Buffer and Controller Minimisation for Time-Constrained Testing of System-On-Chip

Anders Larsson, Erik Larsson, Petru Eles and Zebo Peng

Embedded Systems Laboratory
Linköpings Universitet
SE-582 83 Linköping, Sweden
{anlar, erila, petel, zebpe}@ida.liu.se

Abstract

Test scheduling and Test Access Mechanism (TAM) design are two important tasks in the development of a System-on-Chip (SOC) test solution. Previous test scheduling techniques assume a dedicated designed TAM which have the advantage of high flexibility in the scheduling process. However, hardware overhead for implementing the TAM and additional routing is required of the TAMs. In this paper we propose a technique that makes use of the existing functional buses for the test data transportation inside the SOC. We have dealt with the test scheduling problem with this new assumption and developed a technique to minimise the test-controller and buffer size for a bus-based multi-core SOC. We have solved the problem by using a constraint logic programming (CLP) technique and demonstrated the efficiency of our approach by running experiments on benchmark designs.

1. Introduction

The need of short time-to-market windows together with the growth in the chip integration capability has forced the designers to develop new design methodologies. One such method is to place reusable cores, e.g. processors, memories, interface devices, etc., into a System-on-Chip (SOC). This method has made it possible to design complex systems and satisfy the short time-to-market requirements. However, more complex designs need more test data and hence longer test time. Furthermore, since the cores are not directly accessible from the SOC's input and output pins, it is necessary to integrate special test access mechanisms (TAM) in order to transport the test data inside the chip. Difficulties like these have made test time and design for test a major cost factor in the design and production of SOCs. In order to lower this cost the application of the tests should be scheduled to minimize the test time and the TAM should be designed with as low area overhead as possible.

Different strategies have been proposed to solve the scheduling and/or the TAM-design problem [1], [4], [7], [13], [14], [15], [16], and [18]. The main disadvantage with these approaches is that they require additional wiring overhead. Three different scan chain architectures as TAM are proposed in [1]. The use of different types of TAMs, e.g. test-buses, transparency, and other bypass modes, are considered in [4]. In [5] it is described how a system, containing an AMBA-bus [2], can be scan-tested using a scan-test harness. In [7] the test access technique of the AMBA bus is discussed. In [15] a packet switching communication network as TAM is presented and in [18] a dedicated test bus is proposed.

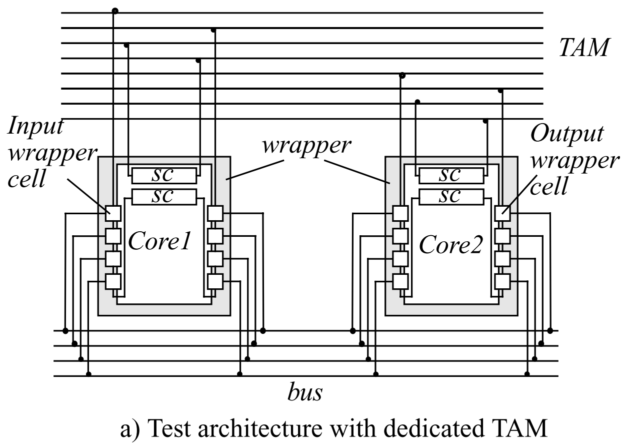
In this work we explore the possibility of using existing components of the design as TAM. We have chosen to use the functional bus as TAM, since a large number of SOCs use a bus-based architecture [9]. There exist several different bus standards; e.g. AMBA, Core Connect, Core Frame, HIBI, VCI, and Wishbone [17]. Common for all is that they have a high bandwidth, e.g. the AMBA bus which is 32-512 bits wide operating at high speed [2], which makes them suitable for transporting test data. In order to shorten the test time by having concurrent application of the tests, buffers are introduced. The buffers, which are placed between the core and the bus, are used for temporary storage of the test vectors. A buffer also makes it possible to test the cores for other types of faults. Besides the stuck-at fault the core can be tested for other types of faults which need at-speed testing. This becomes possible since the tests are stored close to the core before being applied. To keep the size of each buffer reasonably small, we assume that each test can be divided into a number of test packages. This will also increase the flexibility in the scheduling of the tests.

The buffers are one part that is added in our approach, which should be considered during the design process. Another issue considered by our approach is the test-controller, which is also discussed in [6]. The test-controller is needed in order to implement the test schedule and to control the transmissions of tests on the bus. In this paper we have solved the scheduling problem and minimised the buffer size, and test-controller complexity. We have formulated the problem, solved it, and demonstrated its significance using Constraint Logic Programming (CLP).

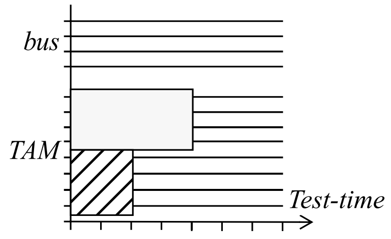
The rest of the paper is organized as follows. An overview of the SOC architecture and test access is given in Section 2. The exact problem formulation is presented in Section 3. The CLP model is described in Section 4 and it is followed by experimental results in Section 5. The paper is concluded and future work is discussed in Section 6.

2. SOC test architectures

A common way to design SOC communication is to use a shared bus, which is connected to all cores. A dedicated TAM for the transportation of test data has the advantage of high flexibility. It also offers the possibility of a trade-off between the test time and the number of wires used in the TAM. A large number of wires requires extra silicon area to the design, but it enables parallel transportation of test vectors, which will shorten the test time. A design with a dedicated TAM and a shared bus can be implemented as illustrated in Figure 1(a). This example shows two cores that are tested with one test per core. In this example the cores are scan tested by two scan-chains per core. The buses are not used during the test mode. By occupying four TAM-wires each, the two tests can be applied concurrently, as illustrated in Figure 1(b).



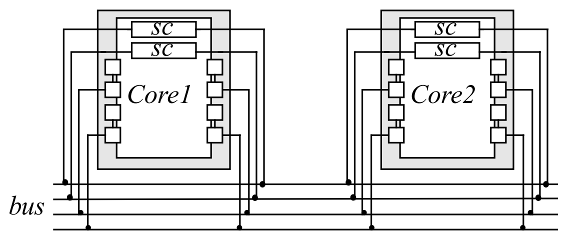
a) Test architecture with dedicated TAM



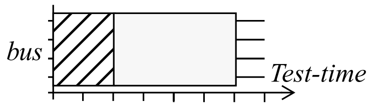
b) Test scheduling with TAM

Figure 1. Test architecture and scheduling with TAM

One of the ideas behind our work is to use the functional bus for transportation of test data. The advantage is that the TAM can be excluded, as shown in Figure 2(a). In the example it is possible to connect the scan-chains and the wrapper cells into four chains. The disadvantages with using the bus are the need for special control and buffers. It can also be difficult to have other types of communication than sequential, shown in Figure 2(b).



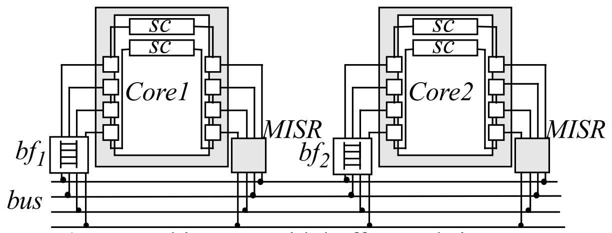
a) Test architecture with only the bus



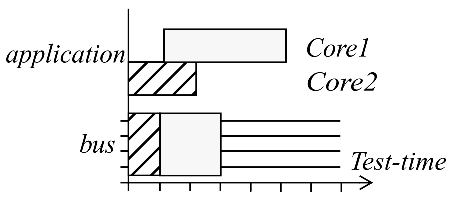
b) Test scheduling on the bus without buffers

Figure 2. Test tranportation and scheduling using the functional bus

Since it is common to have a high transfer rate on the bus and a width of 32 bits or more, it is possible to transport the test vectors at a very high speed on the bus. Placing a buffer between each core and bus makes it possible to take advantage of the high speed transportation. In the small example in Figure 3(a), the tests can be sent on the four wires of the bus, and in the buffer, the test is converted from parallel to serial in order to fit the scan chains (load). This means that it will take longer time to apply the test than to transport it. The test is transported in parallel on the bus and when the test reach the core it is serialized into the scan- chains. Since the test is stored in the buffer while applying it to the core, another test can be transported on the bus. This is illustrated in Figure 3(b).



a) Test architecture with buffers and signature analyzers (MISR)



b) Test scheduling and application with buffers

Figure 3. Functional bus and buffers

The buffers can also be used for other purposes. For example, they may be complemented with logic that supports at-speed testing, parallel to serial conversions or other types of manipulations needed to adjust the test before applying it to the core.

The position of the buffers in the system is illustrated by a small example in Figure 4. The system consists of three cores, c_1, c_2, and c_3, all connected to the bus b. Each core c_i is associated with a buffer bf_i, which is placed between the core and the bus. Also connected to the bus are two test components, T_{src} and T_{ctrl}. In this paper we assume that the tests are all produced in the test-source T_{src}. The test-controller, T_{ctrl}, is responsible for the invocation of transmissions of the tests on the bus. The system is tested by applying the tests, generated or stored in the test source, to the cores. We assume that the core itself handles the evaluation of the test results. This can be done by, for example, a signature analyser. Information needed for the final test result evaluation is also sent via the bus.

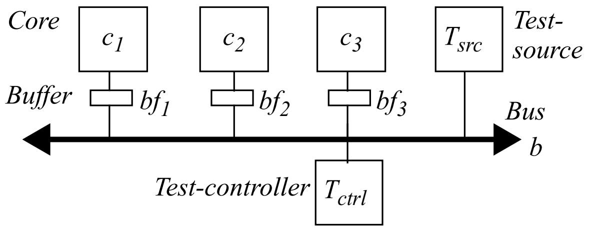


Figure 4. Bus-based architecture

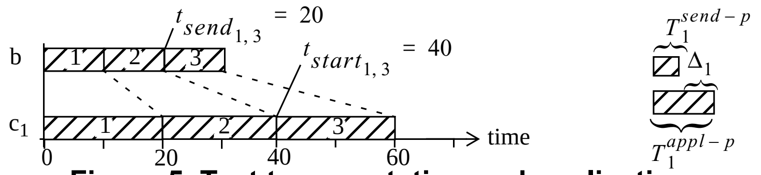


Figure 5. Test transportation and application

In our approach, each test, T_i, can be divided into m_{T_i} packages (a set of test vectors). The designer decides the size of the packages (the number of vectors). There are two reasons for dividing the tests into packages. The first reason is to shorten the test time of the system by giving more flexibility to the scheduling, and the second is to decrease the buffer size needed at each core.

We assume that the transportation-time, $T_i^{send\text{-}p}$, of a package on the bus is shorter than the application-time, $T_i^{appl\text{-}p}$. The reason for this can, for example be, that the test data consists of scan test that is transported in parallel on the bus and then applied in serial to the core. The example indicates again the need of storing the test in a buffer while it is applied so that another test can be transported on the bus. The size of the buffer, however, does not have to be equal to the size of the packages. This is explained by the fact that the test data in a package can be applied immediately when it arrives at the core. The buffer size bs_i, associated to a core c_i, is calculated by the following formula:

$$bs_i = max(k_i \times (t_{start_{ij}} - t_{send_{ij}}) + \Delta_i),\ j \in (1, m_{T_i}),$$

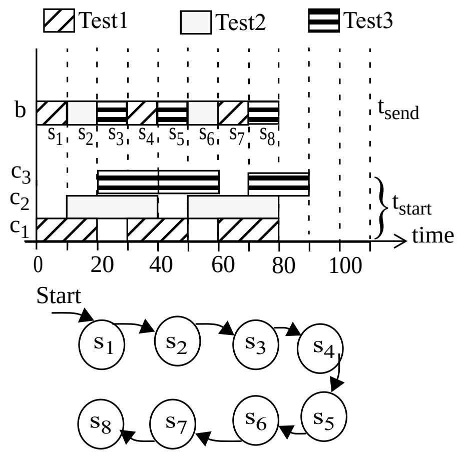
where the constant k_i represents the rate of which the core can apply the test, the time $t_{start_{ij}}$ is the scheduled start time of the application of the package j from test T_i at the core, and $t_{send_{ij}}$ is the start time for sending the package on the bus. The constant Δ_i represents the leftover package size, which is the size of the test vectors that remain in the buffer after the transportation of the package terminates. This constant Δ_i is determined by the difference between $T_i^{appl-p}$and T_i^{send-p}, which is multiplied by the constant k_i.

The calculation of the buffer size is further illustrated in Figure 5, which shows the bus schedule and the application of a test T_1 to core c_1, with $T_1^{appl} = 60$, $T_1^{send} = 30$, $m_{T_1} = 3$, and $k_1 = 1$. In this small example the core has not finished the testing of the package sent at time point $t_{send_{1,2}} = 10$ before the package sent at $t_{send_{1,3}} = 20$ arrives at the core. This forces the buffer size to be increased. For this example the buffer size will be equal to $1 \times (40 - 20) + 10 = 30$, which is the difference between the termination of applying the last test package and the end point of transporting this package.

In the example in Figure 5 there is only one core and one test and the example does not show the impact of the scheduling of packages on the complexity of the test controller. The following example illustrates the minimisation of the buffer size and the test controller complexity. We make use of the example system from Figure 4, which consists of three cores c_1, c_2, and c_3 which are tested with three tests, T_1, T_2, and T_3, respectively. We have divided the tests into a total number of 8 packages, all with the same application-time and minimum package size, but different transportation-times. The characteristics of the tests are given in Table 1. In this work we assume that the total test time of the system is given as a constraint by the designer. In this example the total test time is set to 90 time units, which is the minimal time for applying these tests. This time is the sum of the transportation times plus the smallest value of all Δ_i.

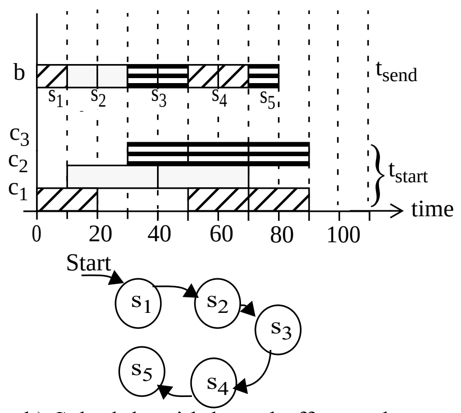
Table 1. Test characteristics

Test	Nr packages	Application-time (T_i^{appl})	Transportation- time (T_i^{send})	Δ_i
T_1	3	60	30	10
T_2	2	60	20	10
T_3	3	60	30	10



▨Test1
□Test2
▤Test3
b
s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8
t_{send}
c_3
c_2
c_1
t_{start}
time
0 20 40 60 80 100
Start
s_1 s_2 s_3 s_4 s_5 s_6 s_7 s_8

a) Schedule with small buffers, which leads to a large number of controll states



b) Schedule with large buffers and few controll states

Figure 6. Scheduling example

Two different schedules for the 8 packages derived from the three tests are illustrated in Figure 6. In the first schedule, shown in Figure 6(a), the packages are sent in such a way that the application of the previous package has finished before a new one arrives. This leads to small buffers since every package can be applied immediately as they arrive, that is $t_{start_{ij}} - t_{send_{ij}} = 0$ for all packages. The buffer sizes for this schedule are, bs_1 =10 ($bs1_1 = 1 \times (0) + 10$), bs_2 =20, and bs_3 =10. In the second schedule, Figure 6(b), some packages are grouped together in pairs, which will produce larger buffers, bs_1 =20 ($bs1_1 = 1 \times (70 - 60) + 10$), bs_2 =40, and bs_3 =20.

The advantage of having larger buffers is the decreasing complexity of the test-controller. In this work we assume that the test-controller is implemented as a finite state machine, where each state is responsible for the transmission of packages to a single core on the bus. A transition between two states represents a change of test to be transported on the bus from one core to another core. Figure 6 illustrates the difference in the number of states between a schedule with small buffers, Figure 6(a), and a schedule with larger buffers, Figure 6(b). If only one package is sent in each state, the number of states will be equal to the total number of packages in the system, as illustrated in Figure 6(a). If the maximum number of packages is sent from each state, the number of states will be equal to the number of tests.

3. Problem formulation

In this section an exact problem formulation is given.

A given system consists of a set of cores $C = \{c_1, c_2, ..., c_N\}$, where N is the total number of cores. Each core c_i, has a buffer, bf_i of size $bs_i, (i = 1, 2, ..., N)$. The system also consists of a test-controller, T_{ctrl}, with a number of states, *NrStates*. A maximum allowed test time for the system, T_{max}, is given as a constraint by the designer. Also given is a set of tests $T=\{T_1, T_2, ..., T_N\}$[1], where T_i is a set of test vecors, which is to be applied to the core c_i. For each test T_i, the following information is given:

- the application-time T_i^{appl} is the time to apply the test,
- the transportation-time T_i^{send} is the time to transport T_i from T_{src} via the bus to core c_i,
- the size (the number of test vectors) of the test, T_i^{size},
- and the size of a package, T_i^{size-p}.

A test T_i, is divided into m_{T_i} packages, each of the same size, T_i^{size-p}, $P=\{p_{11}, p_{12}, ..., p_{21}, p_{22}, ..., p_{Nm_{T_N}}\}$.

1. It is possible that a test for a core can be composed of test-vetors from different tests.

The package size is determined by the following formula:[2]

$$T_i^{size-p} = \left\lceil \frac{T_i^{size}}{m_{T_i}} \right\rceil$$

The time to apply a package, T_i^{appl-p}, is determined by:

$$T_i^{appl-p} = \frac{T_i^{appl}}{m_{T_i}}$$

Associated to each package p_{ij} of test T_i where $j \in (1, m_{T_i})$, are three time points, $t_{start_{ij}}$, $t_{send_{ij}}$, and $t_{finish_{ij}}$. The time to send, $t_{send_{ij}}$, represents the start of the transmission of package, p_{ij}, on the bus. The time, $t_{start_{ij}}$, is the start time of the test at the core c_i. Finally, $t_{finish_{ij}}$ is the time when the whole package has been applied. The finish time, $t_{finish_{ij}}$ is computed by the following formula:

$$t_{finish_{ij}} = t_{start_{ij}} + T_i^{appl-p}$$

The objective of our proposed technique is to find $t_{start_{ij}}$ and $t_{send_{ij}}$ for each package in such way that the total cost is minimised while satisfying the test time constraint, T_{max}. The total cost for the test is computed by a cost-function, that consists of the system's total buffer size and the complexity of the controller given as follows:

$$Cost_{Tot} = \alpha \times Cost_{Controller} + \beta \times Cost_{Buffer},$$

where α and β are two coefficients which can be used to set the weight of the controller and the buffer costs and $Cost_{Controller} = k_1^C + k_2^C \times NrStates$; $Cost_{Buffer} = k_1^B + k_2^B \times BufferSize$,
where the constants k_1^C and k_1^B are the base costs, which is the basic cost for having a controller and buffers, respectively and k_2^C and k_2^B are design-specific constants that represents the implementation cost parameters for the number of states and the buffer size.The total buffer size in the system is determined by:

$$BufferSize = \sum_{i=1}^{N} bs_i$$

The complexity of the test-controller, T_{ctrl}, is determined by the number of states, *NrStates*.

4. CLP modelling

In this section we describe the problem formulated as a constraint logic programming (CLP) problem.

The development of CLP [10] started in the late 80's and is descended from logic programming. It combines constraint solving with logic programming and uses an exhaustive method in the search for the best solution to a problem. CLP is a declarative language, which means that the program contains a description of the solution to the problem. The description is composed of a set of constraints (relations between objects), which limits the search space thus making CLP suitable for solving combinatorial problems such as scheduling and resource allocation.

In this work we have modelled the system in a CLP program, consisting of two main components, *Test* and *Package.* The Test component contains all given information for the tests and is used as the input to the program. In order to find a feasible solution that minimises the total cost the program ensures that a number of different constraints are fulfilled. These constraints are:

- the packages belonging to the same test have to be sent in a given order, i.e., $t_{start_{ij+1}} \geq t_{finish_{ij}}$,
- the start-time of a package should be later or equal to the time of transmission on the bus: $t_{start_{ij}} \geq t_{send_{ij}}$,
- the time when a package has been completely applied to the core is equal to the time it starts

2. This means that the last test package may have a smaller number of test vectors than T_i^{size-p}. We assume that this package is filled up with empty vectors.

the application plus the time used for the application: $t_{finish_{ij}} = t_{start_{ij}} + T_i^{appl-p}$,

- the finish time of any test can not exceed the total test time limit, T_{max}: $t_{finish_{ij}} \leq T_{max}$.

The buffer size at a core is determined by the following formula :

$$bs_i = max(k_i \times (t_{start_{ij}} - t_{send_{ij}}) + \Delta_i) .$$

The cost of a test is given by $Cost_{Tot} = \alpha \times Cost_{Controller} + \beta \times Cost_{Buffer}$,

where $Cost_{Controller}$ is connected to the number of states and $Cost_{Buffer}$ is connected to the buffer size as described in section 3 Problem formulation.

With the above constraint set, the constraint solver searches for a schedule that minimises the cost of the test.

5. Experimental results

We have used four different designs in our experiments. One small example, Ex1, which has been described in Section 2, and three benchmarks, ASIC Z [19], [3], Kime [11] and System L [13]. Kime consists of 6 cores, while ASIC Z and System L consists of 9 respectively 13 cores. Detailed information for these benchmarks can be found in [12]. The main characteristics of the four designs are presented in Table 2.

Table 2. Design characteristics

Design	Nr Tests	Nr Packages	Min Buffer size	Max Buffer size
Ex1	3	8	40	100
Kime	6	20	93	340
ASIC Z	9	38	111	419
System L	13	39	280	988

We have used the CLP-tool CHIP (V 5.2.1) [8] for the implementation. The experiments have been performed in two steps. In the first step the minimal test time is obtained assuming no division of the tests into packages, which corresponds to the traditional approach assumed by several test scheduling techniques. For experimental purposes we have used the obtained testing time as the time constraint in the second step, where the cost is minimised using our CLP approach. In this particular experiments, the cost is minimised according to the following cost function:

$$Cost = 1 \times Cost_{Controller} + 1 \times Cost_{Buffer},$$

where $Cost_{Controller} = 10 + 5 \times NrStates$ and $Cost_{Buffer} = 10 + 1 \times BufferSize$.

For simplicity, we have, in the experiments, set the size of each test, T_i^{size}, to be equal to the application time, T_i^{appl}.

The experimental results are presented in Table 3, where the cost from our approach has been compared to the cost obtained by a straightforward approach. The result shows a decrease with 35 to 61 percent of the cost, which demonstrates that our approach can decrease the cost by minimising the buffer and the controller, without exceeding the test time limitation.

6. Conclusions and future work

We have approached the problem of test application of multiple-core systems, both in terms of test time and considering the trade-off between the test controller complexity and the size of the buffers. We have used constraint logic programming to model and solve our problem. Since constraint logic programming uses an exhaustive search approach, execution times can become large for complex examples. A natural extension of the work is to find a heuristic that would work for larger examples. The problem could be extended to include sharing of test evaluation mechanism and scheduling on multiple buses or networks on chip.

Table 3. Experimental results

Design	Test Time	Straightforward Approach (Sequential Scheduling)			Our Approach			Cost Comparison
		Nr States	Buffer size	Cost	Nr States	Buffer size	Cost	
Ex1	111	3	100	135	8	40	100	-35%
Kime	257	6	340	390	14	215	305	-28%
Asic Z	294	9	419	484	34	128	318	-52%
System L	623	13	987	1072	37	460	665	-61%

7. References

[1] J. Aerts, E. J. Marinissen, "Scan Chain Design for Test Time Reduction in Core-Based ICs," *Proceedings of the International Test Conference*, pp. 448-457, 1998.

[2] AMBA Specification Rev 2.0, ARM Ltd., 1999

[3] R. Chou, K. Saluja, V. Agrawal, "Scheduling Tests for VLSI Systems Under Power Constraints," *IEEE Trans. On VLSI Systems*, vol. 5, no. 2, pp. 175-185, 1997.

[4] E. Cota, L. Carro, A. Orailoglu, and M. Lubaszewski, "Test planning and design space exploration in a core-based environment," *Proceedings of Design, Automation and Test in Europe (DATE)*, pp. 478-485, 2001.

[5] C. Feige, J. ten Pierick, C. Wouters, R. Tangelder, and H. Kerkhoff, "Integration of the scan-test method into an architecture specific core-test approach", *Proceedings of the IEEE European Test Workshop,* pp. 241-245, 1998.

[6] S. K. Goel, E. J. Marinissen, "Control-Aware Test Architecture Design for Modular SOC Testing," *IEEE European Test Workshop Digest of Papers,* pp. 129-134, 2003.

[7] P. Harrod, "Testing reusable IP-a case study," *Proceedings of International Test Conference,* pp. 493-498, 1999.

[8] P. Van Hentenryck, "The CLP language CHIP: constraint solving and applications," *Compcon Spring '91. Digest of Papers ,* 25 Feb-1 Mar 1991 , pp. 382 -387, 1991.

[9] J. Huang, M.K. Iyer, and K. Cheng, "A self-test methodology for IP cores in bus-based programmable SoCs," *Proceedings of the19th IEEE VLSI Test Symposium (VTS),* pp. 198-203, 2001.

[10] J. Jaffar and J.-L. Lassez, "Constraint Logic Programming," *Proceedings of the 14th. ACM Symposium on Principles of Programming Languages (POPL),* pp. 111-119, 1987.

[11] C. R. Kime, "Test scheduling in testable VLSI circuits," *Proceedings of International Symposium on Fault-Tolerant Computing*, pp. 406-412, 1982.

[12] E. Larsson, A. Larsson, and Z. Peng, "Linkoping University SOC Test Site," *http://www.ida.liu.se/labs/eslab/soctest/*, 2003.

[13] E. Larsson and Z. Peng, "An integrated System-On-Chip test framework," *Proceedings of Design Automation and Test in Europe (DATE),* pp. 138-144, 2001.

[14] E. J. Marinissen, R. Arendsen, et al., "A structured and scalable mechanism for test access to embedded reusable cores," *Proceedings of International Test Conference,* pp. 284-293, 1998.

[15] M. Nahvi and A. Ivanov, "A packet switching communication-based test access mechanism for system chips," *IEEE European Test Workshop (ETW),* pp. 81-86, 2001.

[16] M. Nourani and C. Papachristou, "An ILP Formulation to optimize test access mechanism in system-on-chip testing," *Proceedings of International Test Conference,* pp. 902-910, 2000.

[17] E. Salminen, V. Lahtinen, Kimmo Kuusilinna, and Timo Hämäläinen, "Overview of bus-based system-on-chip interconnections," *IEEE International Symposium on Circuits and Systems (ISCAS),* vol. 2, pp. 372-375, 2002.

[18] P. Varma and B. Bhatia, "A structured test re-use methodology for core-based system chips," *Proceedings of International Test Conference,* 1998, pp. 294-302, 1998.

[19] Y. Zorian, "A distributed BIST control scheme for complex VLSI devices, " *Proceedings of VLSI Test Symposium,* pp. 4-9, 1993.