International Journal of Electronics
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tetn20

Thermal-aware test scheduling using network-on-chip under multiple clock rates

Hassan Salamy [a] & Haidar M. Harmanani [b]
[a] Ingram School of Engineering , Texas State University , San Marcos , TX , USA
[b] Department of Computer Science , Lebanese American University , Byblos , Lebanon
Published online: 06 Sep 2012.

To cite this article: Hassan Salamy & Haidar M. Harmanani (2013) Thermal-aware test scheduling using network-on-chip under multiple clock rates, International Journal of Electronics, 100:3, 408-424, DOI: 10.1080/00207217.2012.713016

To link to this article: http://dx.doi.org/10.1080/00207217.2012.713016

International Journal of Electronics, 2013
Vol. 100, No. 3, 408–424, http://dx.doi.org/10.1080/00207217.2012.713016

Thermal-aware test scheduling using network-on-chip under multiple clock rates

Hassan Salamy[a] and Haidar M. Harmanani[b]*

[a]*Ingram School of Engineering, Texas State University, San Marcos, TX, USA;* [b]*Department of Computer Science, Lebanese American University, Byblos, Lebanon*

(*Received 22 August 2011; final version received 1 May 2012*)

The increasing trend in the number of cores on a single chip has led to scalability and bandwidth issues in bus-based communication. Network-on-chip (NoC) techniques have emerged as a solution that provides a much needed flexibility and scalability in the era of multi-cores. This article presents an optimal integer linear programming (ILP) formulation and a simulated annealing (SA) solution to thermal and power-aware test scheduling of cores in an NoC-based SoC using multiple clock rates. The methods have been implemented and results on various benchmarks are presented.

Keywords: network-on-chip; test scheduling; SoC; ILP; simulated annealing

1. Introduction

As more power and versatility are needed from a single system-on-chip (SoC), more cores are being utilised. 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, Very Large Scale Integration (VLSI) and electronic design, the current trend in modern complex embedded system design is to deploy a multiprocessor system-on-chip (MPSoC). As more cores are being deployed, communication in such systems 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. 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 the size SoC. NoC greatly improves the scalability of SoCs and it provides higher levels of power efficiency in complex SoCs compared to buses.

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 (Chakrabarty Chakrabarty 2000). An effective test-scheduling approach must minimise the test time while addressing resource conflicts among cores arising from the use of shared test access mechanisms (TAMs), on-chip Built-In Self-Test (BIST) engines and power dissipation constraints. Obviously, the minimal test time would be achieved by maximising the

*Corresponding author. Email: haidar.harmanani@lau.edu.lb

simultaneous tests of individual functions or cores; however, design constraints may prevent this full parallelism. For example, power consumption is an important factor that may impact 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 the normal mode. This is usually because cores that do not normally operate in parallel may be tested concurrently to minimise testing time. However, the Rosinger, Al-Hashimi, and Chakrabarty (2005) showed that power constraints do not necessarily achieve thermal safety of each core in the system. Thermal-aware test scheduling is therefore essential in order to ensure that the maximum thermal safety temperature is not exceeded by any core.

For a system that contains an 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. 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 packet-based scheduling algorithms where test vectors will be embedded inside packets. Many NoCs 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. Increasing clock rates increases the power consumption of the system cores. What clock rate should be assigned to what core to minimise the overall test time and such that the safety power and thermal constraints are met is one of the main concerns of this article.

In this article, the problem of minimising test scheduling in a system that contains an NoC using multiple clock rates is studied. Power and thermal constraints are set for the safety and protection of the system under test. The main objective is to minimise 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 for small instances and a simulated annealing (SA) for larger instances. To the best of our knowledge, this is the first optimal and suboptimal solution to thermal-aware test scheduling using NoC with multiple clock rates.

This article is organised as follows. Section 2 presents the related work in this area. Section 3 presents a brief overview of NoC. Section 4 deals with test scheduling using multiple clock rates. Section 5 presents our ILP formulation. Section 6 presents the SA solution for the test-scheduling problem in an NoC using multiple clock rates. Section 7 summarises the results. Finally, Section 8 presents our conclusions.

2. Related work

Various researchers have studied the NoC test-scheduling problem. Vermeulen, Dielissen, Goossens, and Ciordas (2003) introduced the general concept of NoC testing where the communication NoC infrastructure should be tested before testing the built-in cores. Testing communication resources are introduced in Pande, Micheli, Grecu, Ivanov, and Saleh (2005). Only after the communication resources are tested, NoC can now be used as a TAM to test the on chip cores. Generally speaking, there are two main approaches to test-scheduling algorithms for NoC-based SoC, namely, core-based and packet-based scheduling. The basic idea in core-based scheduling is to determine the test order of each core so that the overall test time is minimised under different constraints (Liu, Iyengar,

Shi, and Cota 2005). On the other hand, packet-based scheduling cores are tested using test packets. The use of packet-switching architecture to test cores is proposed in Nahvi and Ivanov (2001). Cota et al. (2003b) proposed test scheduling based on a packet-switching protocol. Test vectors and test responses for each core are represented as packets to be transmitted throughout the network. The main idea is to be able to schedule all the packets in a way to minimise the total test time through parallelism. This algorithm was further improved and power constraints were included in Cota, Carro, Wagner, and Lubaszewski (2003a). In Ahn and Kang (2006), multiple test clocks are used to the reduce the overall schedule time of NoC-based SoCs.

Power-aware testing scheduling for NoC-based systems have been studied by many researchers, such as Cota, Carro, and Lubaszewski (2004) and Liu et al. (2005). However, it has been shown in Liu et al. (2005) and Rosinger et al. (2005) that power-aware constraints do not guarantee thermal safety. Thermal-ware test scheduling has been studied by many researchers, such as Liu et al. (2005), Rosinger et al. (2005), Tafaj, Rosinger, Al-Hashimi, and Chakrabarty (2005) and Liu and Iyengar (2006). In Rosinger et al. (2005), a fast heuristic is used that guarantees the thermal safety. A test session thermal model is used to determine heat transfer during the test. In Tafaj et al. (2005), variable test clock frequencies are used to control the power consumption by each core so that the thermal safety is guaranteed. In Liu et al. (2005), layout information and progressive weighting are used to reduce hot spot temperatures. Liu and Iyengar (2006) presented two heuristics to achieve thermal optimisation based on multiple clock frequencies. However, we believe that an optimal or close to optimal solutions are essential in minimising the overall test time under thermal safety and that the overhead in execution time can be tolerated, thus the techniques in this article. The results in this article can also be used as a metric of how close heuristics results are from the optimal solution.

Others like He, Peng, and Eles (2010), Yao, Saluja, and Ramanathan (2011) and Aghaee, He, Peng, and Eles (2010) dealt with issues related to thermal-aware safety test scheduling.

3. NoC test scheduling

Testing embedded cores is a challenging problem since test data cannot be transported through a dedicated TAM. This would lead to difficulty of routing due to the fact that the network (routers, channels, etc.) already imposes significant routing overhead (Wang, Stroud, and Touba 2007). Core-based designs are usually tested after assembly, at the end of the system implementation. Typically, the core test problem is solved by surrounding a hard core with test logic, known as a test-wrapper or a test collar. During external test mode, the wrapper element drives the host chip in order to test the interconnect while during internal test mode the wrapper element tests the core by observing the core output. Bus-based systems are suffering from bandwidth and scalability issues with the advent of multi-cores. The new paradigm in the new SoC testability is to use an NoC. The NoC usually uses a $2D$ mesh topology with XY routing and wormhole switching. In a mesh structure, each router is connected to the four neighbouring routers. The communication channels are assumed to be of 32-bit or 16-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 the wormhole technique. 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 packet-based scheduling algorithms where test vectors will be embedded inside packets. NoCs usually use a message-passing communication model where information is sent via packets. 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 out through an output port.

There are two different approaches to go for test core scheduling, namely preemptive and non-preemptive. In the preemptive approach, the core need not 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 period 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 increase 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 the 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

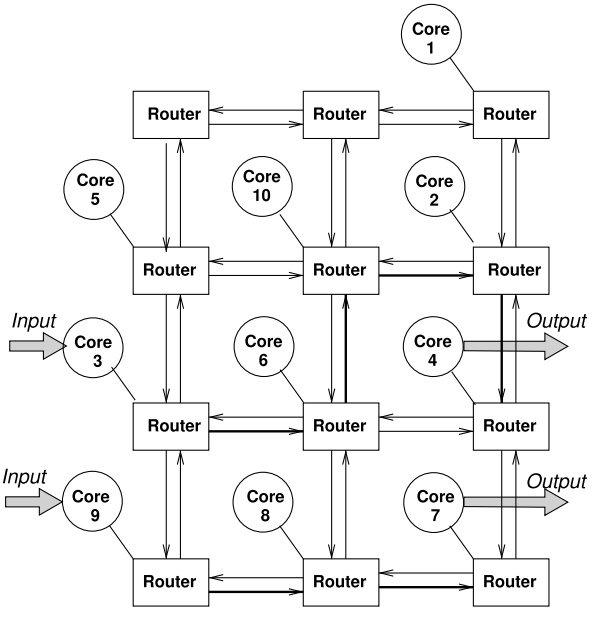

Figure 1. NoC-based *d*695 with two routing paths scheduled.

cores 8 and 10 for the *d*695 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 using the functional ports on cores 3, 4, 7 and 9.

4. Test scheduling using multiple clock rates

Cores in the SoC under consideration usually use test wrappers to adapt to the TAM. The core wrapper affects the parallelisation of tests and thus the testing times. Since some cores cannot utilise all the channel width, a channel can be used by multiple cores simultaneously. The idle channels can also be used to generate faster clock rates. The faster clock can be generated using an on-chip PLL whereas slower clocks can be generated using the frequency divider. Typically, excessive power dissipation and inadequate heat convection/conduction can cause some cores to be significantly hotter than others, the so-called *hot spots*. Applying the entire test suite continuously can lead to dangerous temperature on these cores (Wang et al. 2007).

In this article, we propose the use of multiple clock rates for thermal management and test time minimisation. One way to reduce the *hot spots* is to use a slower clock rate. Accurate core temperature is usually best calculated using thermal simulation. In order to obtain the temperature of a core during test, we use a computationally efficient thermal modelling tool called HotSpot (Huang, Ghosh, Sankaranarayanan, Skadron, and Stan 2006). HotSpot accounts for heat dissipation within each functional block as well as the heat flow among different blocks. The thermal transfer resistance of R_{ij} of IP block PE_i with respect to PE_j is defined as the temperature rise at PE_i due to one unit of power dissipated at PE_j as the following:

$$R_{ij} = \frac{\Delta T_{ij}}{\Delta P_j} \tag{1}$$

Using a single slow clock rate is usually inefficient as it may result in a higher testing time for the cores. Hence, a single slower clock adversely affects the test time and increases the cost. Although slower clock rates can bring down the hot spots, it is not efficient in achieving thermal balance across the chip. As a result, we propose to use multiple clock rates instead of one slow clock. Generally speaking, a slow clock is assigned to hot spots to cool them down. The problem is not that simple. As mentioned before, a slow clock rate usually means a higher test time. The problem is to find the best clock rates for different spots and a test schedule to minimise the test time of all the cores such that the core temperatures are below a certain threshold for thermal safety.

In NoC, some cores cannot utilise 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 slower clock for high spot cores and faster clocks to cool cores so that the thermal safety constraints are met and the overall test time is minimised. 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 3, there are a total of seven clock frequencies, namely $f/4$, $f/3$, $f/2$, f, $2f$, $3f$, $4f$. The problem is now to assign the on-chip clocking rates to different cores under test. The objective is to minimise the overall test time while not exceeding the maximum core temperature allowed for safety issues.

5. Test scheduling: ILP formulation

The problem can now be defined as follows. Given an SoC with an 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 minimised. This problem can usually be solved through optimal assignment of the $2n+1$ clock rates to the m cores and optimal test scheduling of cores to the network input/output ports and channels. Test scheduling for SoC is equivalent to the resource constraints problem and is thus NP-complete (Liu, Cota, Sharif, and Pradhan 2004).

In this section, we will present a solution based on ILP to the problem of test scheduling for NoC 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 increasing.

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 minimised.

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 the clock frequency c, $1 \le i \le N_c$, $1 \le u \le N_p$ and $c \in F_c$.

S_i = Start time of core i.

I_{ix} = Input/output pair of core i, $1 \le x \le N_p$.

Each core in the system must be assigned to one and only one input/output pair. We can formulate this unity condition by introducing the binary variable I_{ip} defined in Equation (2) with the unity condition formulated in Equation (3).

$$I_i x = \begin{cases} 1, & \text{if variable } i \text{ is assigned to pair } x \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

$$\sum_{x=1}^{N_p} I_{ix} = 1 \quad \forall i \tag{3}$$

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 constraints in Equations (4) and (5), respectively. The test time for core i on I/O x can now be defined as $\sum_c T_{ixc} \cdot F_{ic}$.

$$F_i c = \begin{cases} 1, & \text{if clock } c \text{ is assigned to core } i. \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

$$\sum_c F_{ic} = 1 \quad \forall i \tag{5}$$

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 minimising 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 ranges 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 an 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:

$$Z_{ixjy} = \begin{cases} 1, & \text{if variables } i \text{ mapped to I/O } x \text{ and } j \\ & \quad \text{mapped to I/O } y \text{ are conflicting} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

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 (7) and (8).

$$Z_{ixjy}\left(S_{ix} - S_{jy} - \sum_c T_{jyc} \cdot F_{jc}\right) \geq 0. \quad \text{or} \tag{7}$$

$$Z_{ixjy}\left(S_{jy} - S_{ix} - \sum_{c'} T_{ixc'} \cdot F_{ic'}\right) \geq 0 \tag{8}$$

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 (7) and (8) are not linear. To linearise, we first introduce the binary variables K_{ixjy1} and K_{ixjy2} such that $K_{ixjy1} + K_{ixjy2} = 1$. Then the constraints in Equations (7) and (8) can be written as the constraint in Equation (9). However, Equation (9) is still nonlinear due to the multiplication of the 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 three additional constraints, as shown in Equations (10)–(12). 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, as in Equation (13) and add nine additional constraints similar to those in Equations (10)–(12). The same technique will be used to linearise the product of the variables K and F in Equation (9).

$$Z_{ixjy} \cdot K_{ixjy1}\left(S_{ix} - S_{jy} - \sum_c T_{jyc} \cdot F_{jc}\right) + Z_{ixjy} \cdot K_{ixjy2}\left(S_{jy} - S_{ix} - \sum_{c'} T_{ixc'} \cdot F_{ic'}\right) \geq 0 \tag{9}$$

$$M_{ixjy1} \leq K_{ixjy1} \tag{10}$$

$$M_{ixjy1} \leq S_{ix} \tag{11}$$

$$M_{ixjy1} \geq K_{ixjy1} + S_{ix} - 1 \tag{12}$$

$$Z_{ixjy}\left(M_{ixjy1}-M_{ixjy2}-K_{ixjy1}\sum_{c}T_{jyc}\cdot F_{jc}\right)+Z_{ixjy}\left(M_{ixjy3}-M_{ixjy4}-K_{ixjy2}\sum_{c'}T_{jyc'}\cdot F_{jc'}\right)\geq 0 \tag{13}$$

The objective function is to decrease the overall test time of a certain SoC and it is defined as follows:

$$\text{Minimise C}:\text{Max}\left(S_{ix}+\sum_{c}T_{ixc}\cdot F_{ic}\right)\quad \forall\text{cores } i,x,c \tag{14}$$

Clearly, as is, the objective function is not linear, and thus we resort to linearisation. We linearise C by minimising C and adding a set of constraints in Equation (15). However, the constraints in Equation (15) are still nonlinear. To linearise, we replace $I_{ix}S_{ix}$ by a new binary variable R_{ix} as in Equation (19) and add the three additional constraints in Equations (16)–(18). The same technique will be used to linearise the product of the variables I and F in Equation (15).

$$C\geq\sum_{x=l}^{N_p}I_{ix}\left(S_{ix}+\sum_{c}T_{ixc}\cdot F_{ic}\right)\quad 1\leq i\geq N_c \tag{15}$$

$$R_{ix}\leq I_{ix} \tag{16}$$

$$R_{ix}\leq S_{ix} \tag{17}$$

$$R_{ix}\geq I_{ix}+S_{ix}-1 \tag{18}$$

$$C\geq\sum_{x=l}^{N_p}\left(R_{ix}+I_{ix}\sum_{c}T_{ixc}\cdot F_{ic}\right)\quad 1\leq i\geq N_c \tag{19}$$

The ILP formulation presented in this section provides the optimal solution to our problem with the drawback of expensive computational time since the number of variables in our ILP formulation is proportional to $n^2/2\cdot p^2$. One way to minimise the computation time is through enumeration (Iyengar, Chakrabarty, and Marinissen 2002). One possible enumeration is through the predetermined assignment of cores to I/Os (Figure 2), where all the possible assignments are enumerated in order to find the optimal solution.

```
Test Scheduling ILP_Enumeration()
{
   FOR all possible I/O assignments:
   Begin
      Find ILP solution S:
      C ≥ S_ix + Σ_c T_ixc · F_ic                    1 ≤ i ≥ N_c.
      Z_ixjy · K_ixjy1(S_ix − S_jy − Σ_c T_jyc · F_jc)+
            Z_ixjy · K_ixjy2(S_jy − S_ix − Σ_c' T_ixc' · F_ic') ≥ 0.
      K_ixjy1 + K_ixjy2 = 1
   End FOR
   Optimal Solution = Min (S)
}
```

Figure 2. ILP model using enumeration.

Enumeration will significantly reduce the computation time of our ILP as the presented model is substantially simplified. It should be noted that the enumeration method provides the optimal solution since the simplified ILP model is solved for all possible I/Os assignment; however, it is only computational efficient for small to medium size instances.

6. Test scheduling: SA

The ILP formulation guarantees optimal solutions, but is computationally inefficient. Although ILP enumeration reduces the computational time, it is still quite expensive for large instances. Furthermore, the inclusion of thermal constraints in our ILP is not straight forward and is quite expensive. In order to get near-optimal solutions for large benchmarks in a reasonable amount of time, we used an SA (Figure 3).

SA (Kirkpatrick and Vecchi 1983) is a global stochastic method used to generate approximate solutions to very large combinatorial problems. The technique originates from the theory of statistical mechanics and is based on the analogy between the annealing process of solids and the solution procedure for large combinatorial optimisation problems. The annealing algorithm begins with an initial feasible configuration, and then a


```
Thermal-Aware SA_Test_Scheduling()
  N_iter = The number of Iteration.
  Time_stop = Stopping temperature .
  K = Temperature reduction ratio.
  S_init = Initial Solution.
  T_thermal_safety = Maximum allowable temperature.
  P_max = Maximum allowable power consumption at the same time.
  Boolean Power, Temperature.
  Build S_init.
  Invoke HOTSPOT(Power File, Floorplan File).
  Cost(S_init) = Test time for S_init.
  Current solution S_curr = S_init.
  While (Time_curr > Time_stop)
    For ( i = 1 to N_iter) do
      Generate a neighboring solution S_new = Neighbour (S_curr)
      Evaluate S_new.
      If (S_new is feasible) then
        If (Power)
          For each core i do
            Find all cores j such that.
             t_j ≤ t_i AND t_j + l_j > t_j /*Overlap .
            Calculate the sum of power consumption of such cores.
            Find maximum power consumption,P, of all overlapping cores.
            If P > P_max then Violation.
        If (Temperature)
          Update temperature file based on current configuration.
          Find core i with maximum temperature T.
           If T > T_thermal_safety then Violation.
        If (No Violation) then
          Find test time Cost(S_New)
          Compute Δ_Cost = CostS_new - CostS_curr
          If (Δ_Cost) < 0) then
            S_curr = S_new.
          Else
            Generate random number R, 0 < R < 1.
            If R < exp(-Δ_Cost/T_curr) then
              S_curr = S_new.
    Update SA parameters .
```

Figure 3. Our thermal-aware SA for test scheduling.

neighbour configuration is created by perturbing the current solution. If the cost of the neighbouring solution is less than that of the current solution, the neighbouring solution is accepted; otherwise, it is accepted or rejected with some probability. The probability of accepting inferior solutions is a function of a parameter, called the temperature T, and the change in cost between the neighbouring solution and the current solution. The temperature is decreased during the optimisation process, and the probability of accepting an inferior solution decreases with the reduction of the temperature value.

The set of parameters controlling the initial temperature, stopping criterion, temperature decrement between successive stages and the number of iterations for each temperature is called the *cooling schedule*. Typically, at the beginning of the algorithm, the temperature T is large and an inferior solution has a high probability of being accepted. During this period, the algorithm acts as a random search to find a promising region in the solution space. As the optimisation progresses, the temperature decreases and there is a lower probability of accepting an inferior solution. The algorithm then behaves like a downhill algorithm for finding the local optimum of the current region.

6.1. *Solution representation and initial solution*

An example of a random feasible solution to our problem and its corresponding test schedule is presented in Figure 4(a). This example solution shows a system of four cores, two I/O ports and two possible clock rates, f_1 and f_2. The initial solution is chosen to be a random feasible assignments of cores to I/O ports for a given set of clock rates.

For simplicity in Figure 4, we assume that each core utilises the whole channel width. This, however, is not required as our SA allows more than one core to share the same channel if a sufficient bandwidth is available. A core cannot be scheduled on a blocked path. A blocked path is such that the cores $C_1 \cdots C_i$ mapped to that path utilise the whole bandwidth in at least one link or needed port. In other words, for a certain link on the path, $W(Channel) = W(C_1) + W(C_2) + \cdots + W(C_i)$ where W is the width used by a certain core.

6.2. *Neighbourhood transformation*

The main operation of the SA approach is the neighbourhood function. Starting from a current solution, the neighbourhood function applies some operations to move into

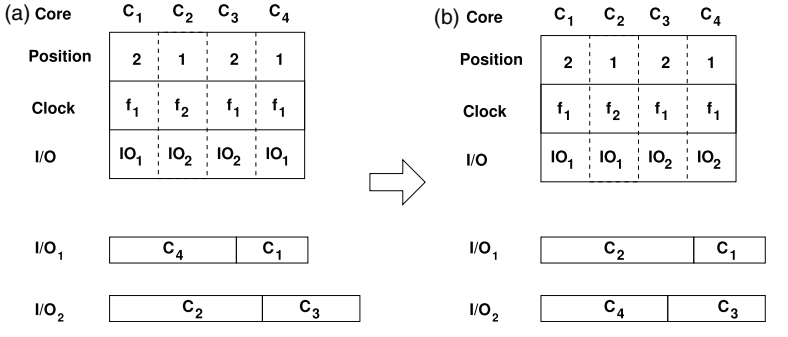

Figure 4. (a) An example the SA solution and its test schedule. (b) The new solution and its test schedule.

a new solution. The neighbourhood function in our proposed SA can perform one of the following operations.

- Change the position of one core.
- Swap the positions of two cores.
- Change the clock of one core.
- Change the I/O of one core.
- Swap the I/O of two cores.

Figure 4 shows an example of swapping the I/O of cores 2 and 4 and the corresponding schedules for the solution before and after applying such an operation. Prior to performing the operation, core 2 (C2) is mapped to input/output IO2 whereas core 4 (C4) is mapped to input/output IO1. After performing the neighbourhood function, input/output ports IO1 and IO2 are mapped to cores C2 and C4, respectively, as is shown in the schedule in Figure 4(b).

6.3. *Cost function and cooling schedule*

Given a test schedule, the cost is the maximum end time for the tests of all the cores in our system. The cooling schedule is the set of parameters controlling the initial temperature, the stopping criterion, the temperature decrement between successive stages and the number of iterations for each temperature. The cooling schedule was empirically determined as follows:

(1) The initial temperature $T_{init} = 4000$. This temperature must be high enough to allow moves to almost neighbourhood state; otherwise, we are in danger of implementing hill climbing) and must not be so hot that we conduct a random search for a period of time.
(2) The temperature reduction multiplier, α, is set to 0.99. Experience in this area has shown that α should be between 0.8 and 0.99, with better results being found in the higher end of the range. Of course, the higher the value of α, the longer it will take to decrease the temperature to the stopping criterion.
(3) The number of iterations, M, is set to 5 while the iteration multiplier, β, is set to 1.5.
(4) The algorithm stops when the current temperature, T, is below 0.001. In practice, in our approach it is not necessary to let the temperature reach zero, as best solutions are reached at higher temperatures in most cases and hence the run time of our SA is reduced.

7. Results

In this section, we present test-scheduling results for six benchmarks from ITC'02 (Marinissen, Iyengar, and Chakrabarty 2002), namely, *f*2126, *a*586710, *d*695, *g*1023, *p*22810 and *p*93791. The simulations were performed on a dual core system of 3 GHz processor. We created a hypothetical layout for each NoC system. As the power information of the cores in these benchmarks is not available, approximation values based on the number of scan flip-flops of each core is used. Based on these approximated values, the power consumption of each core defined in Equation (20) is adjusted by replacing n_{ff} by the number of scan flip-flops and n_{gt} by an estimated number of gates for the module.

In Equation (20), C_L is the load capacitance, T is the clock period and σ is the switching activity factor.

$$Power = C_L \cdot V_{dd}^2 \cdot \frac{1}{T} \cdot [(\sigma_{ff} + 1) \cdot nb_{ff} + \sigma_{gt} \cdot nb_{gt}] \tag{20}$$

7.1. *Experiment set 1*

In this subsection, we present test-scheduling results using our ILP formulation (using CPLEX; ILOG Inc.) and for three benchmarks from ITC'02 (Marinissen et al. 2002), namely, *f*2126, *a*586710 and *d*695 consisting of 4, 7 and 10 cores, respectively. We present detailed test data statistics for the *d*695 benchmark in Table 1. First column shows the core number and 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 is equal to 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.

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 5 shows the detailed results of the *d*695 assuming single clock rate with three input/output ports. The test time

Table 1. Test statistics for *d*695 benchmark for $W = 32$ and $W = 16$.

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	12,192
6	468	790	9869	11,978
7	190	684	3359	4219
8	194	228	4605	4605
9	24	2048	836	1659
10	136	1742	3863	7586

Table 2. Optimal test times for SoC benchmarks with a single clock rate.

Benchmark	I/O	Test time
*f*2126	2/2	382,446
*a*586710	2/2	11,986,249
*d*695	3/3	12,663

in this situation is 12,663 cycles compared to 13,228 cycles achieved using the techniques by Liu et al. (2005).

Then, we repeated our experiments using different clock rate scenarios. We assume that there are three clock rates $f/2$, f and $2f$. The test times for channel width of 32 and 16 are shown in Table 3. Single CLK represents the results under a single test clock mode. Case 1 represents the case where the $2f$ clock rate can only be assigned to core i if $T_i(W) = T_i(W/2)$. In case 2, the test clock rates of f or $2f$ can be chosen according to the TAM width available. We list in brackets the results achieved in Liu et al. (2005) for comparison. The results clearly show the importance of using multiple clock rates, especially to decrease the test time for cores with long test times. Our ILP technique can optimally solve small to medium benchmarks within few seconds to 20 min. Optimal solutions to large benchmarks is quite expensive, and hence we use our SA approach in the next subsection.

7.2. *Experiment set 2*

In this set of experiments, we tested the benchmarks using the SA approach presented in Section 6. We tested our SA on the following benchmarks, *d*695, *g*1023, *p*22810 and *p*93791. It should be noted that in most of the cases, best solutions were found way before

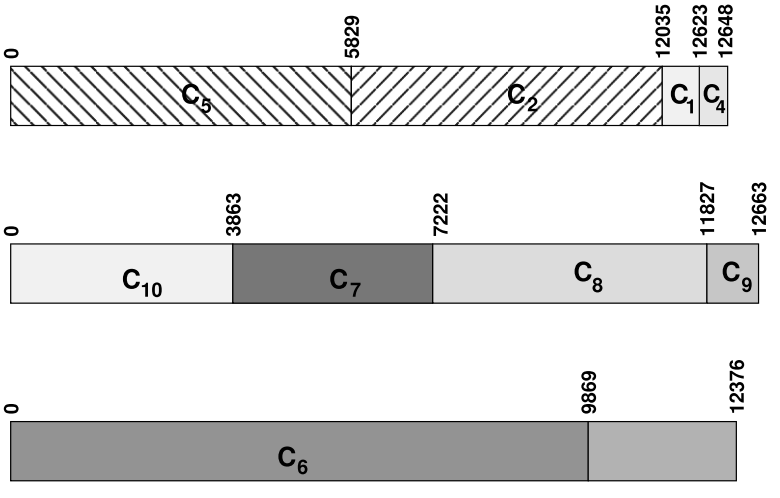

Figure 5. Scheduling results of *d*695 with a single clock rate using our ILP formulation.

Table 3. Optimal test times for the *d*695 SoC benchmark under different conditions.

Case	$W = 32$ Ours (Liu et al. 2005)	$W = 16$ Ours (Liu et al. 2005)
Single CLK	12,663 (13,228)	17,581 (17,807)
Case 1	10,457 (10,705)	15,524 (15,787)
Case 2	8474 (8534)	15,145 (15,369)

Table 4. Test times using a single clock ($W = 32$ and $W = 16$).

Benchmark	I/O	Single CLK (Liu 2005) $W = 32$	Single CLK (Liu 2005) $W = 16$
*d*695	2/2	13,227 (15,510)	22,250 (26,200)
*g*1023	3/3	14,794 (17,925)	14,794 (19,620)
*p*22810	3/3	112,793 (141,594)	202,681 (220,925)
*p*34392	3/3	544,579 (563,795)	544,579 (575,088)

Table 5. Test times using a multiple clock rates ($W = 32$ and $W = 16$).

Benchmark	I/O	Multiple CLKs (Liu 2005) $W = 32$	Multiple CLKs (Liu 2005) $W = 16$
*d*695	2/2	10,905 (13,336)	21,614 (24,848)
*g*1023	3/3	7397 (10,061)	12,297 (15,760)
*p*22810	3/3	92,554 (102,595)	175,350 (187,663)
*p*34392	3/3	275,409 (303,364)	300,117 (312,649)

the stopping criteria is met and hence less run time. Our results were achieved within few seconds to 10 min for small to large benchmarks. We compared our results with those reported in Liu et al. (2005) and Liu and Iyengar (2006).

First, we tested our SA without any power or temperature constraints under the assumption of a single clock rate. We compared our results with the techniques in Liu et al. (2005). Results are shown in Table 4 for the channel width of 32 and 16. The results from Liu et al. (2005) are presented in brackets. Then, we repeated our experiments using multiple clock rates. We assume that there are three clock rates, $f/2$, f and $2f$. The test times for the channel width of 32 and 16 are shown in Table 5. The results clearly show the importance of using multiple clock rates, especially to decrease the long test time of some cores. On the other hand, faster clock rates mean higher power consumption and consequently higher temperature. Our technique improved over the results in Liu et al. (2005) in all cases. On average, our single clock and multiple clock techniques reduced over Liu et al. (2005) by 7.02% and 7.7%, respectively.

We then tested our technique based on stringent power constraint under the scenario of multiple clock rates, where we set the power limit at a certain point not to exceed 25% of the total power consumption of all the cores. In Table 6, we show the testing time for different benchmarks using 25% stringent power constraint for $W = 16$. Results from our power-aware techniques assuming a channel width of 32 are presented in Table 7. Our techniques improved over the results in Liu et al. (2005) in most cases. Our power-aware single clock approach improved up to 11.67%, whereas our power-aware multiple clock technique reduced the test time over Liu et al. (2005) by 0.6%–23.4%. We also calculated the temperature of hot spots using 25% stringent power constraint. Although the power constraint is very stringent, the hot-spot temperature was still way over the acceptable thermal safe temperature of 127°C (third column of Table 7). Decreasing the maximum

Table 6. Test times using power constraints with multiple clocks ($W = 16$).

Benchmark	I/O	Power-aware (Liu et al. 2005)
*d*695	2/2	26,234 (26,683)
*g*1023	3/3	25,682 (29,076)
*p*22810	3/3	227,945 (248,287)
*p*34392	3/3	1,422,927 (1,422,927)

Table 7. Test times using power constraints with multiple clocks ($W = 32$).

Benchmark	I/O	Power-aware (Liu et al. 2005)	Max temperature
*d*695	2/2	18,786 (20,040)	136
*g*1023	3/3	16,749 (21,873)	203
*p*22810	3/3	142,213 (160,689)	227
*p*34392	3/3	781,834 (786,274)	214

Table 8. Test times using thermal constraints ($W = 32$).

Benchmark	I/O	Temperature-aware (Liu and Iyengar 2006)
*d*695	2/2	13,227 (13,547)
*g*1023	3/3	16,473 (18,368)
*p*22810	3/3	112,550 (150,714)
*p*34392	3/3	769,284 (785,488)

allowable power further will cause a significant increase in the test time and will render some benchmarks unschedulable.

It is clear from Table 7 that power consumption constraint is not sufficient to keep the whole chip safe as the temperature of some cores still exceeds the highest allowable temperature even though the total power is still less than the maximum allowable power. In this part of our results, we test our SA with the temperature constraint where we set $T_{\max}$ to 127°C. We first set up the parameters for thermal simulation for the Hotspot tool, such as layers, thermal resistance, thickness of layers and materials, and the chip dimensions. The testing time of our thermal safe schedule are presented in Table 8 for a channel width size of 32. Our thermal-aware SA solution was able not only to ensure thermal safety but also to decrease the testing time compared to the power constraints solution. On average, our temperature-aware technique improved over those reported in Liu and Iyengar (2006) by 6%.

8. Conclusion

In this article, we presented an optimal ILP solution to the power-constraint test scheduling for NoC under multiple clock rates. A suboptimal solution based on an SA is also presented where near optimal solution can be reached in a reasonable amount of time for big systems. Results on different benchmarks show the importance of our techniques, especially as the number of cores built on a single chip is continuously increasing.

References

Aghaee, N., He, Z., Peng, Z., and Eles, P. (2010), 'Temperature-aware SoC Test Scheduling Considering Inter-chip Process Variation', in *Asian Test Symposium*, December, pp. 395–398.

Ahn, J., and Kang, S. (2006), 'Test Scheduling of NoC-based SoCs Using Multiple Test Clocks', *ETRI Journal*, 28, 475–485.

Chakrabarty, K. (2000), 'Test Scheduling for Core-based Systems Using Mixed-integer Linear Programming', *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 19, 1163–1174.

Cota, E., Carro, L., and Lubaszewski, M. (2004), 'Reusing an On-chip Network for the Test of Core-based Systems', *ACM Transactions on Design Automation of Electronic Systems*, 18, 471–499.

Cota, E., Carro, L., Wagner, F., and Lubaszewski, M. (2003a), 'Power-aware NoC Reuse on the Testing of Core-based Systems', in *International Test Conference*, September, pp. 612–621.

Cota, E., Kreutz, M., Zeferino, C.A., Carro, L., Lubaszewski, M., and Susin, A. (2003b), 'The Impact of NoC Reuse on the Testing of Core-based Systems', in *VLSI Test Symposium*, May, pp. 128–133.

He, Z., Peng, Z., and Eles, P. (2010), 'Multi-temperature Testing for Core-based System-on-chip', in *Design, Automation and Test in Europe*, March, pp. 208–213.

Huang, W., Ghosh, S., Sankaranarayanan, K., Skadron, K., and Stan, M.R. (2006), 'HotSpot: A Compact Thermal Modelling Methodology for Early-state VLSI Design', *IEEE Transactions on Very Large Scale Integration Systems*, 14, 501–513.

ILOG Inc. (2002), '*ILOG CPLEX 8.1 Reference Manual*'. http://www.ilog.com/products/cplex.

Iyengar, V., Chakrabarty, K., and Marinissen, E.J. (2002), 'Co-optimization of Test Wrapper and Test Access Architecture for Embedded Cores', *Journal of Electronic Testing: Theory and Applications*, 18, 213–230.

Kirkpatrick Jr, S., Guatt, C.D., and Vecchi, M.P. (1983), 'Optimization by Simulated Annealing', *Science*, 220, 671–680.

Liu, C., Cota, E., Sharif, H., and Pradhan, D.K. (2004), 'Test Scheduling for Network-on-chip with BIST and Precedence Constraints', in *International Test Conference*, October, IEEE, pp. 1369–1378.

Liu, C., and Iyengar, V. (2006), 'Test Scheduling with Thermal Optimization for Network-on-chip Systems Using Variabe-rate On-chip Clocking', in *Design, Automation and Test in Europe*, March, pp. 652–657.

Liu, C., Iyengar, V., Shi, J., and Cota, E. (2005), 'Power-aware Test Scheduling in Network-on-chip Using Variable-rate On-chip Clocking', in *VLSI Test Symposium*, May, IEEE, pp. 349–354.

Marinissen, E.J., Iyengar, V., and Chakrabarty, K. (2002), 'A Set of Benchmarks for Modular Testing of SOCs', in *International Test Conference*, October, IEEE, pp. 519–528.

Nahvi, M., and Ivanov, A. (2001), 'A Packet Switching Communication-based Test Access Mechanism for System Chips' in *IEEE European Test Workshop*, May, pp. 81–86.

Pande, P.P., Micheli, G.D., Grecu, C., Ivanov, A., and Saleh, R. (2005), 'Design, Synthesis, and Test of Networks on Chips', *IEEE Design & Test of Computers*, 22, 404–413.

Rosinger, P., Al-Hashimi, B., and Chakrabarty, K. (2005), 'Rapid Generation of Thermal-safe Test Schedules', in *Design, Automation and Test in Europe*, March, pp. 840–845.

Tafaj, E., Rosinger, P., Al-Hashimi, B., and Chakrabarty, K. (2005), 'Improving Thermal-safe Test Scheduling for Corebased System-on-chip using Shift Frequency Scaling', in *International Symposium on Defect and Fault-tolerance in VLSI*, October, pp. 544–551.

Vermeulen, B., Dielissen, J., Goossens, K., and Ciordas, C. (2003), 'Bringing Communication Networks on a Chip: Test and Verification Implications', *IEEE Communication Magazine*, 41, 74–81.

Wang, L., Stroud, C., and Touba, N. (2007), *System on Chip Test Architectures*, San Francisco: Morgan Kaufmann.

Yao, C., Saluja, K.K., and Ramanathan, P. (2011), 'Thermal-aware test scheduling using On-chip temperature sensors', in *International Conference on VLSI Design*, January, pp. 376–381.