AN OPTIMAL FORMULATION FOR TEST SCHEDULING NETWORK-ON-CHIP USING MULTIPLE CLOCK RATES

Hassan Salamy

Ingram School of Engineering
Texas State University
San Marcos, Texas, 78666, USA

Haidar M. Harmanani

Department of Computer Science
Lebanese American University
Byblos, 1401 2010, Lebanon

ABSTRACT

With the growing trend of increasing number of cores on a single chip, bus-based communication is suffering from bandwidth and scalability issues. As a result, the new approach is to use a network on chip (NoC) as the main communication system on a SoC. NoC provides the flexibility and scalability much needed in the era of multi-cores. NoC-based systems also provide the capability of multiple clocking that is widely used in many SoC nowadays. In this paper, an optimal integer linear programming (ILP) solution for test scheduling of cores in a NoC-based SoC using multiple clock rates is presented. Results on different benchmarks show the effectiveness of our techniques.

1. INTRODUCTION

As more power and versatility are needed from a single system-on-chip (SoC), more cores are being utilized. System-on-a-chip (SoC) technology is the packaging of all the necessary electronic circuits and parts for a system on a single integrated circuit, generally known as a microchip. Thanks to recent advances in architecture, VLSI and electronic design, the current trend in modern complex embedded system design is to deploy a multiprocessor system-on-chip (MP-SoC). As more cores are needed, the communication between these cores becomes an important growing problem. Bus communication has been widely accepted as the main communication platform. However, bus communication is no more able to keep up with the increasing complexity of SoC. The bus communication suffers from two main problems, bandwidth and scalability. The new paradigm nowadays is to deploy a network-on-chip (NoC) as the main communication system. An NoC provides an attractive solution to the problems brought forth by increasing complexity and size of system on chip. NoC greatly improves the scalability of SoCs, and provides higher power efficiency in complex SoCs compared to buses. From a system design viewpoint, with the advent of multi-core processor systems, a network is a natural architectural choice.

It is widely recognized that testing embedded cores is a major bottleneck. A major challenge in testing embedded cores is test scheduling which determines the order in which various cores are tested. Test scheduling for SoC, even for a simple SoC, is equivalent to the NP-complete m-processor open shop scheduling problem [1]. An effective test scheduling approach must minimize the test time while addressing resource conflicts among cores arising from the use of shared Test Access Mechanisms (TAMs), on-chip BIST engines, and power dissipation constraints. Obviously, the minimal test time would be achieved by maximizing the simultaneous test of all individual functions or cores; however, design constraints may prevent this full parallelism. For example, power consumption is an important factor that may impact the test parallelism. Power dissipation during testing is a function of time and depends on the switching activity resulting from the application of test vectors to the system. SoC in test mode can dissipate up to twice the amount of power they do in normal mode, since cores that do not normally operate in parallel may be tested concurrently to minimize testing time.

For a system that contains a NoC as the communication platform, the access to cores is already available and thus the idea of reusing the NoC as the communication platform for testing becomes a natural solution. The NoC contains the asset of input/output cores and thus test vectors and the responses can be communicated to and from the cores under test. Such systems usually use a packet based scheduling algorithm where the test vectors will be embedded inside packets. Many NoC have the flexibly of multiple clock rates. Not all the cores have to be tested using the same clock rate. A core with long testing time can be assigned a higher clock rate to reduce the test time.

In this paper, the problem of minimizing test scheduling in a system that contains a NoC using multiple clock rates is studied. The main objective is to minimize the overall test time of the system. Proper allocation of input/output ports in the NoC to cores and the proper allocation of clock rates to different cores are essential in achieving this objective. We present an optimal integer linear programming (ILP) solution to this problem.

The remainder of the paper is organized as follows. Section 2 presents related work in this area. Section 3 presents a brief overview of NoC and description of multiple clock rates. Section 4 presents our ILP formulation. Section 5 summarizes the results. Finally Section 6 presents our conclusions.

2. RELATED WORK

Vermeule et al. [2] introduced the general concept of NoC testing. First, the communication NoC infrastructure should be tested before the built-in core test. Testing communication resources are introduced in [3]. Only after the communication resources are tested, NoC can now be used as a TAM to test the on chip cores. Nahvi et al. [4] proposed network-oriented test architecture to utilize NoC for testability. Generally speaking, there are two main approaches to test scheduling algorithms for NoC-based SoC, namely, core-based and packet-based scheduling. In core-based test scheduling, the scheduler assigns a dedicated routing path for each core to transport test vectors and test responses. The basic idea is to determine the test order of each cores so that the overall test time is minimized under different constraints [5].

On the other hand, packet-based scheduling cores are tested using test packets. The basic idea is to determine the order of gener-

CCECE 2011 Niagara Falls, Canada
978-1-4244-9789-8/11/$26

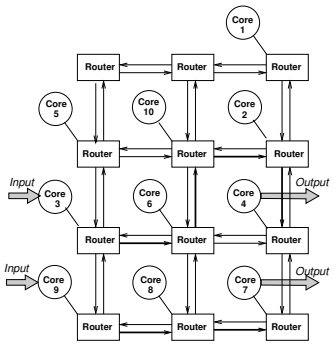

Fig. 1. NoC-based d695 with two routing paths scheduled.

ation and transmission of test packets for cores. The use of packet switching architecture to test cores is proposed in [6] and [7]. Nahvi et al. [6] proposed the use of a packet switching communication-based TAM for system on chip. Cota et al. [8] proposed test scheduling based on a packet-switching protocol with power constraints. Embedded processors have been used for test sources and sinks to increase test parallelism [9]. In [10], [11], multiple test clocks are also used to reduce overall schedule time of NoC-based SoCs.

Liu et. al [5] presented a power-aware testing scheduling algorithm for NoC-based systems. Thermal-ware test scheduling has recently been studied by many researchers [12, 13, 14].

3. NOC TEST SCHEDULING

The new paradigm in the new SoC testability is to use a NoC. NoC usually uses a 2D mesh topology with XY routing and wormhole switching. In a mesh structure, each router is connected to the four neighboring routers. The communication channels are assumed to be 32-bit width and bidirectional. The scheduling is usually based on XY routing where a packet is usually routed in the X direction then in the Y direction till it reaches its destination. XY routing is collision and deadlock free. Switching is usually based on wormhole technique. Each packet is divided into flow control units (flits) where a flit is the smallest unit over which flow control is performed. The flit size is usually equal the channel width. NoC usually uses a message passing communication model where information is sent via a packet. A packet is composed of a header, a payload, and a trailer. Test vectors are delivered through packets. Control information is also delivered in the form of packets in the test header or by assigning specific bits in the payload. Packets are routed from an input port to the core under test and then the results are routed to an output port.

There are two different approaches to go for the test core scheduling namely, preemptive and non-preemptive. In the preemptive approach, the core does not have to be tested till completion at once. Hence packets can be scheduled in an individual manner. Preemptive techniques can increase parallelism as paths are not dedicated to cores for long periods of times. However, preemption is not always an appealing solution especially in the presence of BIST tests. It is usually desirable that the test pipeline of a certain core is not interrupted. This is hard to achieve in preemptive pipelining as such pipeline has to be interrupted if the test vector or test response cannot be scheduled at this time. Such a problem can complicate the wrapper control as well as the overall test time of the schedule.

Our approach is based on a non-preemptive core-based approach where a path is dedicated to a core from the beginning of the test till the end. In such a case, the test pipeline of the core under test is not interrupted as this path is always available for test vectors and test responses. The dedicated routing path usually consists of an input port, an output port, and corresponding channels that transport test vectors to the core and transport the test responses from the core. Even though packet-based scheduling algorithm is more parallel in nature, the test time of its schedules are usually higher than those of a core-based scheduling technique. Figure 1 shows two routing path schedules for cores 8 and 10 for the $d695$ example benchmark using two input and two out ports. The routing paths are built in an XY fashion. Each path will be dedicated to the corresponding core throughout the process of testing. The input/output ports in our example are implemented by using the functional ports on cores 3, 4, 7, and 9.

Huang et. al[10] showed that on NoC, some cores cannot utilize the entire width of the network channels. Those idle channels can be used to transport more test data and increase the clock rate for that specific core. The general idea is to use faster clocks to cores with long test time so that the overall test time is minimized. A faster clock means more channel width is used by the core under test and this sometimes decreases the levels of parallelism between tests due to shortage in channel lines. on the other side, a slower clock increases the test time of a core but frees a percentage of channel width and hence higher levels of parallelism is achievable. Slower clocks can be generated using a frequency divider. To simplify the problem, we assume that there are $2n$ + 1 different on-chip clock frequencies. For example, assume an on-chip clock rate of f and n equals 2 there are a total of 5 clock frequencies, namely, f/3, f/2, f, $2f$, $3f$. The problem is now to assign the on-chip clocking rates to different cores under test. What clock rate should be assigned to what core is a compilcated task and is one of the objectives of this paper. The objective is to minimize the overall test time. Although not studied in this paper, one way to reduce the hot spots is to use slower clock rates.

4. TEST SCHEDULING: OPTIMAL SOLUTION

The problem can now be defined as follows. Given a system on chip with on-chip clock frequency f, n different on-chip clock rates, and m cores to be tested, find the test schedule of all the cores such that the overall test time is minimized. This problem can usually be solved through optimal assignments of the $2n$ +1 clock rates to the m cores and optimal test scheduling of cores to the network input/output and channels. Test Scheduling for SoC is equivalent to the resource constraints problem and is thus NP-complete [15].

In this section, we will present an optimal solution to the problem of test scheduling for network on chip using multiple clock rates. This optimal solution is desirable as the reduction in test schedule time is essential as the number of cores built on a system is substantially increasing. The optimal solution is based on an integer linear formulation to the problem in hand.

Problem Definition: Given a system of N_c cores, N_p input/output ports, and a set of clock rates F_c, map the cores to the input/output ports so that the overall test time is minimized.

First define the following terms for each test core in the system.
T_{iuc} = Test time of core i on input/output pair u under clock frequency c, $1 \leq i \leq N_c$, $1 \leq p \leq N_p$ and $c \in F_c$.
S_i = Start time of core i.
I_{ix} = Input/output pair of core i, $1 \leq x \leq N_p$.

Each core in the system must be assigned to one and only one input/output pair in the system. We can formulate this unity condition by introducing the binary variable I_{ip} defined in Equation (1) with the unity condition formulated in Equation (2).

$$I_i x = \begin{cases} 1, \text{ if variable } i \text{ is assigned to pair x} \\ 0, \text{ otherwise} \end{cases} \quad (1)$$

$$\sum_{x=1}^{N_p} I_{ix} = 1 \qquad \forall\, i \quad (2)$$

As mentioned earlier, we formulate the problem under different clock rates. A higher clock rate usually means lower test time. Each core can be assigned to one clock rate. To take care of this constraint, we define the binary variable F and the constraint in Equations (3) and (4), respectively. The test time for core i on I/O x can now be defined as $\sum_c T_{ixc} \cdot F_{ic}$.

$$F_i c = \begin{cases} 1, \text{ if clock } c \text{ is assigned to core i} \\ 0, \text{ otherwise} \end{cases} \quad (3)$$

$$\sum_c F_{ic} = 1 \qquad \forall\, i \quad (4)$$

As mentioned earlier, each core i will be assigned to one input/output pair p. However, this may present some resource conflicts between two simultaneous cores under tests if there is at least one resource that needs to be used at the same time between the test procedures of both cores. The basic idea for minimizing the schedule test time is to test as many cores in parallel as possible. Two cores cannot be run in parallel if they share some common resources. In other terms, the test live range of two cores cannot overlap if they share some resources. Two cores i and j assigned to input/output pairs x and y, respectively, are said to be tested at the same time if the tests of such cores overlap. The tests of two cores overlap if and only if either one of the following two conditions is valid.

1. $S_{ix} \geq S_{jy} + \sum_c T_{jyc} \cdot F_{jc}$, (or)
2. $S_{jy} \geq S_{ix} + \sum_c T_{ixc'} \cdot F_{ic'}$

Two cores assigned to two input/output pairs in a NoC that require the same resources are not allowed to have an overlapping time according to the two conditions mentioned before. We use a binary variable Z_{ixjy} for cores i and j assigned to pairs x and y to represent conflicts between the tests of cores i and j as follows.

$$Z_{ixjy} = \begin{cases} 1, & \text{if variables } i \text{ mapped to I/O } x \text{ and } j \text{ mapped to I/O } y \\ & \text{are conflicting} \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

The constraint to prevent conflicting cores from being scheduled at the same time can now be formulated using the binary variable Z_{ixjy} as in Equations (6 & 7).

$$Z_{ixjy}(S_{ix} - S_{jy} - \sum_c T_{jyc} \cdot F_{jc}) \geq 0. \text{or} \quad (6)$$

$$Z_{ixjy}(S_{jy} - S_{ix} - \sum_c T_{ixc'} \cdot F_{ic'}) \quad (7)$$

where $1 \leq$ i,j $\leq N_c$, $1 \leq$ x,y $\leq N_p$ and $c/c' \in F_c$

Clearly, the constraints in Equations (6 & 7) are not linear. To linearize, we first introduce the binary variables K_{ixjy1} and K_{ixjy2} such that $K_{ixjy1} + K_{ixjy2} = 1$. Then, the constraints in Equations (6 & 7) can be written as the constraint in Equation (8). Equation (8), however, is still non linear due to the multiplication of two variables K and S. In order to transform it to a linear constraint, we replace $K_{ixjy1} \cdot S_{ix}$ by new binary variables M_{ixjy1} that takes a value of 1 if K_{ixjy1}= 1 and S_{ix} = 1. This can be enforced by adding 3 additional constraints as shown in Equations (9-11). Following the same procedure, we replace $K_{ixjy1} \cdot S_{iy}$, $K_{ixjy2} \cdot S_{jy}$, and $K_{ixjy2} \cdot S_{ix}$ by new binary variables M_{ixjy2}, M_{ixjy3}, and M_{ixjy4}, respectively and add nine additional constraints similar to those in Equations (9-11).

$$Z_{ixjy} \cdot K_{ixjy1}(S_{ix} - S_{jy} - \sum_c T_{jyc} \cdot F_{jc}) + Z_{ixjy} \cdot K_{ixjy2}(S_{jy} - S_{ix} - \sum_c T_{ixc'} \cdot F_{ic'}) \geq 0 \quad (8)$$

$$M_{ixjy1} \leq K_{ixjy1} \quad (9)$$

$$M_{ixjy1} \leq S_{ix} \quad (10)$$

$$M_{ixjy1} \geq K_{ixjy1} + S_{ix} - 1 \quad (11)$$

The objective function is to decrease the overall test time of a certain SoC and thus it can be defined as in the equation below.

$$\text{Minimize C} : Max(S_{ix} + \sum_c T_{ixc} \cdot F_{ic}) \qquad \forall \text{ cores i, x, c} \quad (12)$$

Clearly, as it is, the objective function is not linear and thus we resort to linearization. We linearize C by minimizing C and adding set of constraints in Equation (13). However, the constraint in Equation (13) is still not linear. To linearize, we replace $I_{ix}S_{ix}$ by a new binary variable R_{ix} and add the three additional constraints in Equations (14-16).

$$C \geq \sum_{x=l}^{N_p} I_{ix}(S_{ix} + \sum_c T_{ixc} \cdot F_{ic}) \qquad 1 \leq i \geq N_c \quad (13)$$

$$R_{ix} \leq I_{ix} \quad (14)$$

$$R_{ix} \leq S_{ix} \quad (15)$$

$$R_{ix} \geq I_{ix} + S_{ix} - 1 \quad (16)$$

The ILP formulation presented in this section provides the optimal solution to our problem with the drawback of expensive computational time. One way to minimize the computation time is through enumeration [16]. One possible enumeration is through the predetermined assignment of cores to I/Os as in Figure 2. Enumeration will significantly reduce the computation time of our ILP as the presented model is substantially simplified. All the possible assignments are then enumerated to find the optimal solution. Please note that the enumeration method still provides the optimal solution since the simplified ILP model is solved for all possible I/Os assignments.

Test Scheduling ILP_Enumeration()
FOR all possible I/O assignments:
Begin
Find ILP solution S:
$C \geq S_i + \sum_c T_{ixc} \cdot F_{ic} \qquad 1 \leq i \geq N_c.$
$Z_{ixjy} \cdot K_{ixjy1}(S_{ix} - S_{jy} - \sum_c T_{jyc} \cdot F_{jc}) +$
$Z_{ixjy} \cdot K_{ixjy1}(S_{jy} - S_{ix} - \sum_c T_{ixc} \cdot F_{ic}) \geq 0.$
$K_{ixjy1} + K_{ixjy2} = 1$
End FOR
Optimal Solution = Min (S)

Fig. 2. ILP model using enumeration.

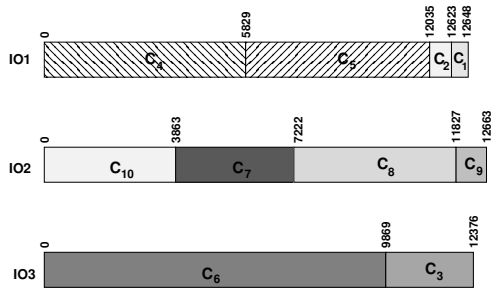

Fig. 3. Scheduling results of d695 with single clock rate using ILP

5. RESULTS

In this section, we present test scheduling results for three benchmarks from ITC'02 [17] namely, $f2126$, $a586710$, and $d695$. The simulations were performed on a dual core system of 3 GHz processor. We created a hypothetical layout for each NoC system.

We tested our ILP formulation on $f2126$, $a586710$, and $d695$ benchmarks consisting of 4, 7 and 10 cores, respectively. We present test data statistics in details for the $d695$ benchmark in Table 1. Column 1 shows the core number whereas the second column shows the number of test packets which is twice the number of test vectors, one for the test vector and one of the test responses. The third column shows the packet size which equals the sum of the scan chains lengths and the output ports. The last two columns show the test time in cycles under channel width of 32 and 16, respectively. Notice that for cores 3, 4 and 8 in Table 1, the test times for W =32 and W = 16 are the same. This is because the number of scan chains combined with the functional inputs and outputs of these cores is smaller than 16.

Table 1. Test statistics for d695 benchmark.

Cores	# test packets	Packet size	Test Cycles W=32	Test Cycles W=16
1	24	32	25	38
2	146	108	588	1029
3	150	33	2507	2507
4	210	250	5829	5829
5	220	1730	6206	12192
6	468	790	9869	11978
7	190	684	3359	4219
8	194	228	4605	4605
9	24	2048	836	1659
10	136	1742	3863	7586

First we tested our ILP formulation assuming there is only one clock. Results are shown in Table 2. The channel width is assumed to be 32. Figure 3 shows the detailed results of the $d695$ assuming single clock rate with three input/output ports. The test time in this situation is 12663 cycles.

Then we repeated our experiments using multiple clock rates. We assume that there are three clock rates $f/2$, f, $2f$. The test times for channel width of 32 and 16 are shown in Table 3. Single CLK represents the results under single test clock mode. Case 1 represents the case where we assign $2f$ clock rate to core i if $T_i(W) = T_i(W/2)$. In case 2, the test clock rate of f or $2f$ is chosen according to the TAM width available. We list in brackets the results achieved in [5] for comparison. The results clearly show the importance of using multiple clock rates especially to decrease the test time for long test time cores.

Table 2. Optimal test times with single clock rate.

Benchmark	I/O	Test time
f2126	2/2	382446
a586710	2/2	11986249
d695	3/3	12663

Table 3. Optimal test times for the d695 SoC benchmark under different conditions.

Case	W = 32 Ours ([5])	W = 16) Ours ([5])
Single CLK	12663 (13228)	17581 (17807)
Case 1	10457 (10705)	15524 (15787)
Case 2	8474 (8534)	15145 (15369)

6. CONCLUSION

In this paper, we presented an optimal ILP solution to test scheduling for NoC under multiple clock rates. Results on different benchmarks show the importance of our techniques especially as the number of cores built on a single chip is continuously increasing.

7. REFERENCES

[1] K. Chakrabarty, "Test scheduling for core-based systems using mixed-integer linear programming," *IEEE Trans. on CAD*, pp. 1163–1174, 2000.

[2] B. Vermeulen, J. Dielissen, K. Goossens, and C. Ciordas, "Bringing communication networks on a chip: Test and verification implications," *IEEE Communication Magazine*, vol. 41, pp. 74–81, 2003.

[3] P. P. Pande, G. D. Micheli, C. Grecu, A. Ivanov, and R. Saleh, "Design, synthesis, and test of networks on chips," *IEEE Design and Test of Computers*, pp. 404–413, 2005.

[4] M. Nahvi and A. Ivanov, "Indirect test architecture for soc testing," *IEEE Trans. on CAD*, pp. 1128–1142, 2004.

[5] C. Liu, V. Iyengar, J. Shi, and E. Cota, "Power-aware test scheduling in network-on-chip using variable-rate on-chip clocking," in *Proc. VLSI Test Symp.*, 2005, pp. 349–354.

[6] M. Nahvi and A. Ivanov, "A packet switching communication-based test access mechanism for system chips," *IEEE European Test Workshop*, 2001.

[7] C. Aktouf, "A complete strategy for testing an on-chip multiprocessor architecture," *IEEE Design & Test of Computers*, vol. 19(1), pp. 18–28, 2002.

[8] E. Cota, L. Carro, F. Wagner, and M. Lubaszewski, "Power-aware noc reuse on the testing of core-based systems," in *Proceedings ITC, pp. 612–621*, 2003.

[9] A. M. Amory, E. Cota, F. Wagner, L. Carro, M. Lubaszewski, and F. G. Moraes, "Reducing test time with processor reuse in network-on-chip based system," in *Proceedings SBCCI, pp. 111–116*, 2004.

[10] J. Ahn and S. Kang, "Noc-based soc test scheduling using ant colony optimization," *ETRI Journal*, vol. 30, pp. 129–140, 2008.

[11] J. Shao et. al, "Multi-clock domain soc test scheduling based on ant colony optimization algorithm," in *Proc. Fourth International Conference on Internet Computing for Science and Engineering*, 2009.

[12] P. Rosinger, B. Al-Hashimi, and K. Chakrabarty, "Rapid generation of thermal-safe test schedules," in *Proc. Design, Automation and Test in Europe (DATE) Conf.*, 2005, pp. 840–845.

[13] E. Tafaj, P. Rosinger, B. Al-Hashimi, and K. Chakrabarty, "Improving thermal-safe test scheduling for corebased system-on-chip using shift frequency scaling," in *Proc. Int. Symp. DFT*, 2005, pp. 544–551.

[14] Chunsheng Liu and Vikram Iyengar, "Test scheduling with thermal optimization for network-on-chip systems using variabe-rate on-chip clocking," in *Proc. Design, Automation and Test in Europe (DATE) Conf.*, 2006.

[15] C. Liu, E. Cota, H. Sharif, and D. K. Pradhan, "Test scheduling for network-on-chip with bist and precedence constraints," in *Proceedings ITC, pp. 1369–1378*, 2004.

[16] V. Iyengar, K. Chakrabarty, and E. J. Marinissen, "On using rectangle packing for soc wrapper/tam co-optimization," in *Proceedings VTS, pp. 253–258*, 2002.

[17] E. J. Marinissen, V. Iyengar, and K. Chakrabarty, "A set of benchmarks for modular testing of socs," in *Proc. Int. Test Conference (ITC)*, 2002, pp. 519–528.