J.R. Wilson, "Toggle-Free Scan Flip-Flop," *United States Patent #5,015,875*, May 1991.

[Greiner 92] Greiner, A., and F. Pecheux, "ALLIANCE: A complete Set of CAD Tools for Teaching VLSI Design," *Proc. Third Eurochip Workshop on VLSI Design Training*, Grenoble, France, pp. 230-237, Sept. 30-Oct. 2, 1992.

[Gupta 90] Gupta, R., and M.A. Breuer, "The BALLAST Methodology for Structured Partial Scan Design," *IEEE Trans. Comp.*, vol. 39, no. 4, pp. 538-544, April, 1990.

[Gupta 91] Gupta, R., and M.A. Breuer, "Ordering Storage Elements in a Single Scan Chain," *Proc. IEEE Intl. Conf. Computer-Aided Design*, Santa Clara, CA, pp. 408-411, Nov. 11-14, 1991.

[Kanjilal 93] Kanjilal, S., S.T. Chakradhar, and V.D. Agrawal, "Synthesis Approach to Design for Testability," *Proc. Intl. Test Conf.*, Baltimore, MD, pp. 754-762, Oct. 17-21, 1993.

[Lee 90] Lee, D.H., and S.M. Reddy, "On Determining Scan Flip-Flops in Partial-Scan Designs," *Proc. IEEE Intl. Conf. Computer-Aided Design*, Santa Clara, CA, pp. 322-325, Nov. 11-15, 1990.

[Lin 89] Lin, B., and A.R. Newton, "Synthesis of Multiple Level Logic from Symbolic High-Level Description Languages," *Proc. IFIP Intl. Conf. VLSI*, Munich, Germany, pp. 187-196, Aug. 16-18, 1989.

[Lin 95] Lin, C.-C., M.T.-C. Lee, M. Marek-Sadowska, and K.-C. Chen, "Cost-Free Scan: A Low-Overhead Scan Path Design Methodology," *Proc. IEEE Intl. Conf. Computer-Aided Design*, San Jose, CA, pp. 528-533, Nov. 5-9, 1995.

[LSI Logic 96] LSI Logic, *G10-p Cell-Based ASIC Products*, Milpitas, CA, 1996.

[McCluskey 86] McCluskey, E.J., *Logic Design Principles*, Prentice-Hall, Englewood Cliffs, NJ, 1986.

[Mukund 91] Mukund, S., S. Thanawastien, and T.R.N. Rao, "Efficient Scan-Path and BIST Latches for Static CMOS ASIC Circuits," *Proceedings of the 33rd Midwest Symposium on Circuits and Systems*, pp. 576-579, Aug., 1990.

[Narayanan 92] Narayanan, S., C. Njinda, and M.A. Breuer, "Optimal Sequencing of Scan Registers," *Proc. Intl. Test Conf.*, Baltimore, MD, pp. 293-302, Sept. 20-24, 1992.

[Narayanan 93] Narayanan, S., R. Gupta, and M.A. Breuer, "Optimal Configuring of Multiple Scan Chains," *IEEE Trans. Comp.*, vol. 42, no. 9, pp. 1121-1131, Sept., 1993.

[Norwood 96] Norwood, R.B., and E.J. McCluskey, "Synthesis-for-Scan and Scan Chain Ordering," *Proc. IEEE VLSI Test Symp.*, Princeton, NJ, pp. 87-92, April 28-May 1, 1996.

[Pradhan 92] Pradhan, D.K., J. Saxena, "A Design for Testability Scheme to Reduce Test Application Time in Full Scan," *Proc. IEEE VLSI Test Symp.*, Atlantic City, NJ, pp. 55-60, April 7-9, 1992.

[Schultz 85] Schultz, D.E., "Static Memory Cell with Dynamic Scan Test Latch," *United States Patent #4,554,664*, Nov. 1985.

[Sentovich 92] Sentovich, E.M., K.J. Singh, C. Moon, H. Savoj, et. al., "Sequential Circuit Design Using Synthesis and Optimization," *Intl. Conf. Comp. Design*, Cambridge, MA, pp. 328-333, Oct. 11-14, 1992.

[Vinnakota 92] Vinnakota, B., and N.K. Jha, "Synthesis of Sequential Circuits for Parallel Scan," *Proc. European Conf. Design Automation*, Brussels, Belgium, pp. 366-370, March 16-19, 1992.

[Yang 91] Yang, S., "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0," Distributed as part of the IWLS91 benchmark distribution.

[Zasio 85] Zasio, J., and L. Cooke, "CMOS Scannable Latch," *United States Patent #4,495,629*, Jan. 1985.