An Efficient Test Scheduling Algorithm in Networks on Chip Architecture

Jin-Ho Ahn[1], Byung In Moon[2], and Sungho Kang[1]
[1] Department of Electrical and Electronic Engineering, Yonsei University
134 Shinchon-dong, Sodaemun-gu, Seoul, 120-749, Korea
[2] School of Electrical Engineering & Computer Science, Kyungpook National University
1370 Sankyuk-dong, Buk-gu, Daegu, 702-701, Korea
E-mail: sominaby@soc.yonsei.ac.kr, bihmoon@knu.ac.kr, shkang@yonsei.ac.kr

Abstract: It may be impractical to have TAM for test usage only in NoC because it causes enormous hardware overhead. Therefore, the reuse of on-chip networks for TAM is very attractive and logical. In network-based TAM, an effective test scheduling for built-in cores is also important to minimize the total test time. In this paper, we propose a new efficient test scheduling algorithm for NoC based on the reuse of on-chip networks.

1. Introduction

NoC (Networks on Chip) can be defined as a kind of SoC (System on a Chip) that has an interconnection structure like micro-networks. Micro-networks, "on-chip networks" in other words, are an on-chip interconnection that uses a network protocol based on communication layers. Some test architectures incorporating NoC characteristics have been proposed recently. TAM (Test Access Mechanism) is the most activated area in those architectures. However, it is not feasible for NoC since the separate test bus causes excessive hardware overhead. Thus the reuse of on-chip networks for TAM becomes inevitable. An effective test scheduling is also important to minimize the total test time because NoC generally includes hundreds of cores.

In this paper, we propose a new efficient test scheduling algorithm for NoC based on the reuse of on-chip networks. The proposed algorithm has two dominant features. One is a deflection routing of test packets to satisfy simple router operations and minimize hardware overhead of routers. The other feature is an asynchronous test clock strategy. The asynchronous test clock platform can enhance test parallelization more than the multi-source/sink platform described in the previous algorithms.

2. Related Work

The general concept of the reuse of on-chip networks for TAM is shown in [1]-[2]. Test scheduling methods in NoC can be grouped roughly into two main categories: a packet-based scheduling and a core-based one. A core-based scheduling method determines the test order of each core [3]. However, it is impractical to use the core-based approach in a real situation because it is impossible to test cores with variable test clocks simultaneously. A packet-based scheduling determines the order of generation and transmission of test packets for cores according to the priority of each core. E. Cota has proposed the test scheduling based on a packet-switching [4]. In [4], test vectors and test responses per core are represented as a set of packets to be transmitted throughout the networks, and the packets are scheduled to minimize the total test time using test parallelism. Test parallelism means that several cores are tested simultaneously through maximizing the network bandwidth. Some enhanced versions of this algorithm have been reported. One is the supplement of power constraints [5]. While a packet-based one having many merits, its experimental results have proven inferior to those of core-based one up to now.

3. Proposed Idea

3.1 Deflection Routing

A deflection routing, also known as a hot-potato routing, is based on the idea of delivering an input packet to an output channel in one cycle within a router. It assumes that each router has the equal number of input and output channels. In a deflection routing, when contention occurs and the desired output channel is not available, an input packet will pick any alternative available output channels to continue moving to the next router instead of waiting.

An NoC test scheduling should be comparable for as many NoC structures as possible. Therefore, the minimal additional logic and simple control for the test are quite helpful to make the scheduling algorithm robust in whatever NoC will be used. Busch's algorithm [6] needs not to have any buffer in a router, and not to require any flow control. Thus the algorithm can be implemented regardless of the router structure and the routing algorithm used for data communication between cores.

While most routing procedures implemented in the proposed test scheduling algorithm are similar to Busch's, test packets for each core can be assigned with a different priority state according to the core priority when the packets are generated in a test source. The priority of a core is determined by its test time. This approach can reduce the total test time in parallel with a sorting of the cores in decreasing order of test time as introduced in previous studies.

3.2 Asynchronous Clock Platform

On-chip networks are generally much faster than testing speeds of embedded cores. Therefore, several cores can be tested at the same time. The difference of clock speed between the network and the core roughly corresponds to the number of test sources and sinks. For example, if on-chip networks operate two times faster than a tested core, it has the same effect as if there are two test sources and two sinks. However, an asynchronous test clock platform shows better results than multi-sources and sinks under the same condition since the asynchronous platform can fully schedule all cores by packets.

3.3 Test Scheduling Procedure

In this study, we used 2-D mesh topology with channels set to be 32bit wide. A test source can inject packets at a rate of one packet per network time step, and a test sink can absorb packets at the same rate as the test source. Each packet contains a header including its destination, priority, and packet id indicating the position of the packet within a test pattern. A router will receive a packet, and decide where to go with it using the proposed routing rules based on the packet's destination. We illustrate the experimental conditions for d695 in Figure 1. The pseudo-code of the algorithm is shown in Figure 2.

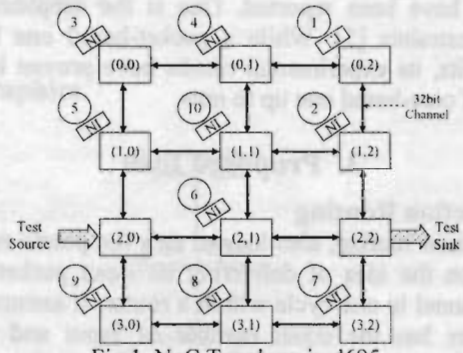

Fig 1. NoC Topology in d695

```
program NoC_test_schedule
 set mesh size, the position of embedded cores;
 set the priority of cores by their test time;
 set test information of each core;
 set the position of a test source and sink;
 set the on-chip network speed;
begin
 until(all cores are tested) {
  repeat network speed {
   Router_operation in all routers;
   Test_source_operation;
   Test_sink_operation;
  }
  test operation in all cores;
 }
end
program Router_operation
begin
 while(there is an input packet) {
  case(packet.destination)
   when 'here':
    if(dest. is core and NI input port is idle)
     transfer packet into NI;
    else if(dest. is sink and sink input port is idle)
     transfer packet into sink;
    else
     deflects to random output port;
   when 'not here':
    deflection routing by the packet destination;
 }
end
program Test_source_operation
begin
 if(there is a free port) {
  fetch a test packet from queue of a test source;
  deflection routing by the packet destination;
  generate a next test packet and queuing;
 }
end
program Test_sink_operation
begin
if(there is a input packet) {
  absorb a test packet;
  update test information;
  analyze the test response;
 }
 end
```

Fig 2. Pseudo-Code of NoC Test Scheduling

4. Experimental Results

The proposed algorithm is evaluated by the test time through C-level simulations. The algorithm takes a minute in a d695, and is not over 10 minutes even in a p93791 on a Sun Microsystems 1.2-GHz UltraSPARC® III. Table 1 displays the simulation results of the proposed algorithm compared to previous results for four different ITC '02 benchmark circuits. Results in (3) based on core-based scheduling is superior to those of the proposed in *some* cases. However, as mentioned in section 2, the core-based approach is so theoretical and impractical that it is not suitable for a real situation. We expected the results of the proposed algorithm would exceed those of (3) if we increase the test clock frequency of networks and cores with high priority. As compared with results in (4) based on packet-based scheduling similar to the proposed one, the test scheduling results using the proposed idea is quite noticeable in all cases. Furthermore, we can see from the results that the proposed algorithm is more effective under conditions of large circuits and high speed networks. This is very encouraging for the practicality and feasibility of the proposed algorithm.

Table 1. Test Scheduling Results

Circuit Name	# of Ports or Relative Clock Speed	Results in (3)	Results in (4)	Proposed
d695	2/2 or 2	18869	26012	20075
	3/3 or 3	13412	20753	12759
	4/4 or 4	10705	14785	10774
	- or 5	N.A	N.A	10088
	- or 6	N.A	N.A	10035
g1023	2/2 or 2	25062	31898	28616
	3/3 or 3	17925	22648	18434
	4/4 or 4	16489	18851	16168
	- or 5	N.A	N.A	15800
	- or 6	N.A	N.A	15360
p22810	2/2 or 2	271384	315708	312296
	3/3 or 3	180905	222432	208595
	4/4 or 4	150921	170999	162661
	- or 5	N.A	N.A	132982
	- or 6	N.A	N.A	115409
p93791	4/4 or 4	333091	435787	342819
	- or 5	N.A	N.A	285793
	- or 6	N.A	N.A	248108

5. Conclusion

The proposed test scheduling algorithm provides superior performance in spite of low hardware overhead. Thus we expect that the algorithm can be widely applicable due to its feasibility and practicality.

References

[1] B. Vermeulen, J. Dielissen, K. Goossen, and C. Ciordas, "Bringing Communication Networks on a Chip: Test and Verification Implications," IEEE Communications Magazine, pp. 74-81, 2003

[2] M. Nahvi and A. Ivanov, "Indirect Test Architecture for SoC Testing," IEEE Trans. on CAD, pp. 1128-1142, 2004

[3] C. Liu, E. Cota, H. Sharif, and D. K. Pradhan, "Test Scheduling for Network-on-Chip with BIST and Precedence Constraints," Proc. IEEE ITC, pp. 1369-1378, 2004

[4] E. Cota et al, "The Impact of NoC Reuse on the Testing of Core-based Systems," Proc. IEEE VTS, pp. 128-133, 2003

[5] E. Cota, L. Carro, F. Wagner, and M. Lubaszewski, "Power-Aware NoC Reuse on the Testing of Core-Based Systems," Proc. IEEE ITC, pp. 612-621, 2003

[6] C. Busch, M. Herlihy, and R. Wattenhofer, "Routing without Flow Control," Proc. the 13th ACM Symposium on Parallel Algorithms and Architectures, pp. 11-20, 2001