IEICE TRANS. INF. & SYST., VOL.E91–D, NO.3 MARCH 2008

PAPER Special Section on Test and Verification of VLSIs

A Self-Test of Dynamically Reconfigurable Processors with Test Frames

Tomoo INOUE[†a)], ***Member***, **Takashi FUJII**[†*], ***Nonmember***, ***and*** **Hideyuki ICHIHARA**[†b)], ***Member***

SUMMARY This paper proposes a self-test method of coarse grain dynamically reconfigurable processors (DRPs) without hardware overhead. In the method, processor elements (PEs) compose a test frame, which consists of test pattern generators (TPGs), processor elements under test (PEUTs) and response analyzers (RAs), while testing themselves one another by changing test frames appropriately. We design several test frames with different structures, and discuss the relationship of the structures to the numbers of contexts and test frames for testing all the functions of PEs. A case study shows that there exists an optimal test frame which minimizes the test application time under a constraint.

key words: *dynamically reconfigurable processors, self-test, optimal contexts, test application time, test frames*

1. Introduction

The reduction in the cost and time for development of LSIs has been a significant issue. Reconfigurable devices, which can implement digital systems required by users in the field, have high potential for solving such a issue. Field-programmable gate arrays (FPGAs) are typical reconfigurable devices. FPGAs have grown in performance and functionality since they were developed more than ten years ago, and accordingly they have been used for various application instead of ASICs, not just for prototyping.

Dynamically reconfigurable processors (DRPs) have been recently developed [1]–[3]. DRPs have several properties which are different from those of fine grain devices such as FPGAs. One is that a DRP is in a coarse grain fashion – it consists of an array of processor elements (PEs), and the function of PEs and the interconnects surrounding the PEs are configured by contexts (or configuration data). Another is its reconfigurability, i.e., a DRP can switch its functions dynamically during its execution by loading another context with a single (or a few) clock cycle. These properties suit applications such as mobile systems, digital audio visual systems, and network controllers [4], [5], requiring flexibility and performance as well as low power.

Testing of dynamically reconfigurable processors, as well as conventional digital LSIs, is important, but considerably complex. Like testing of FPGAs [6]–[9], a DRP can be tested by repeating configurations of the system and applications of test data to the configured system. The time required for configuration of an FPGA is much larger than the cycle time of the configured system, and hence many literatures focused on reducing the number of configurations for reduction in the total test application time. Unlike FPGAs, however, a DRP can reconfigure itself dynamically at operational speed, and therefore, in order to perform test application for DRPs efficiently, an appropriate combination of the set of contexts and test data to be applied to the configured systems must be required.

Katoh and Ito [10] proposed a built-in self-test (BIST) design for processor elements in a DRP. In this method, they modify registers in a PE so that the registers can be linear-feedback shift registers (LFSRs) for test pattern generation and multiple-input signature registers (MISRs) with extra hardware overhead, and achieve small test application time.

In this paper, we propose a self-test method for testing PEs in DRPs without hardware overhead. We introduce a test frame, which is a fundamental structure of test configurations. A test frame consists of a test pattern generator (TPG), PEs under test (PEUTs) and response analyzers (RAs), which are all composed of PEs themselves. The pair of TPG and RAs tests multiple PEUTs simultaneously, and by switching several test frames, all the PEs in a DRP are tested. We discuss the relationship of the test frames to the number of contexts required for test and the test application time while exploring several designs of test frames, and show that our self-test method can achieve test application for DRPs without hardware overhead. Furthermore, we show that there exists an optimal test frame which minimizes the test application time for a given constraint.

2. Model of Dynamically Reconfigurable Processors

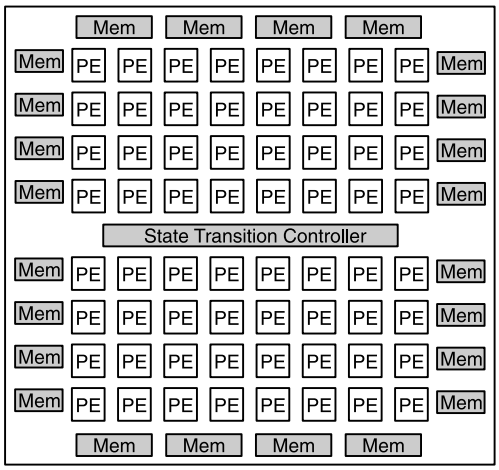
The model of dynamically reconfigurable processors considered in this paper is made by referring to a DRP-1 [1], [3], which is a commercial coarse grain device. In the following discussion, we refer to our model of dynamically reconfigurable processors as a DRP for short.

A DRP consists of an array of PEs (Processing Elements), as shown in Fig. 1 (a), whose functions and interconnection can be reconfigured dynamically, i.e., while processing data on each PE, the DRP can change all the functions implemented on the PEs and the network among the PEs at a single clock cycle. Figure 1 (b) shows the model of a PE. A PE has two 8-bit data inputs, one 8-bit data output and a flag input as well as a flag output, and includes an ALU for

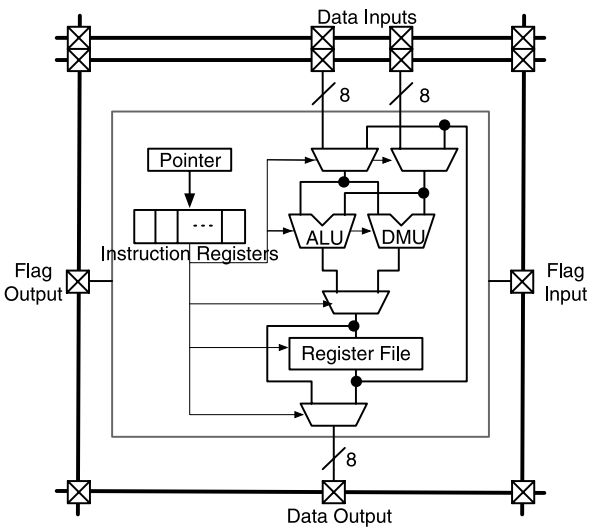
Manuscript received April 9, 2007.
Manuscript revised August 10, 2007.

[†]The authors are with the Graduate School of Information Sciences, Hiroshima City University, Hiroshima-shi, 731–3194 Japan.
[*]Presently, with the Canon Inc.
a) E-mail: tomoo@hiroshima-cu.ac.jp
b) E-mail: ichihara@hiroshima-cu.ac.jp

DOI: 10.1093/ietisy/e91–d.3.756



(a) Array of PEs.



(b) PE with interconnect.

Fig. 1 Architecture of DRP (dynamic reconfigurable processor).

arithmetical and logical calculation, a DMU (Data Manipulation Unit) for bit operations such as shifts and masks, and a register file. Each PE also has a set of instruction registers. An instruction register can store a configuration data of its function as well as interconnection of surrounding buses.

All the PEs are controlled by the state transition controller (STC). The STC broadcasts a context number to all the PEs simultaneously. The context number is used as a pointer to the instruction register. Each PE is configured according to the content of the register pointed by the context number, and its configuration as well as the connection to other PEs are changed at every clock cycle.

In the following discussion, concentrating the configuration of PEs for testing them, we do not make concrete assumption about the number of interconnects surrounding PEs and their switches. Also, we assume that the size of register file is not restricted, and the time required to execute any operation in a PE is a constant unit delay independent of the operation.

3. Self-Test of DRPs

An effective method for testing complex devices like reconfigurable processors is a divide-and-conquer approach – by testing all the components of the device severally, we can eventually achieve the test of the whole of the device. In this paper, we discuss the testing of PEs in a dynamically reconfigurable processors. One important approach to reconfigurable devices is self-testing. Stroud *et. al.* proposed a method for self-test of FPGAs without hardware overhead [7]. Moreover, since a DRP consists of processor arrays, and hence functional test such as software-based self-test for processors [11], [12] can be also applied to testing for DRPs. Thus, we apply these concepts to the test of dynamically reconfigurable processors.

3.1 Test Frame

In order to test DRPs effectively, we introduce a *test frame* as a fundamental structure of test configurations. A test frame consists of three components: a test pattern generator (TPG), a set of response analyzers (RAs) and a set of PEs under test (PEUTs).

The *TPG*, configured by several PEs, generates pseudo-random test patterns for PEUTs. There is just one TPG in a test frame, and the TPG is shared by all the PEUTs, i.e., the patterns generated by the TPG are distributed into all the PEUTs in parallel[†].

A certain number of *PEUTs* compose a block in which all the PEs configure the same function. Note that, since all the PEUTs are fed with the same patterns by the TPG, all the output responses from the PEUTs are identical unless there exists a *faulty* PEUT.

An *RA* is also composed of several PEs. It receives the output responses from a PEUT block, and detects a faulty PE in the block by comparing the output responses – a faulty PE will have an erroneous output which is different from those of the other (fault free) PEs in the same block. When detecting some error in an output response from a block, the RA stores the error into its signature register. By checking the content of the signature register after finishing the test application with a test frame, we can identify the existence of faulty PEs. Note that every PEUT block requires one RA, and the number of PEs required to compose the corresponding RA increases in proportion to the number of PEUTs in a block.

Figure 2 shows an example of a test frame. In this test frame, the TPG, PEUTs, RAs are composed of 11, 24 and 28 PEs, respectively. This test frame F_1 has 6 PEUT blocks, and each block includes 4 PEs. The data outputs are observed by a single RA, while the flag outputs are observed

[†]In general, different functions require different complete test sets. We consider, however, that the TPG generates test patterns common among the functions to be tested. As mentioned below, an LFSR, which is a pseudo-random pattern generator, is such a TPG.

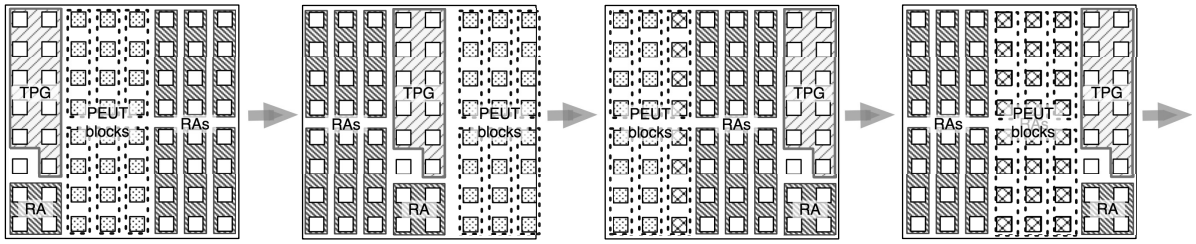
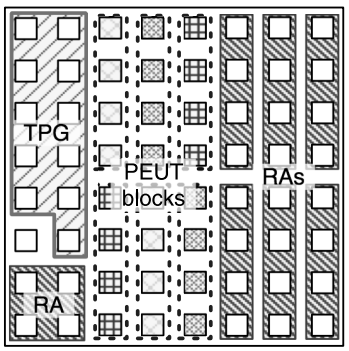
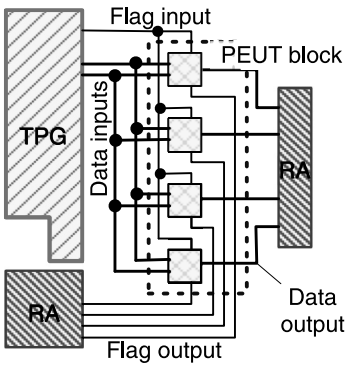


Fig. 3 Test frame shifting.

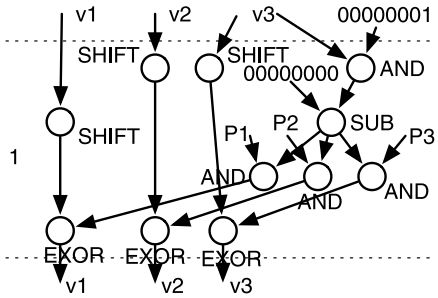


(a) Components in test frame.

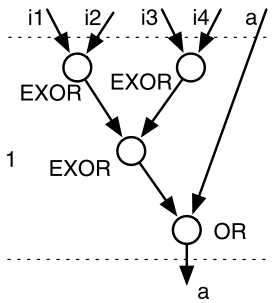


(b) Interconnect between TPG, PEUT block and RAs.

Fig. 2 Test frame F_1.



(a) TPG.



(b) RA.

Fig. 4 Data flow graphs of TPG and RA in test frame F_1.

by another RA. As shown in this example, some PE may be idle. Note that the function configured in a PEUT block can be different from that in another.

The test application for PEs can be completed by switching different test frames so as to cover all the PEs with every PEUT block (Fig. 3). In other words, PEs test themselves by shifting their roles in test frames. Let n_p be the number of PEUTs in a test frame. The number of test frames required for testing for all PEs can be expressed as

$$n_f = \lceil MN/n_p \rceil, \tag{1}$$

where N and M are the numbers of PEs in the DRP and of functions that can be performed by the PE, respectively[†]. For example, using the test frame shown in Fig. 2, when the number N of PEs on a DRP is 64 and the number M of functions to be tested is 10, we need $n_f = \lceil 10 \times 64/24 \rceil = 27$ test frames.

In the following discussion, we assume a single PE fault, i.e., at most one faulty PE occurs. By designing appropriate test frames, we can apply this test concept to different fault models.

3.2 Configurations of TPG and RAs

Figure 4 (a) shows an example of implementation of the TPG, which configures a 24-bit LFSR. In this figure, a vertex represents an operation (or instruction) which is executed by a single PE, and an edge represents a variable. The digit "1" at the left side denotes a control step, i.e., this data flow is executed in one cycle. Note that the operations in the same control step are executed on their separate PEs, and hence the number of PEs required for this TPG is 11. Furthermore, some operations or PEs are connected without registers (i.e., they are chained), and the cycle time of this flow requires 4 unit delays (a critical path is (AND, SUB, AND, EXOR).)

The set of variables v_1, v_2 and v_3 denotes a current state of the shift register, and the set of constants P_1, P_2 and P_3

[†]The number of test patterns for each function is assumed to be identical for the sake of simplicity.

represents a characteristic polynomials of the LFSR. A pattern (v_1, v_2, v_3) is generated in a single latency whose cycle time is 4 unit delays, and is applied to PEUTs while stored as the next state of the LFSR. For applying a test to a PE block which has two 8-bit data inputs and one 1-bit flag input (17 bits in total) (Fig. 2 (b)), 17 of 24 bits in a pattern (v_1, v_2, v_3) generated by the LFSR can be used[†].

Figure 4 (b) shows an example of implementation of an RA comparing the data output responses obtained from the PEs via its input (i_1, i_2, i_3, i_4). If one of the output responses is different from the others, a result including bit 1 s will be stored as a signature a at a register. If the resultant signature a remaining after test application includes no bit 1, all the PEs in the corresponding block are fault free. The number of PEs required for this RA is 4. The latency is one, and its cycle time is 3 unit delays.

The *test latency* of a test frame, referring to the number of cycles required for applying a test pattern to PEUTs and for storing the corresponding responses to the register of RAs in the test frame, can be expressed as the sum of latencies of the TPG and an RA plus one cycle for a PEUT block. The resources (or PEs) of the components TPG, PEUT and RA are independent of one another, and hence they can perform in a pipeline fashion where the number of stages is three: test pattern generation (TPG), test application (PEUT) and response observation (RA). Thus, the test application time for testing all the PEUTs with a test frame F is denoted by

$$T = t_c n_f (N_{TP} - 1 + \lambda) + T_L \tag{2}$$

where t_c is the cycle time of F (or the maximum cycle time of the components), n_f is the number of test frames (Eq. (1)), N_{TP} is the number of test patterns for a function of the PE, λ is the test latency of F, and T_L is the time required for loading configuration data. Note that T_L depends on the scheme for loading configuration data. For example, if the DRP can load configuration data while it operates, the loading time is considerably small. For example, when using the test frame F_1 shown in Fig. 2 (TPG and RA are shown in Figs. 4 (a) and (b), respectively.), its test latency is $\lambda = 1 + 1 + 1 = 3$ with cycle time 4, and thus the test application time can be expressed as $T = 4{\times}27(N_{TP}{-}1{+}3){+}T_L = 108N_{TP}{+}216{+}T_L$.

3.3 Optimal Test Frame for Test Application Time

Figure 5 shows another implementation of TPG. The latency of this TPG is 2, while the number of PEs for it is 6, derived from the maximum number of operations for all the control steps. Note that the latency of a TPG corresponds to the number of contexts required for this TPG. According to the implementation of the TPG, the RA can be also implemented so as to reduce the number of PEs for the RA with the same latency, and we can obtain another test frame as shown in Fig. 6. As shown in this figure, the numbers of PEs for the TPG and RAs are 6 and 22 respectively (each RA consists of one PE), and consequently the number of PEUTs can be 36. Thus, all the PEUTs can be covered with

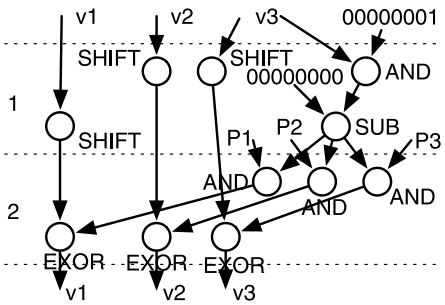


Fig. 5 Data flow graph of TPG in test frame F_2.

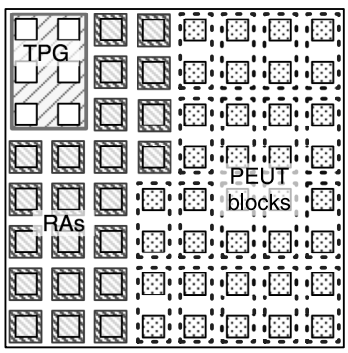


Fig. 6 Test frame F_2.

$n_f = 17$ test frames.

Note that the latency of the TPG (Fig. 4) in test frame F_1 is one, while that of the TPG (Fig. 5) is two. This means that the TPG in test frame F_2 requires two instructions or contexts. The maximum latency of the components (TPG, PEUT and RA) corresponds to the number of cycles required for each stage in the pipeline application, and it is equal to the number of contexts required for a test frame. Therefore, by denoting the number of contexts required for a test frame as n_{fc}, we can rewrite Eq. (2) as

$$T = t_c n_f (n_{fc}(N_{TP} - 1) + \lambda) + T_L. \tag{3}$$

Hence, the test application time with test frame F_2 is derived from $T = 2{\times}17{\times}(2{\times}(N_{TP}{-}1){+}5){+}T_L = 68N_{TP}{+}102{+}T_L$.

From the above observation we can see that according to the number of PEs required for the TPG, the number of PEUTs that can be tested with each test frame will change while the cycle time and the test latency of the test frame also differ. Thus, it is important to choose an appropriate test frame for reducing the total test application time.

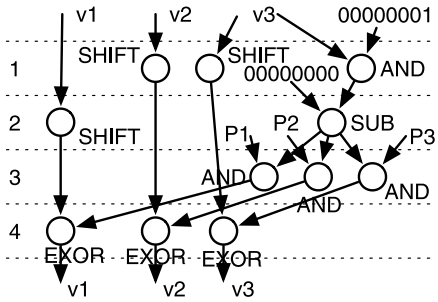
Here, let n_t and n_r be the numbers of PEs required for TPG and RA, respectively. The numbers n_t and n_r affect all the parameters in Eq. (3). The relationship between n_t/n_r and the other parameters are determined by the scheduling and binding for TPGs and RAs.

In order to analyze the relationship among the numbers

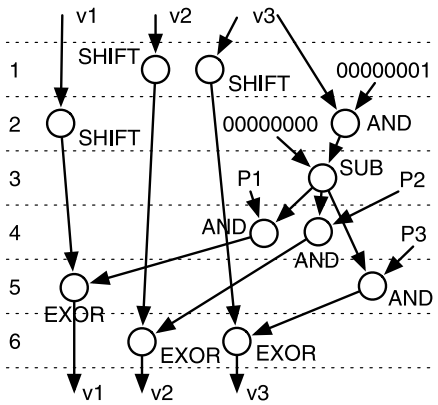
[†]A preliminary experimental result shows that the number of test patterns required by the TPG (or LFSR) is 3,000 enough for testing any function under consideration. The details are discussed in Sect. 3.4.

Table 1 Characteristics of test frames.

test frame	test latency λ	#PEs TPG n_t	#PEs RAs n_r	#PEs PEUTs n_p	#test frames n_f	#contexts per test frame n_{fc}	#contexts n_c	cycle time t_c	test application time T	test efficiency e
F_1	3	11	28	24	27	1	27	4	$108N_{TP} + 216$	6.00
F_2	5	6	21	36	17	2	34	2	$68N_{TP} + 102$	9.00
F_3	9	3	14	46	14	4	56	1	$56N_{TP} + 70$	11.00
F_4	13	2	12	50	13	6	78	1	$78N_{TP} + 91$	8.33



(a) TPG in test frame F_3.



(b) TPG in test frame F_4.

Fig. 7 Different TPG designs.

of PEs for the TPG, PEUTs and RAs, the cycle time (or the number of contexts) and the test application time, we designed several test frames and estimated them as shown in Table 1 and Fig. 7[†] . Here, we assume that the time T_L required for loading configuration data is zero. Note that as the index of test frame F_i $(1 \leq i \leq 4)$ increases, the test latency λ also increases. Designing the RAs whose latency is the same as that of the TPG in each test frame (F_2, F_3, F_4) for the purpose of pipeline application, in the same way as the TPG and RA of test frame F_1 in Fig. 4, we can calculate the test latency at $\lambda = 2 \times n_{fc} + 1$, where n_{fc} is the latency of the TPG and RAs and the latency of the PEUTs is one.

Note that, unlike conventional FPGAs, a DRP can reconfigure itself at every cycle by switching the contexts. Therefore, as shown in Table 1, while reducing the number $(n_t + n_r)$ of PEs used for the TPG and RAs (i.e., sharing the PEs), we can increase the number n_p of PEUTs that are tested simultaneously in every test frame. Although a test frame with small TPGs and RAs requires many contexts $(n_c = n_f n_{fc})$ due to dynamic reconfiguration, the cycle time t_c of the test frame decreases according to the increase in test latency λ. As a result, we can reduce the test application time T with a test frame whose latency is large.

From this table, however, we can also see that a test frame in which the number of PEs for TPGs and RAs is small cannot always achieve small test application time. Test frame F_4 achieves 50 PEUTs, which is larger than 46 ones by test frame F_3, whereas the cycle time of F_4 is not smaller than that of F_3 and the latency of F_4 is larger than that of F_3. Accordingly, in spite of reduction in the number of PEs for the TPG and RAs, the test application time with F_4 increases compared to that with F_3.

We can verify such optimality of test frames for reduction in test application time by means of *test efficiency*, expressed as

$$e = n_p/(n_{fc}t_c). \tag{4}$$

This equation means the ratio of the total number of functions of PEs are tested by one PE. The rightmost column in Table 1 shows test efficiencies of test frames F_1 through F_4. As shown, the test efficiency of F_3 is maximum, which means that F_3 is the best test frame for reducing the test application time. Thus, there exists an optimal test frame which can minimize the test application time.

3.4 Case Study

To ascertain the validity of this TPG configuration, we made preliminary experiments. In these experiments, we targeted the functions implemented by the ALU and DMU of a PE, e.g., ADD, SUB, AND (corresponding to the ALU), SRL (shift right logical) and SLL (shift left logical) (corresponding to the DMU), etc. We designed several combinational circuits which can execute these functions of the ALU and DMU, and then applied test patterns generated by several LFSRs, which are implemented by the TPG configuration, to the designed circuits. As a result we found out an appropriate LFSR (a feedback polynomial and a seed) that can detect all testable (non-redundant) stuck-at faults in any of the combinational circuits with 3,000 test patterns enough. This means that the configured TPG can generate effective test patterns for the functions implemented by the ALU and

[†]Because of the limitation on the output of PEs, the TPG, shown in Fig. 7 (b), in test frame F_4 requires an ingenious scheme to output a test pattern (v_1, v_2, v_3), e.g., the value of v_1 and the values of v_2, v_3 are separately fed to the PEUTs at distinct clock cycles. The detailed analysis of this scheme is omitted because the first objective is to analyze the relationship among the parameters shown in Table 1.

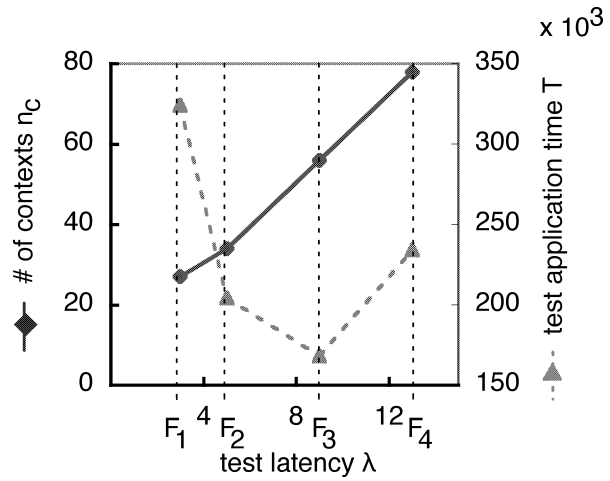


Fig. 8 Total number of contexts and test application time versus test latency.

DMU in PEs, and the number N_{TP} of test patterns is $3,000$ enough. In the following discussion, N_{TP} is fixed to $3,000$.

Figure 8 shows the relationship of the total number n_c of contexts and the test application time to the test latency for a test frame. Here, we assume that the time T_L for loading configuration data is zero. According to this figure, we can choose the best test frame for minimizing the test application time of a DRP for given test constraint and environment. For example, in case of no restriction on the number of contexts used for test application, e.g., a DRP has a rich context memory, by choosing test frame F_3, we can achieve the testing for all the PEs in the DRP with the minimum time ($T = 56\times3,000 + 70 + 0 = 168,070$). This can be also derived from the test efficiency mentioned above.

On the other hand, if the number of contexts used for test application is restricted, for example, in case 40 contexts can be stored by the DRP itself, we do not choose test frames F_3 and F_4, either. In this case, we can minimize the test application time ($T = 68\times3,000 + 102 + 0 = 204,102$) by means of test frame F_2.

Thus, there exists an appropriate test frame for given test constraint and environment, and accordingly, by selecting it we can efficiently achieve the testing of DRPs without extra hardware overhead.

4. Conclusions

This paper proposed a method for self-testing of dynamically reconfigurable processors (DRPs). We introduced a test frame in which the processor elements (PEs) perform test application to themselves while shifting their roles of a test pattern generator (TPG), PEs under test (PEUTs) and response analyzers (RAs). Our test method with test frames requires no hardware overhead. We analyzed the relationship of the structure of test frames to the number of contexts and test application time by exploring several designs of test frames, and showed that there exists an optimal test frame which minimizes the test application time for DRPs. In other words, the design of test frames is important to achieve an efficient test application.

In this paper, we focused mainly on the testing of PEs. Testing of interconnect and memories as well as controllers in DRPs is a remaining problem. We shall also discuss the architecture of testable dynamically reconfigurable processors.

Acknowledgements

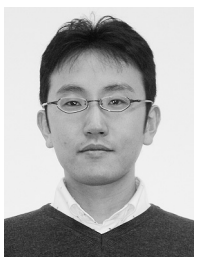
The authors would like to thank Prof. Yuki Yoshikawa and members of Computer Design Lab., Hiroshima City University for their valuable comments.

References

[1] H. Amano, "A survey on dynamically reconfigurable processors," IEICE Trans. Commun., vol.E89-D, no.12, pp.3179–3187, Dec. 2006.
[2] T. Sugawara, K. Ide, and T. Sato, "Dynamically reconfigurable processor implemented with IPFlex's DAPDNA technology," IEICE Trans. Inf. & Syst., vol.E87-D, no.8, pp.1997–2003, Aug. 2004.
[3] http://www.necel.com/drp/en/
[4] Y. Hasegawa, S. Abe, H. Matsutani, H. Amano, K. Anjo, and T. Awashima, "An adaptive cryptographic accelerator for IPsec on dynamically reconfigurable processor," Proc. IEEE Int'l Conf. Field-Programmable Tech., pp.163–170, 2005.
[5] H. Amano, "A dynamically adaptive hardware on dynamically reconfigurable processors," IEICE Trans. Commun., vol.E86-D, no.12, pp.3385–3391, Dec. 2003.
[6] T. Inoue, H. Fujiwara, H. Michinishi, T. Yokohira, and T. Okamoto, "Universal test complexity of field-programmable fate arrays," Proc. IEEE Asian Test Symp, pp.259–265, 1995.
[7] C. Stroud, S. Konala, P. Chen, and M. Abramovici, "Built-in self-test of logic blocks in FPGAs (Finally, a free lunch: BIST without overhead!)," Proc. IEEE VLSI Test Symp., pp.387–392, 1996.
[8] T. Inoue, S. Mitazaki, and H. Fujiwara, "Universal fault diagnosis for lookup table FPGAs," IEEE Des. Test Comput., vol.15, no.1, pp.39–44, Jan.-March 1998.
[9] M. Renovell, J. Figueras, J. Figueras, and Y. Zorian, "Testing the interconnect of RAM-based FPGA," IEEE Des. Test Comput., vol.15, no.1, pp.45–50, Jan.-March 1998.
[10] K. Katoh and H. Ito, "Built-in self-test for PEs of coarse grained dynamically reconfigurable devices," Proc. European Test Symp., pp.69–74, 2006.
[11] M. Inoue, K. Kambe, V. Singh, and H. Fujiwara, "Software-based self-test of processors for stuck-at faults and path delay faults," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J88-D-I, no.6, pp.1003–1011, June 2005.
[12] L. Chen, S. Ravi, A. Raghunathan, and S. Dey, "A scalable software-based self-test methodology for programmable processors," Proc. 40th Design Automation Conf., pp.548–553, 2003.

Tomoo Inoue is a professor of Graduate School of Information Sciences, Hiroshima City University. His research interests include test generation and high-level synthesis and design for testability and dependability, as well as design and test of reconfigurable devices. He received the B.E., M.E. and Ph.D. degrees from Meiji University, Kawasaki, Japan, in 1998, 1990 and 1997, respectively. From 1990 to 1992, he was with Matsushita Electric Industrial Co., Ltd. From 1993 to 1999, he was an assistant professor of Graduate School of Information Science, Nara Institute of Science and Technology. In 1999, he joined Faculty of Information Sciences, Hiroshima City University as an associate professor. Tomoo Inoue received WRTLT (Workshop on RTL and High Level Testing) 2004 Best Paper Award. He is a member of the IEEE Computer Society, and IPSJ.

Takashi Fujii received his Bachelor and M.E. degrees of Information Engineering from Hiroshima City University in 2005 and 2007, respectively. He is currently with the Canon Inc.



Hideyuki Ichihara received his M.E. and Ph.D. degrees from Osaka University in 1997, 1999, respectively. He was a research scholar of University of Iowa, U. S. A. from February to July in 1999. Since December 1999, he had been an assistant professor of Hiroshima City University, and he is currently an associate professor of the university. He received IEICE Best Paper Award 2004 and Workshop on RTL and High Level Testing 2004 Best Paper Award. His research interests are VLSI testing and design for testability. He is a member of the IEEE Computer Society.