IEEE 5th Workshop on RTL and High Level Testing (WRTLT'04), pp. 13-18, Nov. 2004.

Serial and Parallel TAM Designs for System-on-Chip Interconnects Based on 2-Pattern Testability

Yuusuke Saga, Tomokazu Yoneda and Hideo Fujiwara
Graduate School of Information Science
Nara Institute of Science and Technology
Kansai Science City, 630–0192 Japan
Email: {yuusu-s, yoneda, fujiwara}@is.naist.jp

Abstract— **Testing crosstalk-induced faults on interconnects of system-on-chip (SoC) has become more important because of high integration of semiconductors. The faults can be tested by 2-pattern testing. 2-pattern testing means application of consecutive two test patterns and observation of one test response. In this paper, we present two DFT methods for 2-pattern testability of interconnect. One DFT method utilizes EXTEST mode of IEEE P1500 wrappers and achieves 2-pattern test through a serial TAM. The other method doesn't use IEEE P1500 wrappers, but utilizes existing interconnects as much as possible in order to achieve 2-pattern test. In case studies, we show advantages that hardware overhead of the proposed method is lower than that of our previous DFT method based on consecutive testability.**

I. Introduction

With high integration and high working frequency of system-on-chip (SoC), coupling capacitance and mutual inductance between long interconnects become significant. Therefore, testing crosstalk-induced faults on interconnects becomes an important problem. Maximal aggressor (MA) model is proposed as a model of crosstalk-induced faults on interconnects[1]. On this MA fault model, there are $4N + 2$ faults for an N bit interconnect. Testing of the MA faults on interconnects is performed by application of consecutive two test patterns and observation of the one test response (2-pattern test). Therefore, $4N + 2$ 2-pattern tests are required for an N bit interconnect.

We have proposed DFT methods based on consecutive testability that supports application of consecutive test sequence of arbitrary length and observation of consecutive test response of arbitrary length[2]. However, there is a room for reduction of hardware overhead because interconnects can be tested by 2-pattern test.

In this paper, we present two DFT methods for 2-pattern testability of SoC interconnects. One DFT method is based on IEEE P1500 wrapper, and achieves 2-pattern test by using EXTEST mode of IEEE P1500 wrapper[3] and a serial test access mechanism (TAM) in an SoC. The other DFT method achieves 2-pattern testability without IEEE P1500 wrapper. In this method, a register is added to each port of each core instead of IEEE P1500 wrapper cells. Then, we design TAMs for 2-pattern testing for interconnects from primary inputs to registers, and from registers to primary outputs by using existing interconnects as much as possible. In case studies, we show advantages that one of the proposed DFT methods is effective when designers adopt the IEEE P1500 wrappers, and hardware overhead of the another proposed DFT method is lower than that of our previous DFT method based on consecutive testability.

The rest of this paper is organized as follows. Section **II** describes target SoCs, IEEE P1500 wrappers and testing of interconnects. Section **III** defines 2-pattern testability of interconnects. In section **IV**, we present two DFT methods for the 2-pattern testability. Results of case studies are presented in section **V**. Finally, section **VI** concludes this paper.

II. Preliminaries

A. SoC Modeling

We assume that an SoC consists of cores, primary inputs, primary outputs and interconnects and all cores operate using single clock frequency.

We introduce ports of each core as interface points in a natural fashion: signals enter into a core through its input ports, and exit through its output ports. An interconnect connects an output port with an input port, a primary input with an input port, or an output port with a primary output. Though any number of interconnects can connect to the same output port (i.e. fanout is allowed), only one interconnect can connect to the same input port. It is not necessary that interconnects are of the same bit-width.

A floor plan is provided for an SoC and each core has placement denoted by (x, y) coordinates of its center of gravity. The area of a signal line is estimated as the product of bit-width and length on the floor plan, where the length of each line is defined as Manhattan distance.

B. IEEE P1500 Wrapper

The IEEE P1500 wrappers is a shell around a core, and it allows the core to be tested as a stand-alone module by separating it from its environment[4]. Fig. 1 shows an example of a core with IEEE P1500 wrapper[3]. It has functional input/output ports. Furthermore, it has a mandatory one-bit input/output port pair, wrapper serial input/output, respectively. Optionally, it has multi-bit input/output port pair, wrapper parallel input/output, respectively. The IEEE P1500 wrapper also contains wrapper input/output cells (Fig. 2) in order to provide controllability and observability of functional input/outputs. The wrapper has four main types of modes: (1) functional mode, (2) INTEST mode, (3) EXTEST mode and (4) BYPASS mode. Here, we describe the detail of EXTEST mode using wrapper serial input/output since this is a mandatory test mode for interconnects between cores. First, a

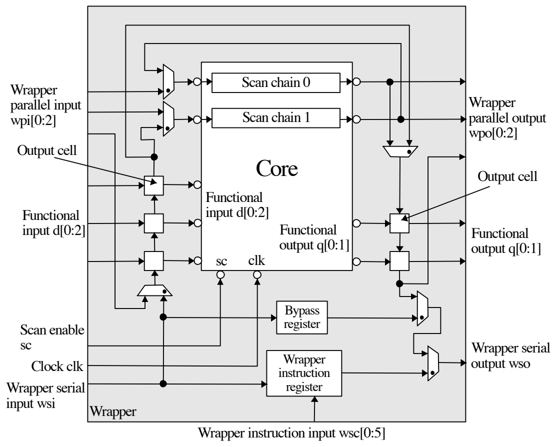

Fig. 1. IEEE P1500 Wrapper

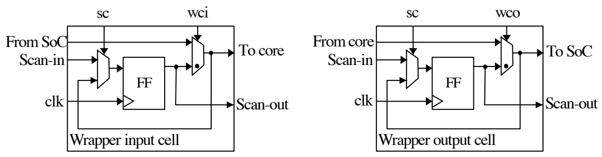

Fig. 2. Wrapper Input Cell and Wrapper Output Cell

test pattern for an interconnect is scanned into wrapper output cells through a wrapper serial input. Then, the test pattern is applied to the interconnect, and the test response from the interconnect are captured in wrapper input cells of next cores. After that, the captured test response is scanned out through a wrapper serial output. In this way, IEEE P1500 wrappers can support 1-pattern test for interconnects. However, 2-pattern testing required for crosstalk-induced faults or delay faults is impossible.

C. Testing of Interconnects

For deep sub-micron technology, coupling capacitance and mutual inductance between interconnects become significant. Cross-coupling effects such as glitches, or delay/speed-up transitions cause signal integrity problems on interconnects. Cuviello et al.[1] proposed the maximal aggressor (MA) fault model which abstracts the crosstalk-induced defects by linear number of faults (detailed models of speed-up transitions are in [5]). In this fault model, one line of an N bit interconnect is a victim line and other $N-1$ lines act as aggressors. Effect of all aggressors appears on the victim line. In MA

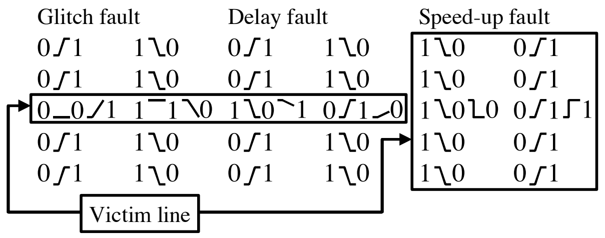

Fig. 3. Test Patterns for MA Fault Model

fault model, there exist $4N+2$ 2-pattern tests for an N bit interconnect. The required transitions on the aggressor/victim lines to test the MA faults are shown in Fig. 3. Each line becomes a victim line that is indicated by arrows in Fig. 3 for testing of positive glitch, negative glitch, falling delay and rising delay. For testing of falling and rising speed-up faults, position of the victim line is not identified. Therefore, there exist two 2-pattern tests for rising/falling speed-up.

III. 2-PATTERN TESTABILITY

In this section, we define 2-pattern testability which can achieve 2-pattern test for interconnects. In **III-A**, we define a port graph for an SoC. We define 2-pattern testability on the port graph when there exist IEEE P1500 wrappers for all cores in an SoC in **III-B**, and we define 2-pattern testability on the port graph when there exist no IEEE P1500 wrappers for all cores in **III-C**.

A. Port Graph

We define a port graph $G=(V,E)$ for an SoC as the following directed graph (Fig. 4).

- $V = V_{PI} \cup V_{PO} \cup V_{port}$ where V_{PI} is the set of all primary inputs of the SoC, V_{PO} is the set of all primary outputs of the SoC, and V_{port} is the set of all ports of cores in the SoC.
- E where $E = \{(x,y) \in V \times V|$ port x is connected to port y $\}$.

We refer to a vertex that has no input edge as a source, and a vertex that has no output edge as a sink. We consider AND and OR as types for edges in a port graph. AND is a type for edges that propagate part of bit-width of v (Fig. 5(a)(c)). OR is a type for edges that propagate all bit-width of v (Fig. 5(b)(d)). Fig. 5 illustrates AND/OR edges and their hardware implementations. For an edge $i \in E$, we define v_i^{in} as a head of i and v_i^{out} as a tail of i. For $v \in V$ in a port graph, we define w_v^{in} as follows.

1) If the input edges of v are AND type, w_v^{in} is the sum of the bit-width of the input edges of v.
2) If the input edges of v are OR type, w_v^{in} is the bit-width of one of the input edge of v.

Similarly, we define w_v^{out} as follows.

1) If the output edges of v are AND type, w_v^{out} is the sum of the bit-width of the output edges of v.
2) If the output edges of v are OR type, w_v^{out} is bit-width of one of the output edge of v.

Let u and v be vertices in G. Reconvergence from u to v are called OR-AND reconvergence if all the following conditions are satisfied.

1) Type of the output edges of u in the reconvergence is OR.
2) Type of the input edges of v in the reconvergence is AND.

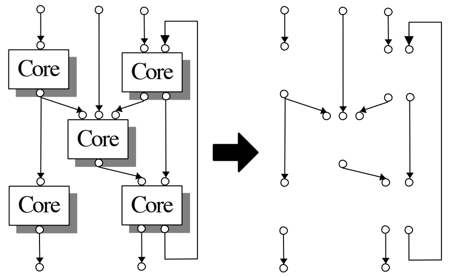

Fig. 4. Port Graph

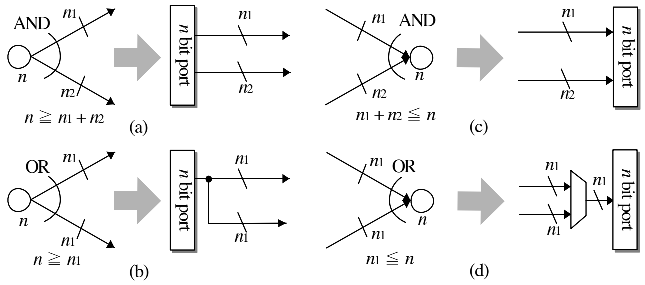

Fig. 5. AND/OR Edges and Their Hardware Implementations

B. 2-pattern Testability with IEEE P1500 Wrappers

When all cores in an SoC have IEEE P1500 wrappers, all input/output ports of all cores are controllable and observable. Thus, a head and a tail of an interconnect are 1-pattern controllable and 1-pattern observable, respectively. Then, we define 2-pattern testability of an interconnect in the case that all cores in an SoC have IEEE P1500 wrappers as follows.

Definition 1: (2-pattern testability of an interconnect with IEEE P1500 wrappers) Let G be a port graph for an SoC, and let i be an interconnect in E, then i is 2-pattern testable if there exists a subgraph G_i that satisfies all the following conditions.

1) All sources in G_i are included in $V_{PI} \cup V_{port} - \{v_i^{in}\}$.
2) Only v_i^{in} is a sink in G_i.
3) $w_{v_i^{in}}^{in} \geq w_i$. Here, let w_i be bit-width of i, □

C. 2-pattern Testability without IEEE P1500 Wrappers

When there exists no IEEE wrapper for all cores in an SoC, we have to propagate the test sequence (two patterns) from primary inputs to an interconnect, and furthermore propagate the test response to primary outputs. In this case, we define 2-pattern controllability, 1-pattern observability and 2-pattern testability of an interconnect as follows.

Definition 2: (2-pattern controllability of an interconnect without IEEE P1500 wrappers) Let G be a port graph for an SoC, and let i be an interconnect in E, then i is 2-pattern controllable if there exists a subgraph G_i^J that satisfies all the following conditions.

1) All sources in G_i^J are primary inputs.
2) Only v_i^{in} is a sink in G_i^J.
3) For all vertices v except for sources in G_i^J, $w_v^{in} \geq w_v^{out}$.
4) Let R_i^J be the set of OR-AND reconvergence in G_i^J. For each OR-AND reconvergence $r \in R_i^J$ from u to v, r satisfies all the following conditions.
   a) Only u has OR type edges.
   b) Let d_p be the number of vertices on a path. Then, for any pair of paths, p_j, p_k, from u to v in r that do not share the same output edge of u, it satisfies $|d_{p_j} - d_{p_k}| \geq 2$. □

Definition 3: (1-pattern observability of an interconnect without IEEE P1500 wrappers) Let G be a port graph for an SoC, and let i be an interconnect in E, then i is 1-pattern observable if there exists a subgraph G_i^P that satisfies all the following conditions.

1) Only v_i^{out} is a source in G_i^P.
2) All sinks in G_i^P are primary outputs.
3) For all vertices v except for sinks of G_i^P, $w_v^{in} \leq w_v^{out}$. □

Definition 4: (2-pattern testability of an interconnect without IEEE P1500 wrappers) An interconnect $i \in E$ is said to be 2-pattern testable if i satisfies all the following conditions.

1) i is 2-pattern controllable.
2) i is 1-pattern observable. □

IV. Design for Testability

In this section, we present two DFT methods that make all interconnects 2-pattern testable. One DFT method called *serial 2-pattern DFT* adds IEEE P1500 wrappers to all cores in an SoC and utilizes a serial TAM to make interconnects 2-pattern testable. The other DFT method called *parallel 2-pattern DFT* does not add IEEE P1500 wrapper, but utilizes existing interconnects as much as possible in order to design TAMs which make all interconnects 2-pattern testable.

A. Serial 2-pattern DFT

We describe a serial 2-pattern DFT method that adds IEEE P1500 wrappers to all cores in an SoC and utilizes serial TAMs to make all interconnects 2-pattern testable. In EXTEST mode of IEEE P1500 wrappers, only 1-pattern test can be performed for interconnects. Then, this method aims to achieve 2-pattern testability by modifying IEEE P1500 wrappers so that two test patterns for an interconnect can be stored in wrapper input/output cells in a core and the patterns can be applied to the interconnect consecutively. We formulate the above design for 2-pattern testability as the following optimization problem.

Definition 5: (Serial 2-pattern DFT problem)

- Input: An SoC
- Output: An SoC with a serial TAM and wrappers that make all interconnects 2-pattern testable
- Optimization: Minimizing hardware overhead (numbers of registers and MUXs, and area of additional lines) □

Serial 2-pattern DFT algorithm consists of the following three steps.

Step 1: (Wrapper and TAM design for each core)

We design IEEE P1500 wrapper with serial EXTEST mode that is mandatory mode in P1500. Moreover, we design a serial TAM that connects all cores in series to realize serial EXTEST mode.

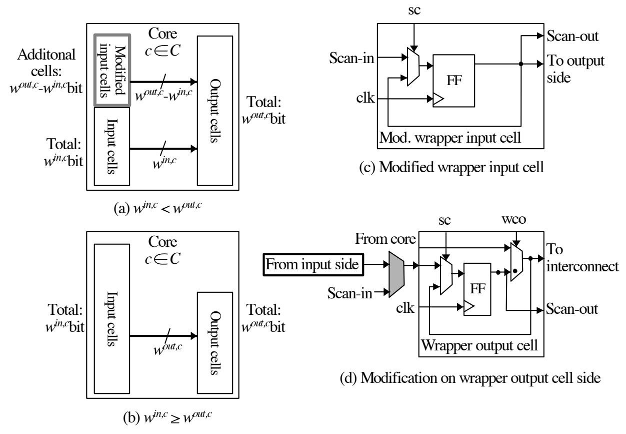

Fig. 6. Serial 2-pattern DFT

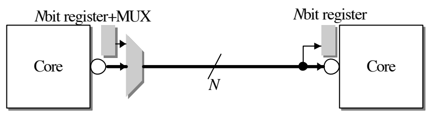

Fig. 7. Additional Hardware for Interconnects

Step 2: (Addition of modified wrapper input cells)

For each core $c \in C$ in an SoC, let $w^{in,c}$ be total bit-width of input ports of c, and $w^{out,c}$ be total bit-width of output ports of c. Then, if $w^{in,c} < w^{out,c}$ we put $w^{out,c} - w^{in,c}$ bit modified wrapper input cells (Fig. 6(c)) to input side of c (Fig. 6(a)). Otherwise, we do not add modified wrapper input cells (Fig. 6(b)).

Step 3: (Addition of paths in wrappers)

We create $w^{out,c}$ bit parallel paths from input cells to wrapper output cells by adding signal lines and MUXs (Fig. 6(a)(b)). MUXs are inserted to scan-in ports of wrapper output cells (Fig. 6(d)).

By using the wrappers designed by the above three steps, the first pattern of 2-pattern test can be scanned into wrapper output cells of the core, and second pattern can be scanned into wrapper input cells of the core. Moreover, by using paths from wrapper input cells to wrapper output cells, these two patterns can be applied to interconnects consecutively.

B. Parallel 2-pattern DFT

We describe a parallel 2-pattern DFT method for 2-pattern testability of interconnects without using IEEE P1500 wrappers. This method adds registers to input/output ports of each core instead of IEEE P1500 wrapper cells (Fig. 7). The location of the added registers are very close to the input/output ports. Then, for each interconnect i, we create paths from primary inputs to the register located at the head of i, and paths from the register located at the tail of i to primary outputs. If there exists an interconnect that cannot be made 2-pattern testable by utilizing existing paths, we add signal lines and MUXs in order to achieve 2-pattern testability. We formulate the above design for 2-pattern testability as the following optimization problem.

Definition 6: (Parallel 2-pattern DFT problem)

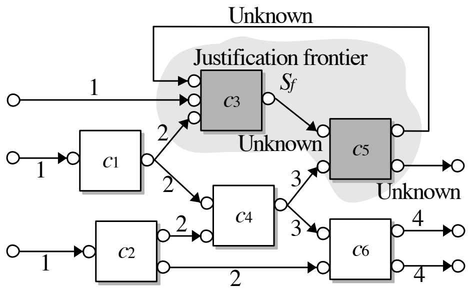

Fig. 8. Determination of Justification Level, and Justification Frontier

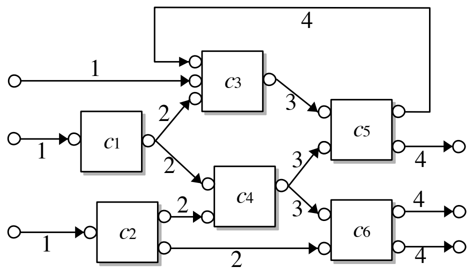

Fig. 9. An SoC with All Interconnects whose Justification Levels are Determined

- Input: An SoC
- Output: An SoC with a parallel TAM that make all interconnects 2-pattern testable
- Optimization: Minimizing hardware overhead (numbers of registers and MUXs, and area of additional lines) □

We describe a heuristic algorithm for the parallel 2-pattern DFT. This algorithm consists of the following four steps.

Step 1: Determine justification level of interconnects
Step 2: Design for 2-pattern controllability of interconnects
Step 3: Determine observation level of interconnects
Step 4: Design for 1-pattern observability of interconnects

In step 1, we determine justification level of each interconnect that is the distance from primary inputs. Step 2 makes interconnects 2-pattern controllable in the ascending order of justification levels. Step 3 determines observation level of each interconnect that is the distance from primary outputs. In step 4, we make interconnects 1-pattern observable in the ascending order of observation levels. We describe the detail of each step as follows.

Step 1: (Determine justification levels of interconnects)

In this step, we determine justification levels of all interconnects. Justification level is a distance from primary inputs. We determine justification levels of all interconnects by the following three processes.

(1) We set justification levels of interconnects directly connected to primary input to 1, and set justification levels of other interconnects to unknown. For each core c, we refer to an interconnect whose tail is an input port of c as an input interconnect, and an interconnect whose head is an output port of c output interconnects. Let S_f be the set of cores that satisfy the following two condition: (i) each core in S_f has one or more input interconnects whose justification levels are

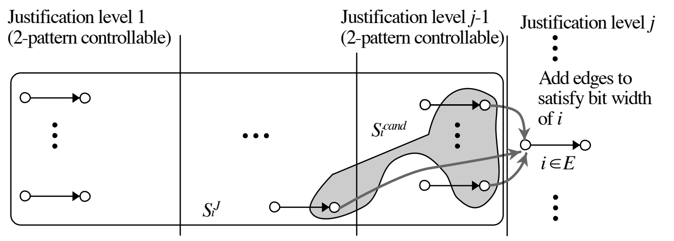

Fig. 10. Making 2-pattern Controllable in Ascending Order of Justification Level

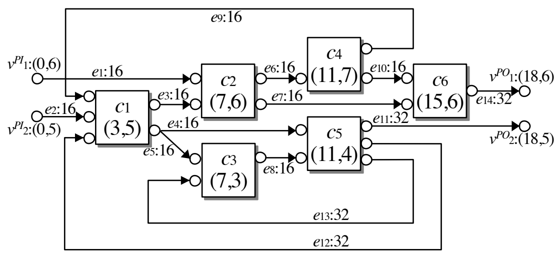

Fig. 11. SoC Example of Case Studies

determined and (ii) each core in S_f has all output interconnects whose justification levels are unknown (Fig. 8). We call S_f *justification frontier*. We define that each core $c \in S_f$ has *core-level* L_c and flag. Core-level is maximum value among known justification levels of input interconnects of the core. The flag has either "visited" or "unvisited", and we initialize all the flags to "unvisited" in this process (1).
(2) We select a core c with "unvisited" flag and minimum core-level from S_f. We set justification levels of output interconnects of c to $L_c + 1$ if all justification levels of input interconnects are not unknown. If the justification levels of output interconnects of c are determined, then we update S_f. Otherwise, we set the flag of c to "visited". We iterate this process (2) until justification levels of all interconnects are determined (S_f becomes empty), or flags of all elements in S_f are "visited" (loops are detected). If S_f becomes empty, the justification levels of all interconnects are determined and we move to step 2. If flags of all elements in S_f are "visited", we go to process (3).
(3) When flags of all cores in S_f are "visited" (i.e., loops are detected), we select a core which have minimum core-level from S_f and set justification levels of output interconnects to $L_c + 1$. Moreover, we reset flags of all elements in S_f to "unvisited", and we go back to process (2).

Fig.9 shows justification levels corresponding to the SoC in Fig. 8 after this step.

Step 2: (Design for 2-pattern controllability of interconnects)

We make all interconnects 2-pattern controllable by using justification levels by the following five processes.
(1) We add a register to each input/output port of each core. Bit-width of the register is the same as that of the port, and the location of the register is very close to the port (Fig. 7).
(2) We select an interconnect i such that i is not 2-pattern controllable and i has minimum justification level. Here, we consider that interconnects with justification level 1 are 2-pattern controllable from the beginning since they are directly connected to primary inputs. On the other hand, interconnects without justification level 1 may not be 2-pattern controllable. If there does not exist such i, all interconnects are 2-pattern controllable and we move to step 3. Let S_i^J be a set of 2-pattern controllable vertices for i which are head and tail vertices of all 2-pattern controllable interconnects. Then, we calculate area costs of signal lines from v_i^{in} to each element in S_i^J. Area cost of a signal line is product of Manhattan distance of the added signal lines and bit-width of the signal lines. Let S_i^{cand} be the set of vertices that are selected as candidate vertices to make i 2-pattern controllable. Let W_i be the bit-width that is currently required to make i 2-pattern controllable. We initialize $W_i = w_i$. Here, let w_i be bit-width of i.
(3) We pick a port v with minimum area cost from S_i^J, and add the port v to S_i^{cand} (Fig. 10). Then, we update S_i^J as $S_i^J = S_i^J - \{v\}$. We iterate this process (3) until sum of bit-width of all elements in S_i^{cand} is W_i or over.
(4) We check 2-pattern controllability (definition 2) of i in the case that there exist AND type edges from all elements in S_i^{cand} to v_i^{in}. If i satisfies 2-pattern controllability, we can make i 2-pattern controllable by adding paths from each element in S_i^{cand} to v_i^{in} with MUXs and signal lines. Then, we go back to process (2). Otherwise, it is impossible to make i 2-pattern controllable by using paths from all elements in S_i^{cand} to v_i^{in} simultaneously. In that case, we go to process (5).
(5) Let S_i^n be the subset of S_i^{cand} whose vertices cannot be utilized simultaneously in order to satisfy 2-pattern controllability of i. Let v_k be a vertex which have maximum bit-width in S_i^n. Then, we add paths from the elements in $\left(S_i^{cand} - S_i^n\right) \cup \{v_k\}$ to v_i^{in} by using MUXs and signal lines. Moreover, let w_c be the sum of bit-width of all elements in $\left(S_i^{cand} - S_i^n\right) \cup \{v_k\}$, then we update W as $W = W - w_c$. We update S_i^{cand} as $S_i^{cand} = \phi$, then we go back to process (3).

Step 3: (Determine observation levels of interconnects)

This step determines observation level of all interconnects. Observation level is the a distance from primary outputs. We set observation levels of interconnects directly connected to primary output to 1, then we process this step in a similar fashion to step 1.

Step 4: (Design for 1-pattern observability of interconnects)

We make all interconnects 1-pattern observable by using observation levels in a similar fashion to step 2.

V. Case Studies

In this section, we present results of case studies obtained by the proposed methods. We apply serial 2-pattern DFT method, parallel 2-pattern DFT method and consecutive DFT method to two SoCs we randomly created since ITC'02 SoC benchmarks[6] have no information about connectivity between cores. SoC1 shown in Fig. 11 has 6 cores and 14 interconnects. SoC2 has 20 cores and 37 interconnects. The consecutive DFT methods uses the same algorithm as the parallel 2-pattern DFT method except for the conditions of

TABLE I
CASE STUDIES RESULT

	SoC	SoC1 (6 Cores, 14 interconnects)				SoC2 (20 Cores, 37 interconnects)			
		2-pattern				2-pattern			
		Serial				Serial			
	DFT	P1500	Add.	Parallel	Consecutive	P1500	Add.	Parallel	Consecutive
Hardware overhead	Registers (bit)	464	80	464	464	1184	128	1184	1184
	MUXs (bit)	928	317	352	384	2368	692	912	944
	Core external lines (bit*length)	27	0	48	288	127	0	256	688
	Core internal lines (bit)	0	237	320	272	0	564	720	688
Test application time (clock)		71524		6064	2452	172132		7787	6493

testability in step 2 and 4. The DFT method uses consecutive testability defined in [2] instead of 2-pattern testability.

Table I shows hardware overhead and test application time obtained by the three DFT methods. In table I, "Serial" and "Parallel" in "2-pattern" denote results of serial 2-pattern DFT method and parallel 2-pattern DFT method, respectively. "Consecutive" denotes results of consecutive DFT method. "P1500" in "Serial" denotes hardware overhead of original IEEE P1500 wrappers, and "Add." in "Serial" denotes hardware overhead for 2-pattern testability except for original wrapper hardware. "Registers" and "MUXs" in "Hardware overhead" denote the bit-width of registers added by each method. "Core external lines" denotes area costs of added signal lines between cores, and "Core internal lines" shows the bit-width of added signal lines to the inside of cores since the distance from an input port of a core to an output port of the core is 0. "Test application time" is a time in the case that we execute MA test. In serial 2-pattern DFT method, all interconnects can be tested simultaneously because each interconnect has unique wrapper input/output cells for 2-pattern test. In parallel 2-pattern and consecutive DFT methods, we assume that each interconnect is tested one by one.

Test application time of serial 2-pattern DFT method is longer than those of other methods. Moreover, hardware overhead of registers and MUXs of serial 2-pattern DFT method are very larger than those of other methods. However, serial 2-pattern DFT method is compliant for IEEE P1500 standard. Therefore, this method can achieve the smallest hardware when designers adopt the IEEE P1500 wrappers.

Test application time of parallel 2-pattern DFT method is about 1.9 times longer than that of consecutive DFT method on average. However, parallel 2-pattern DFT method can achieve about 73% reduction of hardware overhead in comparison with consecutive DFT method on average. The reason is difference between 2-pattern testability and consecutive testability. The 2-pattern testability allows common parts of control paths and observation paths, and OR-AND reconvergent paths. You can see that parallel 2-pattern DFT method can achieve reduction of hardware overhead effectively with little increase of test application time compared to consecutive DFT method by this difference.

VI. CONCLUSION

In this paper, we introduced 2-pattern testability of SoC interconnects that is a property to achieve testing of crosstalk-induced faults on SoC interconnects. Moreover, we proposed two DFT methods for SoC interconnects based on the 2-pattern testability. One of the proposed DFT methods is serial 2-pattern DFT which utilizes IEEE P1500 wrappers. This is compliant for IEEE P1500 standard. Therefore, this method is effective when designers adopt the IEEE P1500 wrappers. Another proposed method is parallel 2-pattern DFT which does not utilize IEEE P1500 wrappers, but utilizes existing interconnects as much as possible. In the case studies, we show that test application time of parallel 2-pattern DFT method is about 1.9 times longer than that of consecutive DFT method. However, 2-pattern DFT method can achieve 73% reduction of hardware overhead compared to consecutive DFT method. Parallel 2-pattern DFT method can achieve reduction of hardware overhead effectively with little increase of test application time compared to consecutive DFT method.

One of our future works is to propose a DFT method in the case that test pattern sources and test response sinks are embedded to the inside of SoCs. Another future work is to achieve co-optimization between test application time and area overhead for the proposed 2-pattern testability.

ACKNOWLEDGMENT

This work was supported in part by 21st Century COE (Center of Excellence) Program and in part by JSPS (Japan Society for the Promotion of Science) under Grants-in-Aid for Scientific Research B(2) (No.15300018). Authors would like to thank Prof. Michiko Inoue and Prof. Satoshi Ohtake and members of Computer Design and Test Lab. (Nara Institute of Science and Technology) for their valuable discussions.

REFERENCES

[1] M. Cuviello, S. Dey, X. Bai and Y. Zhao, "Fault Modeling and Simulation for Crosstalk in System-on-Chip Interconnects," in Proc. of Int. Conf. on Computer-Aided Design (ICCAD), pp.297-303, 1999.

[2] T. Yoneda and H. Fujiwara, "Design for Consecutive Testability of System-on-a-Chip with Built-In Self Testable Cores," Journal of Electronic Testing: Theory and Applications (JETTA), vol.18, no.4/5, pp.487-501, Aug. 2002.

[3] IEEE P1500 Standard for Embedded Core Test (SECT), http://grouper.ieee.org/groups/1500/.

[4] Y. Zorian E. J. Marinissen and Sujit Dey, "Testing Embedded-Core Based System Chips," in Proc. of Int. Test Conf. (ITC), pp.130-143, 1998.

[5] W.-C. Lai, J.-R. Huang and K.-T. (T.) Cheng, "Embedded-Software-Based Approach to Testing Crosstalk-Induced Faults at On-Chip Buses," in Proc. of VTS, pp.204-209, 2001.

[6] E. J. Marinissen, V. Iyengar and K. Chakrabarty, "ITC'02 SOC Test Benchmarks Web Site", http://www.extra.research.philips.com/itc02socbenchm/.