Published in IET Computers & Digital Techniques
Received on 20th October 2006
Revised on 19th June 2007
doi: 10.1049/iet-cdt:20060175

ISSN 1751-8601

Adaptive routing strategies for fault-tolerant on-chip networks in dynamically reconfigurable systems

J.L. Nunez-Yanez[1] *D. Edwards*[2] *A.M. Coppola*[3]

[1]*Department of Electrical and Electronic Engineering, Bristol University, Woodland Road, Bristol BS8 1UB, UK*
[2]*School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK*
[3]*STMicroelectronics, 12, rue Jules Horowith, Grenoble 38019, France*
E-mail: j.l.nunez-yanez@bristol.ac.uk

Abstract: An investigation into an effective and low-complexity adaptive routing strategy based on stochastic principles for an asynchronous network-on-chip platform that includes dynamically reconfigurable computing nodes is presented. The approach is compared with classic deterministic routing and it is shown to have good properties in terms of throughput and excellent fault-tolerance capabilities. The challenge of how to deliver reliability is one of the problems that multiprocessor system architects and manufactures will face as feature sizes and voltage supplies shrink and deep-submicron effects reduce the ability to carry out deterministic computing. It is likely that a new type of deep-submicron complex multicore systems will emerge which will be required to deliver high performance within strict energy and area budgets and operate over unreliable silicon. Within this context, the paper studies an on-chip communication infrastructure suitable for these systems.

1 Introduction

Network-on-chips (NoCs) have been proposed as a technology able to overcome the performance bottleneck faced by traditional shared bus designs. Conventional bus structures (AMBA, Coreconnect, and so on) have being coping with the increasing number of connected IP blocks with the introduction of complex hierarchies. This has helped to alleviate problems such as shared-medium access contention and wasteful data broadcast but the scalability of these solutions is limited. Additionally, the increases in noise and crosstalk expected in sub-65 nm dimensions will also hinder the performance of conventional bus structures [1]. The multicore technology of the future is poised to use on-chip packet or circuit-switched networks for effective communication. This will enable increases in bandwidth as more processing nodes are added, as well as providing support for error resilience and energy-saving techniques [2].

Synchronous and asynchronous NoCs have been proposed by different academic and commercial research teams. Asynchronous NoCs offer some clear advantages in solving power and delay problems associated with clock distribution in large chips. They also have some serious challenges regarding verification of the self-timed logic together with the possible area overheads introduced by having to add wires and logic to eliminate problematic glitches and support the handshaking protocols that replace the global clock. The lack of commercial electronic design automation (EDA) support for asynchronous logic is also a limiting factor in increasing its wide acceptance. Intensive work in the computational domain involving the design of complex self-time processor cores has been conducted in the Amulet group in Manchester University [3]. The challenges of self-timed logic are evident in this work and it is hard to imagine that this type of logic will replace well-understood synchronous design in the processing elements of a multicore platform in the short to medium future. On the other

hand, the higher regularity and fewer gates of the communication domain mean that the advantages of asynchronous design are more apparent [4].

The dynamic reconfiguration feature means that a single silicon platform could support multiple applications via rapid reconfiguration of the logic gates. Existing and emerging consumer audio/video standards, and the need for product differentiation by introducing new feature updates in already deployed devices, mean that the cost of the design, verification and fabrication of billion-transistor dedicated application specific integrated circuit (ASIC) [5] will be increasingly difficult to justify compared with a reconfigurable platform. Current platform FPGAs already incorporate a highly structured network at a very fine grain level used to connect the basic logic cells that form the FPGA computing fabric [6]. FPGAs, however, tend to consume a lot of energy most of it on the routing wires which can follow convoluted routes in the fabric as determined by the routing algorithm. Routing complexity increases with chip size and this tendency also decreases the performance since the high capacitance of long wires negatively affects the achievable clock rates. Introducing the NoC offers a natural way to structure this complexity so that FPGA routing domains are confined within the areas defined by the NoC wires and global communication is limited to using NoC resources.

The combination of an asynchronous communication infrastructure with dynamically reconfigurable computing nodes presents some interesting opportunities and challenges in the design of an effective communication protocol and their exploration constitutes the objective of this work. The paper investigates both the throughput and the fault-tolerance performance of this adaptive multicore platform which has been named network on a reconfigurable chip (NoRC). The main focus is how the presence of dynamically reconfigurable computing resources affects the design of a flexible routing strategy based on stochastic principles. Stochastic routing means that routes are discovered dynamically with little area overhead. Throughput, however, can degrade since typically more hops are needed for a packet to reach its destination. The paper is organised as follows. Section 2 describes the related work in the area of fault-tolerance strategies and adaptive routing in on-chip communication architectures. Section 3 presents the features of the NoRC platform and Section 4 describes how adaptive routing is implemented by NoRC. Section 5 studies the trade-off between the throughput and platform complexity, varying the number of functions supported in the processing nodes in NoRC, compared with a deterministic routing strategy. Then, in Section 6, an investigation is carried into the fault-tolerance capabilities and throughput offered by the stochastic approach compared with the same deterministic routing approach. Finally, Section 7 presents the conclusions and indicates future research directions. It should be noted that the simulator developed in this work does not pretend to be a cycle accurate model of a particular process but rather its role is to provide an initial insight into the features and limitations of the proposed NoRC platform.

2 Related work in NoC fault-tolerance and adaptive routing

Reliability in a multiprocessor system on chip (MPSoC) platform such as NoRC could be affected by soft/transient errors (cosmic rays, alpha particles, crosstalk and coupling noise) and hard/permanent errors (defective open or short circuits, electro-migration) introduced by the low yield and process variations that are likely in deep-submicron processes. The globally asynchronous locally synchronous (GALS) platform combined with dynamic voltage scaling could also generate synchronisation errors between the asynchronous and synchronous domains that become difficult to trace. Both transient and permanent errors should be taken into account when designing a fault-tolerance strategy.

Traditionally, analysis of transient faults assumes that some bits in the data packet are flipped to the wrong value and the literature focuses on the use of linear codes such as cyclic redundancy check (CRC) and Hamming to detect and/or correct these errors and the area/energy/performance trade-offs involved.

In [7], a detailed study is conducted on the energy-reliability trade-off comparing correction at the receiver stage against retransmission of the corrupted data. Error-correcting detection is based on cyclic codes and Hamming codes. The paper points out that advance interconnect schemes such as NoCs tend to have shorter point-to-point links which should favour retransmission rather than error correction from an energy point of view. Communication reliability can be obtained at different levels of granularity: it should be possible, for example, to add Hamming codes to each data flit while using a single CRC code to protect the whole packet. Retransmission at the switch level translates to a need for power hungry buffering resources so that end-to-end retransmission at the network interface (NI) level is generally preferred. Similar conclusions are reached in [8] which also investigate the advantages of either retransmitting the erroneous packets after error detection or fixing the problem using error-correcting techniques without the need for retransmission. The paper finds that the overheads of error correction in terms of

doi: 10.1049/iet-cdt:20060175

extra area are high and this also translates into high-energy consumption. A hybrid scheme is proposed that uses error detection and retransmission for short distances and error correction for long distances with a crossover point of six hops. This approach offers the best energy trade-off but it requires the presence of hardware modules for error detection and correction in the network interfaces and its area overhead is large.

The work presented in [9] studies the energy consumption incurred by different error-correcting/recovery schemes depending on whether they are deployed at the switch or network interface level. This work assumes static routing with paths set at the sender network interface. Critical information such as packet headers is protected with hardware redundancy such as triple modular redundancy (TMR). The scheme relies heavily on data buffering and these buffers are found to be the main cause of the power overhead. The conclusions are that switch-based error control works better for local communication while end-to-end is better suited for global communication.

In [10], a technique is proposed that uses different levels of error protection depending on the reliability of the communication link. As reliability of the link changes depending on level of noise present, the error-detection scheme changes from single error bit detection to double and triple error detection. The work focuses on error detection coupled with retransmission, and obtains energy savings because the complexity of the error-detection scheme is adjusted to meet the channel needs. The obtained energy savings are modest at 10% since the hardware required to support the most complex scheme needs to be physically present event if it is not used.

In [11], Hamming codes are also used to detect errors and retransmission used as the recovery method as it has been reported as being more energy efficient. The paper recognises that even with the addition of error-correcting capabilities to the system, the approach will not work in the presence of permanent errors since the error could affect the data in such a way that the error-correcting technique proves ineffective and multiple retransmissions would only deliver multiple copies of incorrect data. To tackle this situation, the approach adds TMR to the hardware with spare components. A reconfiguration strategy is used to select components that are functionally correct after error detection using Hamming codes. This works well so long as the extra links provided are able to eliminate the problem, but there is an obvious area overhead.

Additionally, it is reasonable to assume that errors could affect not only the data lines but also the control lines of the communication links and the logic of the router nodes [12]. This means that transient errors could result in routers which temporarily stop responding to requests or stop forwarding data correctly. A fault-tolerance strategy should also take into account these types of problems and the described techniques that rely on CRC codes/Hamming codes and retransmission and/or correction cannot do that on their own.

In [13], a comprehensive study of the most common failure types expected in NoCs is conducted and architectural solutions proposed for a generic synchronous NoC router. The approach is based on a combination of error detecting/correcting codes and retransmission. Buffers are heavily used in the router nodes to support hop-to-hop retransmission when the error correction codes are insufficient for solving the problem. Deadlock recovery is also included as part of the fault tolerance infrastructure. Both intra-router errors in the combinatorial logic and classic link faults are considered. It is apparent in the paper that the extra complexity added to the router to support all these fault-tolerant features is considerable.

The routing protocol employed has also a major impact on fault tolerance. Dynamic and adaptive approaches offer fault tolerance to temporal and permanent errors in the network compared with static schemes. The disadvantage is the complexity introduced in the routing nodes with extra processing, larger buffers and large routing tables that can quickly make the communication network complexity unmanageable in an embedded system. Dynamic routing algorithms typically use tables to determine the position of the receiver of a data packet [14]. The route is dynamically discovered and path changes can be accommodated easily.

A simpler approach to the dynamic discovery of the route that removes the need for routing tables and complex processing is the randomised gossip protocols proposed in [15]. In this case, if a router has a packet pending to send, it will forward it to a randomly chosen subset of neighbour routers. These routers will upon reception of the packet behave in the same way. This type of probabilistic broadcast algorithm has very good fault-tolerance properties but generates a lot of redundant traffic with multiple replicated copies of the same packet using scarce resources in the network which could result in energy waste and network congestion. The work conducted in [16] investigates the flooding mechanism used in [15] and concludes that excessive unnecessary redundancy is generated. A variation is proposed in which a limited number of packets move towards the destination in a directed way. This means that routing decisions are not totally random but the routers know the destination of the packet and some weighting is performed so that the

doi: 10.1049/iet-cdt:20060175

ports that lead towards the destination have a higher probability of being chosen than the other ports.

The routing approach used in this work shares some of the stochastic features of [15] and avoids packet replication by using a single request flit to establish the communication followed by a variable number of deterministically routed data packets. It combines packet switching for the initial single-flit request packets with circuit switching for the data flits. It is designed to exploit the features of the dynamically reconfigurable platform by replacing the inflexible destination addresses with service requests. This adaptive routing offers fault tolerance to permanent failures in the processing and routing nodes. For transient errors affecting the packet header (that identifies the service requested) and/or payload, a combination of error-detecting Hamming codes at the flit level and CRC codes at that packet level can be used to detect erroneous bits packet headers and payloads. Upon detection of errors, the packets are dropped and a retransmission requested since this technique is simpler and it has been found in the reviewed literature to be more efficient that full-blown error correction. One of the most attractive features of the routing used in NoRC compared with the reviewed methods is its simplicity since it does not require buffering of the packets at the router level, routing tables or hardware redundancy to operate efficiently.

3 NoRC platform

Fig. 1 illustrates the NoRC platform which combines NoC, a reconfigurable fabric, hardwire processing functions such as a RISC CPU and embedded digital signal processor (DSP) blocks, dynamic voltage scaling and GALS technology in a single silicon chip.

Our research into this system is based on the on-chip communication network (OCCN) [17]. OCCN is an open-source object-oriented SystemC library which provides an efficient framework for the modelling, simulation and design space exploration of on-chip communication architectures. OCCN enables the separation between the communication and computation with the use of the communication interface which is the only way to interact with the behaviour of a processing node. This enables both the communication and behaviour components to be described at different levels of detail and these modules to be freely intermixed as along as the interface protocols remain the same. Packets in OCCN are called protocol data units and typically consist of a header field, payload field and a trailer field. Packets are defined as the smallest part of a message that can be routed independently through the network. The packets are transmitted and divided into flits.

Segmentation and reassembly are implemented among different layers of communication (transport, network and link layer). Communication refinement enables, for example, replacing the abstract communication channel that transfer 32-bit flits between the master and slave ports into a detailed description of a dual-rail return-to-zero asynchronous protocol similar to CHAIN links.

In this work, we have ported the SystemC OCCN library and classes to the ModelSim simulation environment. Modelsim is considered the standard in RTL (VHDL, Verilog) simulation and for some time now it also supports SystemC simulation. The three languages can be intermixed in the same simulation and this opens some interesting avenues for communication and computation refinement. The initial NoRC platform is based on a 2D mesh Manhattan topology that have been shown to be VLSI friendly [18] and well suited for allocation of the regular blocks of reconfigurable fabric. Typical parameters in the SystemC code describing NoRC include: network dimensions, packet injection rate, transient error rate, permanent error location and timing, router and link latency, retry latency, buffer, packet and flit size. Fig. 2 gives an overview of the different components used in the SystemC NoRC model. Each computing tile can behave both as a master or slave. The figure only shows a control thread running in the router but in reality five control threads (one per port) run in parallel monitoring activity in each of the ports.

We have aimed at minimising the complexity of routers since they represent the main area and power overhead in the on-chip network [19]. Buffering in the router nodes is typically introduced to provide guaranteed throughput (GT) so routers can buffer packets if high priority packets arrived that must be forwarded immediately. Additionally, buffers temporarily store clean copies of data units (packets, flits) and can be used to retransmit the data if errors are detected obtaining fault tolerance. On the negative side, buffering also means that delay times go up and the buffers can quickly become the highest contributor to the area overhead. In the following sections, we investigate a routing/switching strategy that can work without buffers while providing good levels of fault tolerance and throughput in the context of the dynamically reconfigurable NoRC platform.

4 Flexible routing in the NoRC platform

NoRC aims at supporting the high-demand processing power expected in multimedia MPSoC. The initia NoRC configuration includes processing nodes

doi: 10.1049/iet-cdt:20060175

(transmitter and receiver), routers and channel classes which have been extended to collect statistical data for throughput and packet loss. The network interfaces move data from the synchronous processing domain to the network asynchronous domain. Computation is synchronised with a clock signal modelling a GALS platform. Buffering is limited to the network interface with the use of asynchronous FIFOs. No extra buffering is provided in the router ports to keep complexity low. To further eliminate the need for buffering in the network interface, NoRC provides a connection-oriented communication

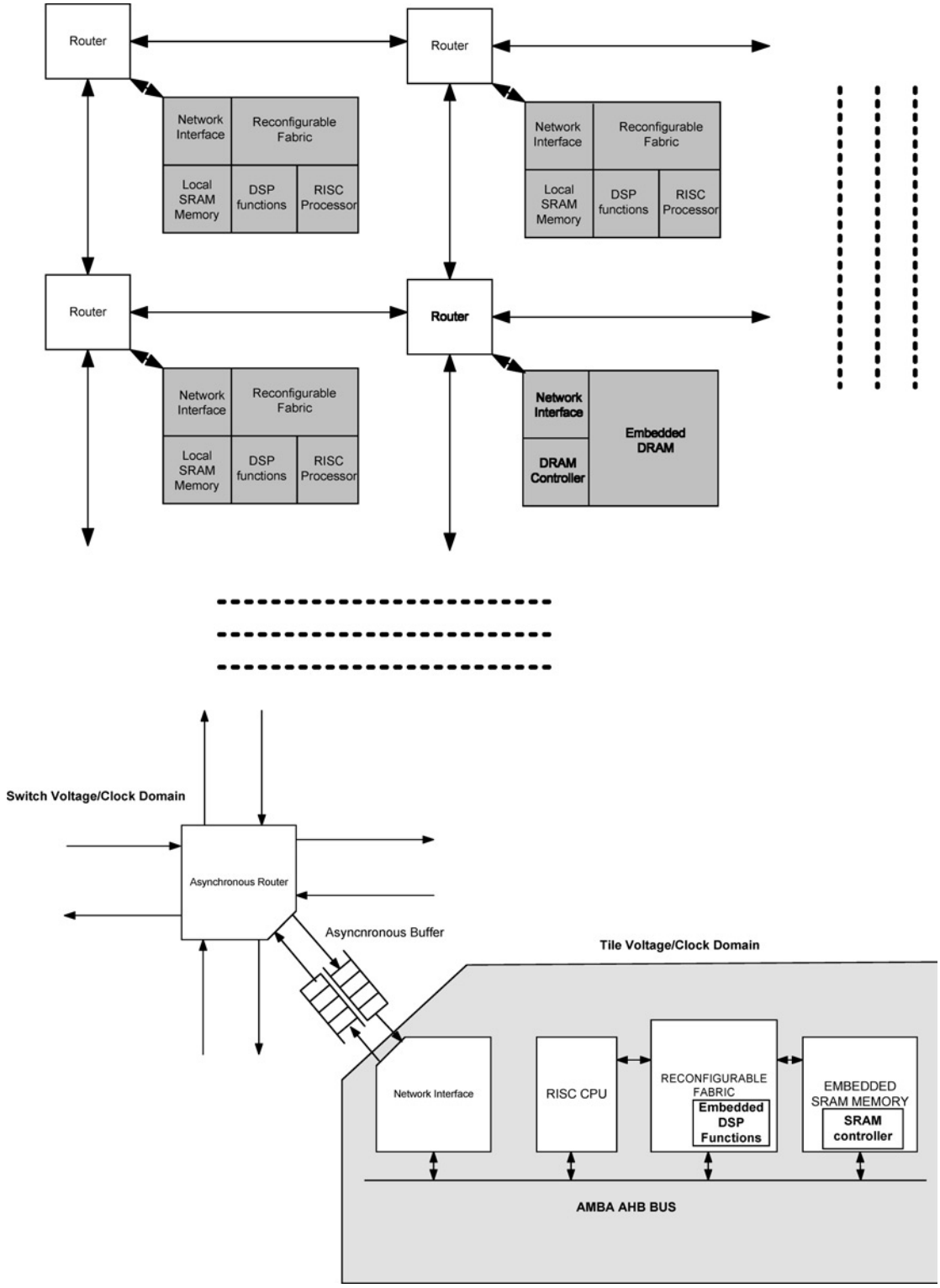

Figure 1 *NoRC components*

doi: 10.1049/iet-cdt:20060175

protocol so the delivery of the payload flits is done in order avoiding the extra complexity of a reordering scheme.

NoRC delivers flexibility by not having the IP of the processing nodes determined at design time. Consequently, more resources can be dedicated to the communication infrastructure at design and fabrication time since these costs can be shared over the multiple applications in the multimedia domain. The predictable wire lengths in the proposed homogenous NoRC mean that inductance, resistance and capacitive parasitic effects can be controlled allowing aggressive circuit design techniques such as low swing drivers and receivers that will be beneficial for the energy-efficient design paradigm.

Communication in NoRC starts with a single request flit used to set-up the connection between the transmitter and receiver followed by a variable number of flits containing the data payload. The final flit contains the CRC code used to verify error-free communications. Successful CRC verification results in a single acknowledge flit send to the transmitter which is also used to release the resources reserved by the initial request flit.

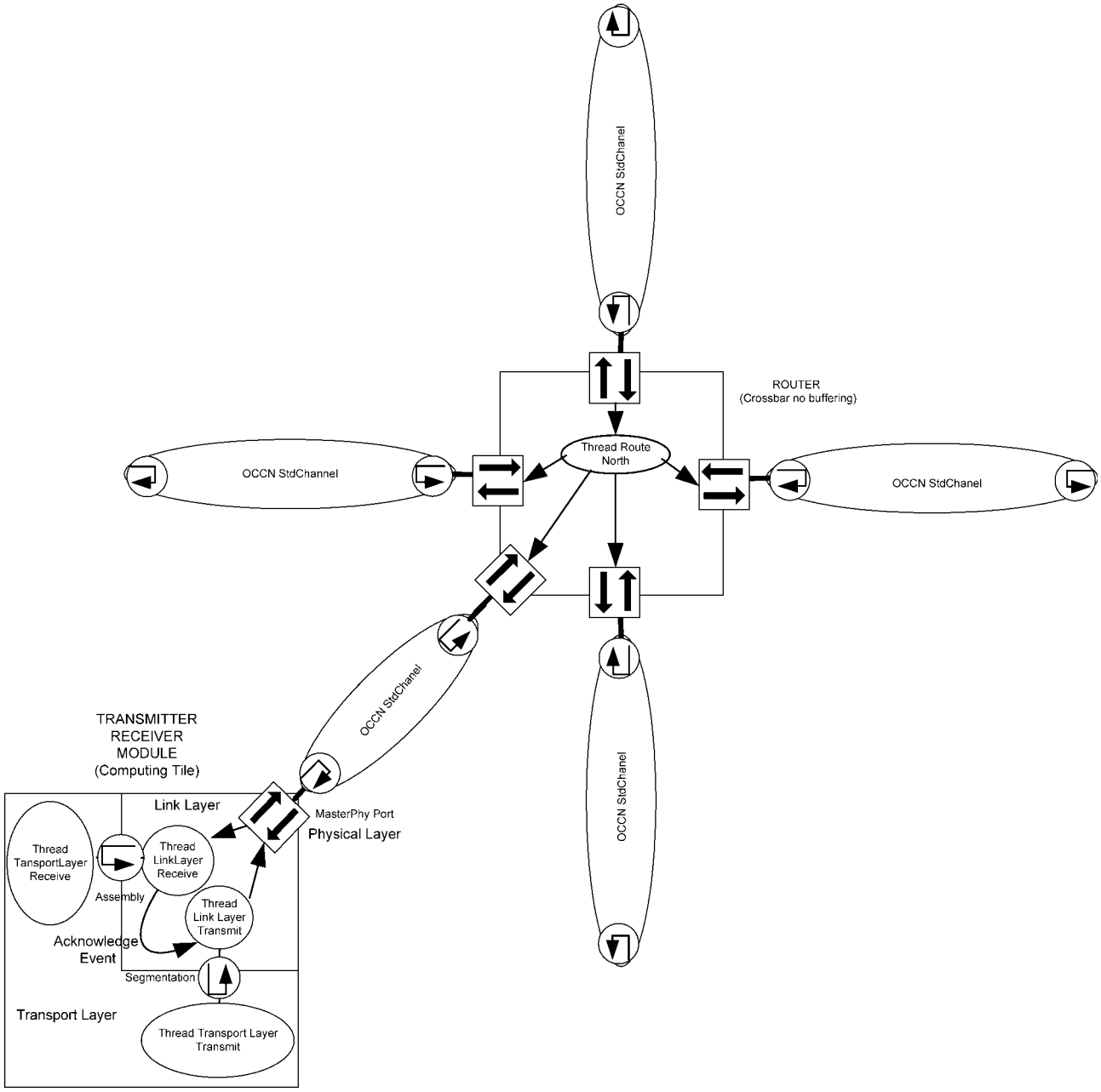

Figure 2 *SystemC NoRC components*

doi: 10.1049/iet-cdt:20060175

The initial request flit follows a store-and-forward switching approach with the single-flit packet being stored in the router before a decision on where to forward it is made. Once the circuit is established payload transfer is done using circuit switching with no further arbitration being required. This enables in order delivery and good levels of GT [20]. The presence of circuits removes the need for buffers since only enough capacity to store a single flit is required.

In NoRC, we envisage that processing nodes perform request for services sending data packets that need certain processing to be done on them (for example, motion estimation in a macroblock in video coding). The request flits routing could be based on a randomised algorithm because we assume that the service provider location is not known in this distributed computing platform based on dynamic reconfiguration and FPGA-like silicon structures. This function-based random-routing strategy means that the request flits do not contain a destination address but a function identifier. A single copy of a request flit exists and when it is received by a router, an initial check is performed to verify if the processing node attached to that router can service the function request. If this is the case, the request flit is delivered to the processing node which will acknowledge it once it is ready to start accepting the data payload. In this work, the flit size has been configured to 32 bits although other flit sizes are certainly possible.

This strategy means that the circuits established for the data payload transfer are not fixed but they can change between packets so adaptation to dynamic traffic is provided. When the first choice of output port for the request flit is busy, a hot potato/deflective approach is used to choose another router port randomly. Request packets are constantly transferred among routers until they reach a processing node that can service the request contained in the flit avoiding the need for extra buffering. If all the output ports are busy, the request flit bounces back with a circuit unsuccessful flag set. The sender will then wait for some small random amount of time before requesting the same service. This new request will follow a different route from the one used by the unsuccessful request.

Fig. 3 shows a flow chart of the routing decisions taken place in one of the five ports available in each router. There are three main flit types: request flits, acknowledge flits and data flits. Each flit arriving at a router port is checked to identify it as one of these three types. A flit identified as a request flit that

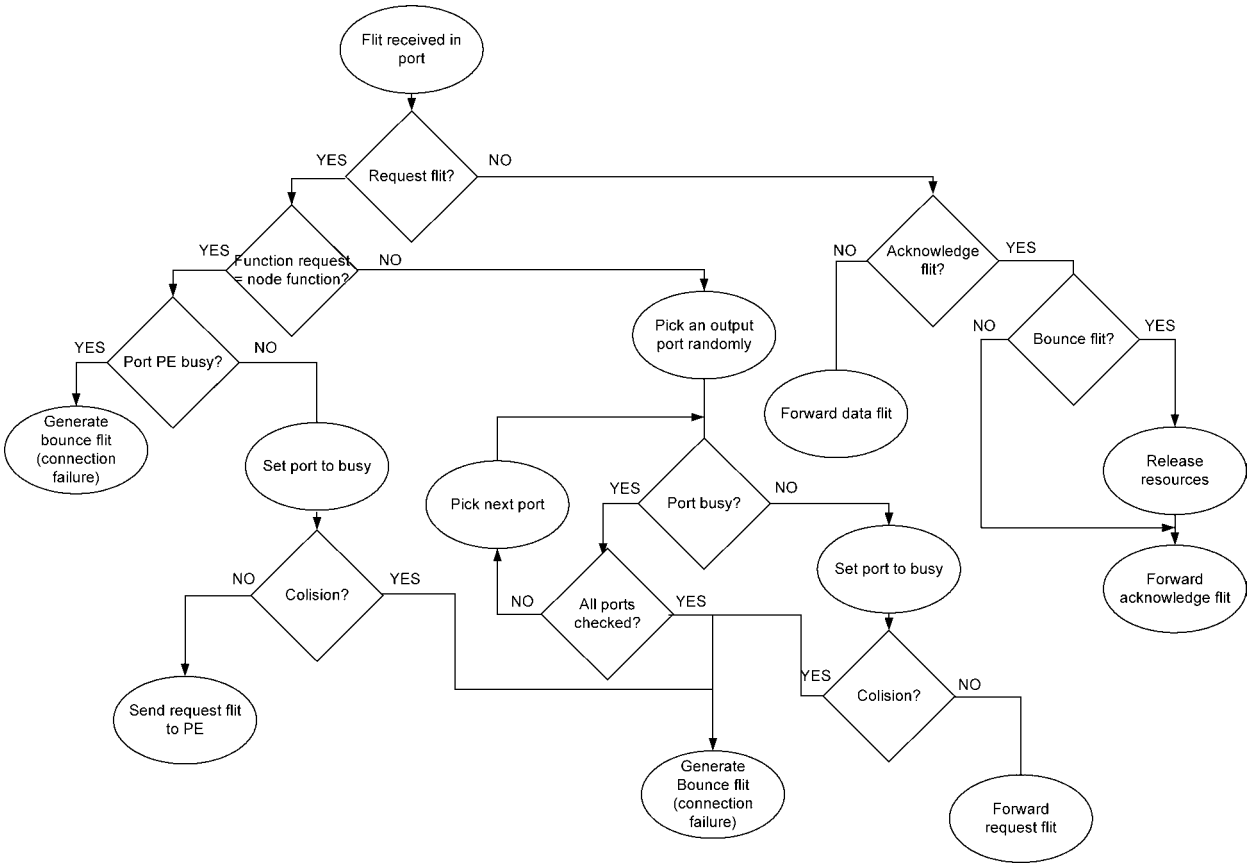

Figure 3 *Router activity flow chart*

doi: 10.1049/iet-cdt:20060175

cannot be serviced in the current processing node, results on a forward operation using a randomly chosen port and setting the corresponding input port, output ports and direction as busy. For example, if the north–south direction is reserved future flits entering through the north port are automatically forwarded to the south port without further arbitration. Similarly, flits entering through the south port are forwarded to the north port without further arbitration. This direction reservation is maintained until an acknowledge flit is used to free the reserved resources. Acknowledge flits are also generated when a request flit cannot progress any further and bounces back (connection failure). This situation could arise with busy or faulty ports. This specially flagged acknowledge flit releases resources as normal and informs the source node that a connection failure has taken place. Possible errors affecting the bits in these controls flits could be detected using error-detection codes. Additionally, if an acknowledge flit gets lost in transit the source is designed to retry the transmission after a certain time out period. Fig. 3 indicates the possibility of collisions among request flits. This circumstance can take place when two request flits originating from different nodes arrive at two ports which share the same channel. For example, ports west and east in routers horizontally adjacent. Under this condition, both request flits bounce back resulting in two connection failures.

Figs. 4 and 5 show examples of successful connections established between the source and receiver nodes. Source nodes cannot request a function that can be provided by themselves. The dark squares represent unavailable busy ports so the router is forced to randomly choose another port. Request flits keep moving sometimes crossing more than once the same router as shown in Fig. 5 until a node that can provide the requested function is found. Fig. 6 shows a situation in which a flit arrives at a router that has its

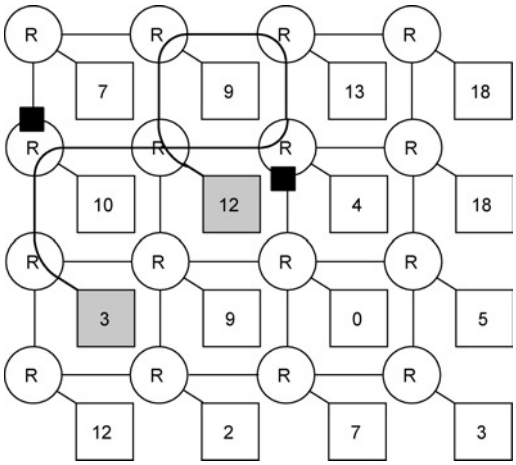

Figure 5 *Process 12 requests service 3*

two possible exit ports busy resulting in a connection failure. The bounced flit is used to release all the reserved resources up to that point and inform the source about the problem. A new retransmission will result in a new route and node chosen to service the request.

To compare this dynamic random-walk/stochastic routing approach, we have implemented the deterministic *XY* algorithm routing leaving the rest of the NoRC infrastructure intact. In *XY* routing, the positions of the processing nodes are described by coordinates, the *X*-coordinate for the horizontal and the *Y*-coordinate for the vertical position. Packets are routed to the correct horizontal position first and then in the vertical direction. *XY* routing produces minimal paths and simple routers but its tolerance to permanent and transient errors is low.

In order to be able to make direct comparisons with the stochastic approach, the initial single-flit request is routed along the horizontal direction until the router location *Y*

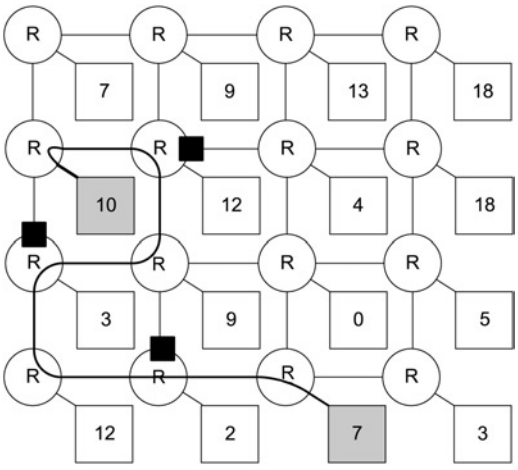

Figure 4 *Process 10 requests service 7*

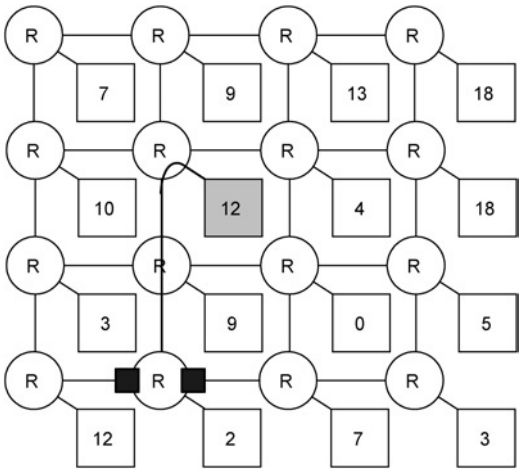

Figure 6 *Process 12 connection failure*

doi: 10.1049/iet-cdt:20060175

index matches the *Y* part of the address. Then, the request flit moves along the vertical direction until the flit finds its destination. If the receiver accepts the request and the acknowledge flit is sent back to the transmitter and a connection established between the sender and transmitter which will persist until the receiver sends an acknowledge flit indicating successful or erroneous payload transfer. Independent of the data payload transmission being successful or erroneous, the transmitter will need to re-arbitrate for the connection resources using another request flit before more data can be sent. If the request packet cannot proceed according to the *XY* route, it bounces back releasing the resources reserved up to that point and informs the transmitter about the failure to establish the connection. Similar to the random case, the transmitter waits for a small random amount of time before trying to establish a connection again.

The *XY* strategy assumes that the transmitter node knows the location of the receiver node. For example, in a multimedia application involving video coding, the fact that a computing node providing motion estimation services is located at position [3, 2] should be known by a control thread processing a particular macroblock in position [1, 1]. The stochastic approach is more flexible and it does not require this knowledge to operate correctly. It is, therefore more suitable to a platform such as NoRC which enables dynamic reconfiguration of the nodes based on power or processing demands so the number and location of processing nodes being able to perform a particular task could vary as requirements change.

In summary, transmitters and receivers behave differently for the *XY* routing and for the random case. In *XY* routing, the transmitter chooses a receiver based on *X*- and *Y*-coordinates, whereas in the random case, it chooses some processing needed on a data packet rather than which is the processing node that is going to perform the task.

In principle, the random strategy should generate more link transversals and consume more energy since *XY* follows the shortest path, but this depends on which is the ratio between the processing nodes and request functions. If the number of processing nodes is much larger than the number of functions available in the system, the chances are that a request flit will soon find a node able to service the requested function. Once the link is established, the flits containing the data payload are transmitted using the reserved resources in the path without further arbitration.

5 Throughput analysis

For the throughput investigation, we have configured the NoRC SystemC model with a link delay of 1.3 ns corresponding to a link length of 2 mm in a CHAIN-style network using 0.18 μm technology [21]. This link delay includes all the transitions required in a return-to-zero dual-rail encoding protocol to transmit one bit of data. We have also configured the router latency to 5 ns and the clock used by the computing nodes to 10 ns. These values are used by both configurations: *XY* and routing and random so a fair comparison can be carried out.

Table 1 Network load points

Description	Time between request packets	
	Min, ns	Max, ns
light load	0	5000
medium load	0	500
high load	0	50
maximum load	0	5

We have defined four operation points in terms of network load. These load points enable the study of network behaviour under different levels of congestion. The load is determined by a random variable with different bounds depending on the level of congestion required. Table 1 gives details of the four loading points used.

Fig. 7 shows the behaviour of a reliable (no retransmissions or failures included) 4 × 4 NoRC for *XY* routing and random. The *Y*-axis shows throughput

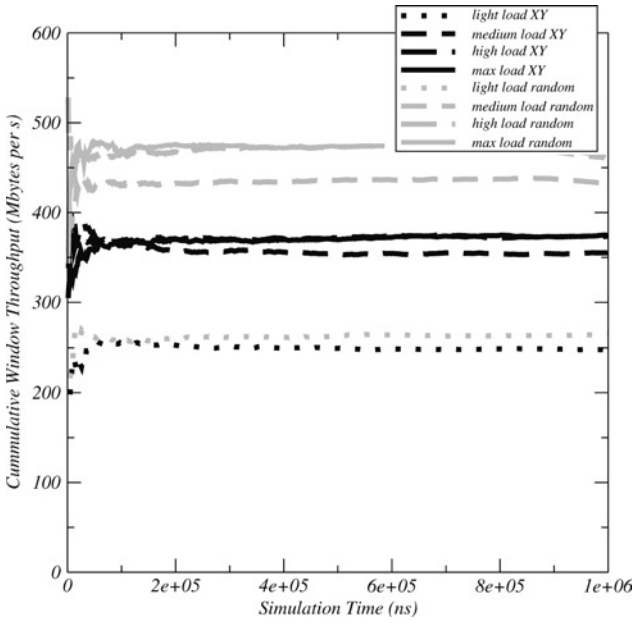

Figure 7 *Throughput analysis (four-service stochastic and XY)*

doi: 10.1049/iet-cdt:20060175

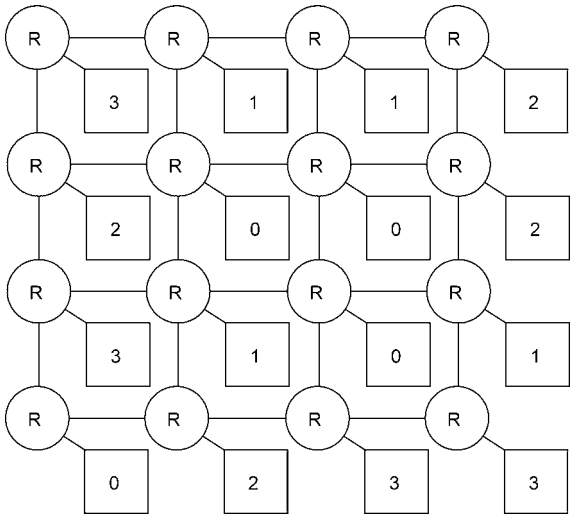

Figure 8 *Distribution for four services*

as a cumulative value obtained by averaging the throughput obtained in multiple small time windows of 50 ns. The *X*-axis shows the simulation time in nanoseconds. The NoRC using stochastic routing is configured with four functions distributed randomly as shown in Fig. 8. From this figure, it is apparent that for this configuration, if a node implements a particular service there is a high chance that the neighbouring nodes will implement the other three services. This is not always the case since, for example, if the top-right node needs service number 3, the request flit will need to transverse the whole network before a node servicing the request can be found.

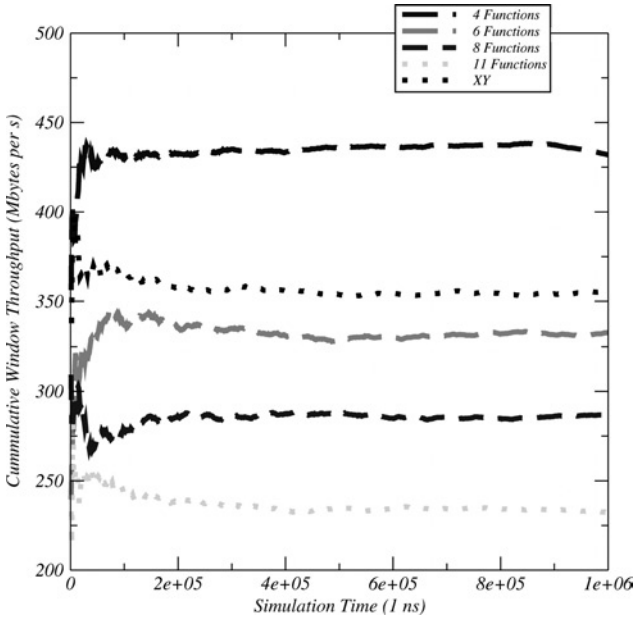

Figure 9 *Throughput analysis (service count impact on stochastic routing)*

In Fig. 7, we observe that the throughput for the *XY* and stochastic cases under light load conditions is very similar. If we increase the load, we observe that the *XY* case saturates the network much earlier than the stochastic case. This means that the random routing can delivered almost 30% extra bandwidth compared with *XY* routing for heavy load conditions.

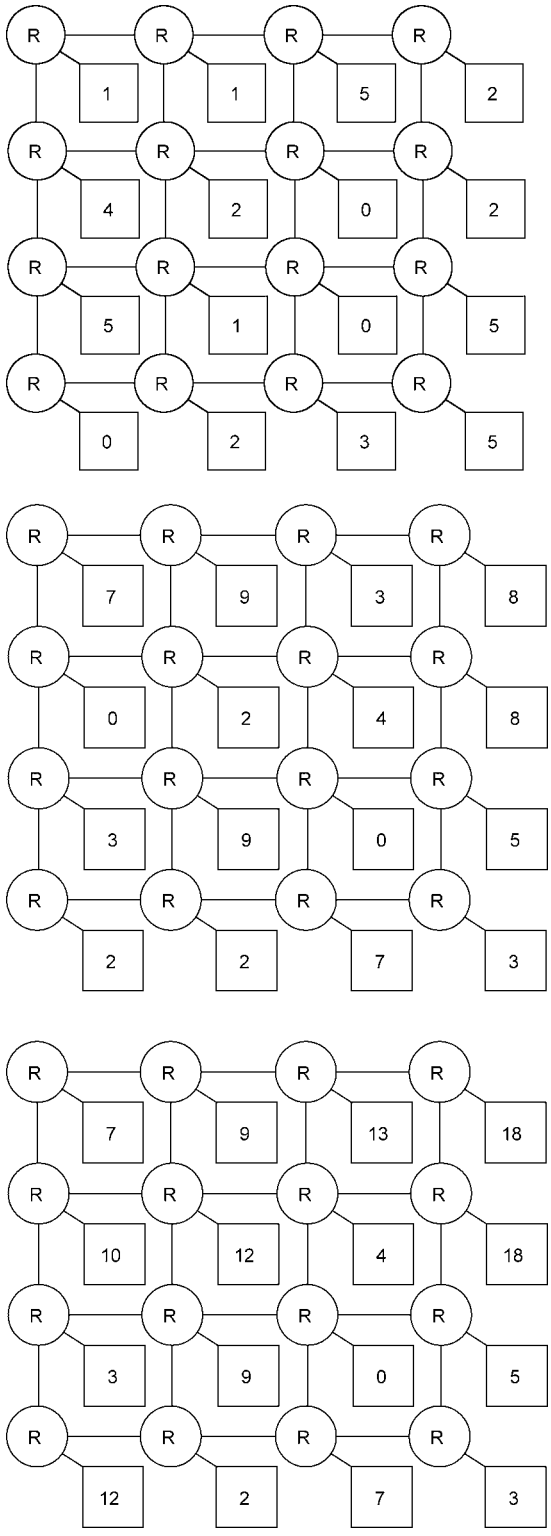

Figure 10 *Distribution for 6, 8 and 11 services*

doi: 10.1049/iet-cdt:20060175

Fig. 9 investigates the behaviour of random routing varying the number of services available in the system for the high load case. We compare the case with four services shown in Fig. 7 with 6, 8 and 11 services. Increasing the number of services means that the probability of quickly finding a node that can service a particular request decreases and network capacity is reduced. In a real implementation, it is expected that the number of services will change dynamically and the capacity of the network as well. Fig. 9 shows that in all the cases the network reaches the steady state quickly and continues to operate without service interruptions (no deadlocks) with data payloads successfully transferred between request originators and service providers. Fig. 10 shows the function distribution in NoRC for each of the new three cases.

Fig. 11 shows the behaviour of the 4 × 4 mesh for the stochastic six-service and *XY* routing with the network saturating at around 350 MB per second. Both algorithms offer similar levels of performance in this particular configuration assuming a reliable NoRC. Since in the random case, the transmitter does not need to know where a particular service provider is located, packet data could contain not only payloads needed to be processed but also bitstreams defining the functionality for the node. For example, a network node that is failing to meet processing deadlines because some particular function is not being performed fast enough could request the generation of a bitstream packet implementing that function. A configuration node could then generate a request for an available node which is not performing any useful function. A node replying to this request could accept

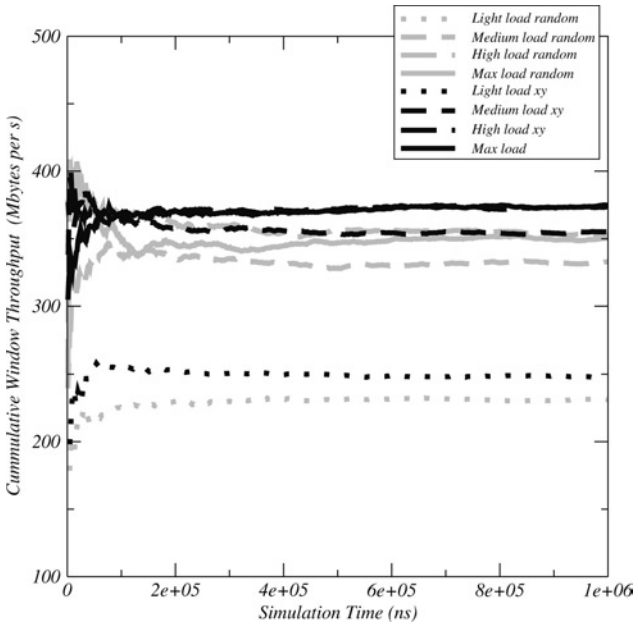

Figure 11 *Throughput analysis (six-service stochastic and XY)*

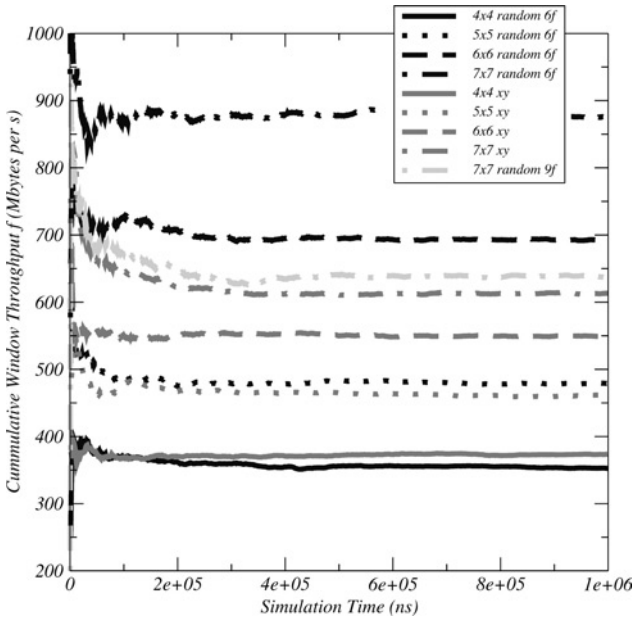

Figure 12 *Throughput analysis (network size)*

the bitstream packet and start the new service. The dynamic discovery of the route provided by the random algorithm means that no explicit communication to identify service provider locations is required.

Fig. 12 compares the capacity of *XY* and random routing using different NoRC sizes: 4 × 4, 5 × 5, 6 × 6 and 7 × 7. The performance of the six-service 4 × 4 and 5 × 5 NoRCs is very similar but, for the 6 × 6 and 7 × 7 cases, it is apparent that the stochastic algorithm works better. We then investigate how many extra services the stochastic algorithm can accommodate before its performance is similar to *XY*. This is shown in Fig. 12 for the nine-service case which works very similar to *XY*. This is then the service count threshold between both algorithms. Increasing the service count further means that the

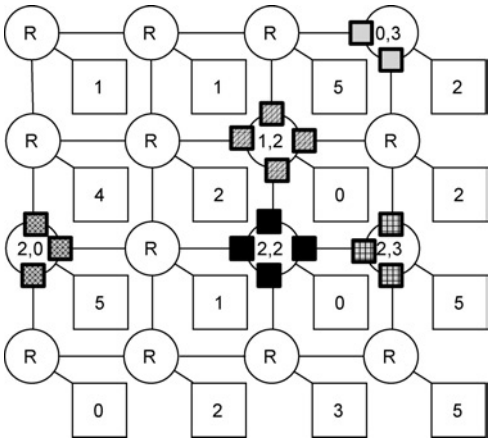

Figure 13 *Router error distribution*

doi: 10.1049/iet-cdt:20060175

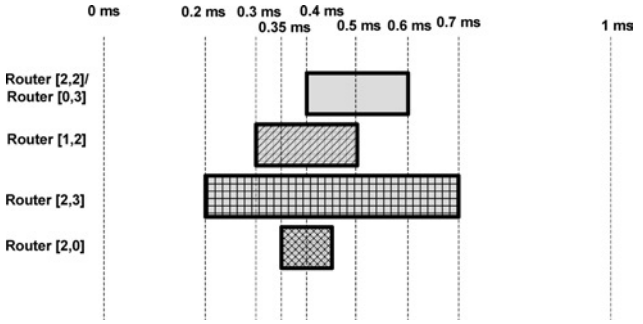

Figure 14 *Router error relative timing*

capacity of stochastic is lower than *XY* but, in any case, it is important to note that capacity is stable and grid lock states are not detected. The next section investigates the advantages of stochastic compared with *XY* in terms of fault tolerance.

6 Fault-tolerant analysis

To explore the fault-tolerance properties of the NoRC approach, we have used a simple 4 × 4 NoRC configured with a total of six different services and high traffic load. The service distribution and the port errors progressively introduced are illustrated in Fig. 13, whereas Fig. 14 shows the relative timing of these errors.

Fig. 15 shows the effects of a single permanent fault in the south port of router [2, 2] in a 4 × 4 NoRC. The graphs use an instantaneous window throughput so each point in the graph represents the throughput of the network in a small window of 50 ns. The fault takes

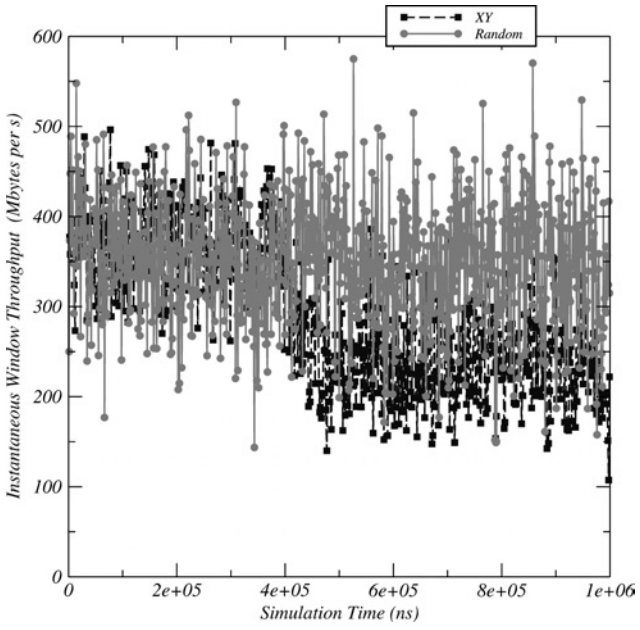

Figure 15 *Fault-tolerant analysis (faulty south port in router [2, 2])*

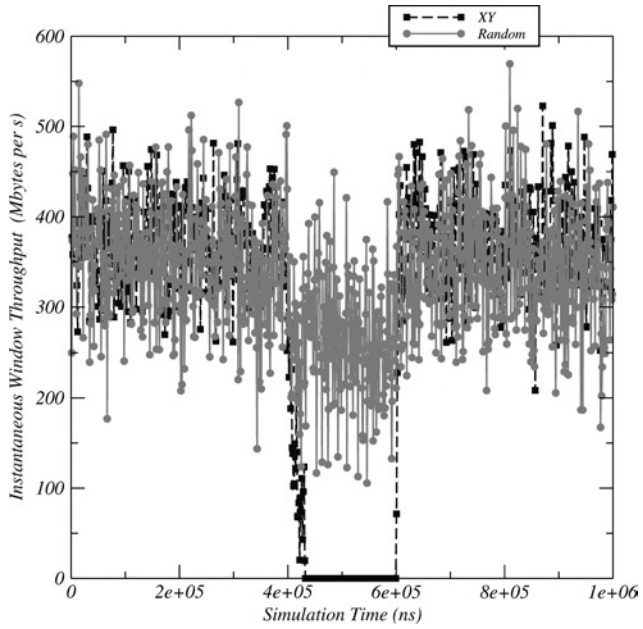

Figure 16 *Fault-tolerant analysis (temporal fault in router [2, 2])*

place at 400 μs and the *XY* algorithm shown in darker colour immediately suffers a degradation in throughput. It is expected that as more request flits try to make use of this router, the bandwidth will decrease further until zero packets can get through. On the other hand, the random algorithm continues to operate without showing any significant reduction in throughput.

Figs. 16 and 17 show the effects of transient and permanent router malfunctions, respectively. The

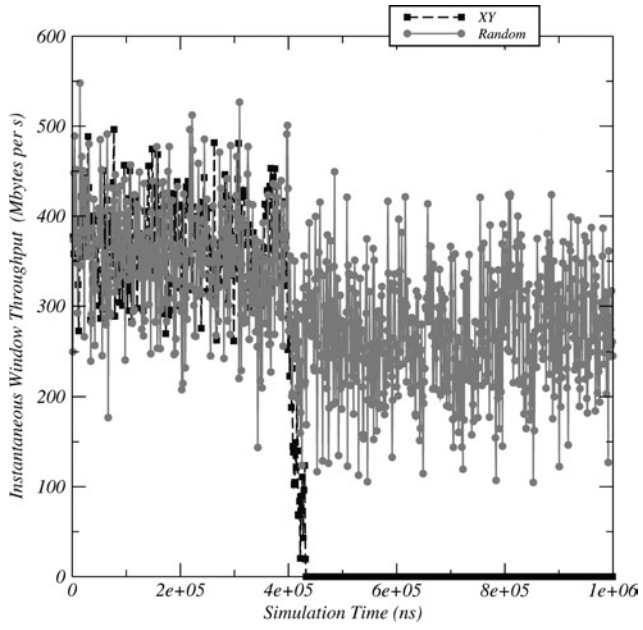

Figure 17 *Fault-tolerant analysis (permanent fault in router [2, 2])*

doi: 10.1049/iet-cdt:20060175

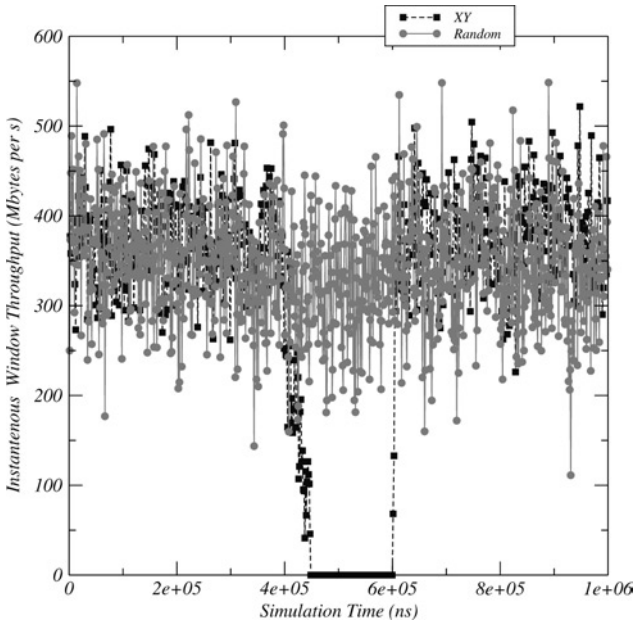

Figure 18 *Fault-tolerant analysis (temporal fault in router [0, 3])*

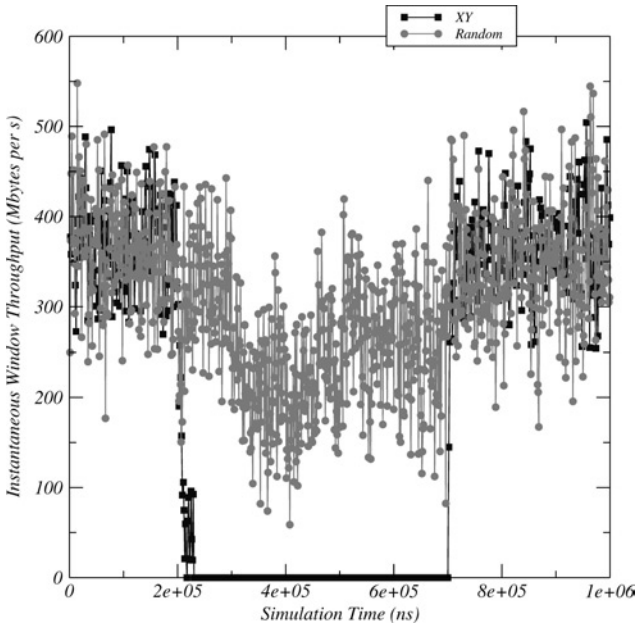

Figure 20 *Fault-tolerant analysis (temporal fault in router [0, 3], [1, 2] and [2, 3])*

same router [2, 2] in the 4 × 4 NoRC is chosen for these experiments.

From the figures, it is apparent that the *XY* routing strategy has no fault-tolerance capabilities and immediately breaks causing the whole platform to stop processing. The reason is that all the processing nodes, within a short time interval, try to route some data through router [2, 2]. Despite the constant failures, the algorithm has no flexibility to choose alternative paths and the network quickly locks up. The random strategy has a reduction in throughput when router [2, 2] stops working but it continues to operate in a stable state and it shows the good fault-tolerant properties expected in an adaptive routing strategy.

Fig. 18 shows the effects of a failure in router [0, 3] instead of router [2, 2] to investigate how the relative location of the router affects the drop in performance. The *XY* router again encounters a dead lock situation

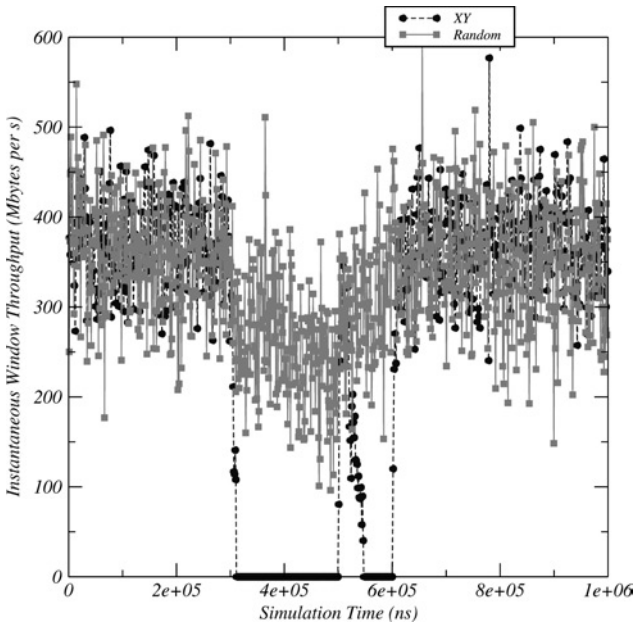

Figure 19 *Fault-tolerant analysis (temporal fault in router [0, 3] and [1, 2])*

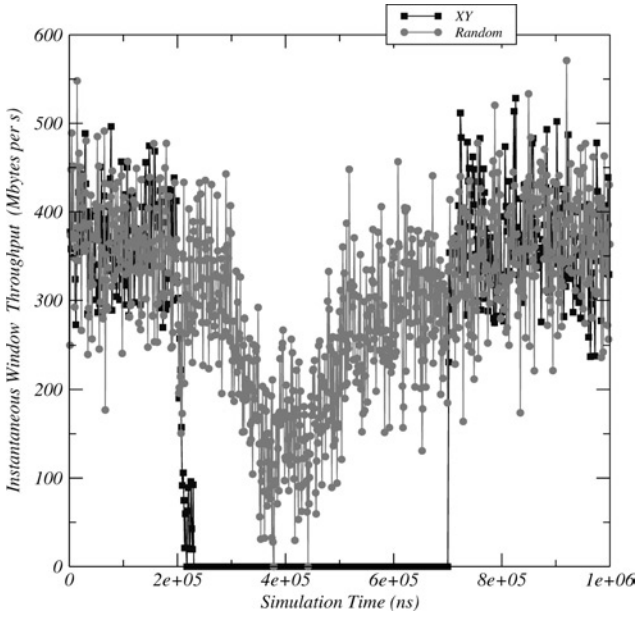

Figure 21 *Fault-tolerant analysis (temporal fault in router [0, 3], [1, 2], [2, 3] and [2, 0])*

doi: 10.1049/iet-cdt:20060175

as nodes progressively try to use a router that keeps bouncing back the packets. The negative effect in the stochastic case is less noticeable since router [0, 3] has one port less and less chances of being used to route traffic. In Figs. 19–21, the number of routers failing is increased to two ([0, 3] and [1, 2]), three (add [2, 3] to the previous two) and four (add [2, 0] to the previous three), respectively. It can be observed in the figures that a failure in a single router for the *XY* case always hangs the system, whereas for the stochastic case, the degradation in performance is evident with the increase in the number of faulty routers but it is not critical. The limit can be seen in Fig. 21 during the time interval in which four routers fail in parallel and there are some time windows in which data do not get through the system. This suggests that for this particular distribution of load, services and network size, a failure rate superior to 4 could hang the network. It is also important to note that in these experiments, we have made sure that at least on service provider for each service in the network remains fault free. Otherwise, requests would be left in the network and the system would eventually hang, as progressively each node makes a request for the unavailable service.

7 Conclusions and future work

This paper has presented the NoRC platform and study how the features expected in a dynamically reconfigurable platform can be exploited by a simple but efficient stochastic routing algorithm. The initial results are compared with a simple deterministic algorithm and show competitive levels of performance in terms of throughput. The main advantage of the stochastic strategy is its excellent fault-tolerant capabilities and the flexibility introduced in the distributed multiprocessor platform with a service-based strategy. This means that there is no need to maintain up-to-date location tables in the processing nodes as the platform dynamically changes. Error resiliency is expected to be a fundamental feature that systems fabricated with billions of transistors in deep-submicron processes should possess to be cost effective. The regularity of the proposed NoRC together with stochastic routing could be the steps in the right direction. The dynamic reconfiguration obtained with the FPGA-like processing nodes should enable one device service different requirements in the multimedia domain. Traditionally, multimedia applications can tolerate small service drops and could be a good match to the stochastic approach. For example, a few erroneous pixels in one video frame of a high definition display working at 50 frames per second do not invalidate the system. Economics dictate that the fabrication costs of one of these complex devices in ultra-deep submicron technology will be very high, of the order of a few million dollars per mask set, and as such this cost should be amortised over the device volume required for multiple applications.

GT traffic could be one of the challenges faced by a stochastic approach such as the one proposed in this paper. A simple option could be to reserve the routing resources needed by GT traffic and let the stochastic algorithm find an alternative path for non-critical traffic. Additionally, the dynamic reconfiguration capabilities could be used to move physically closer transmitters and receivers involved in the GT traffic. We expect to investigate these alternatives as work progresses.

Further work should also investigate the energy consumption introduced by the stochastic alternative. In principle, more flits need to be retransmitted to found a service provider but the request flits are very small and need a fraction of the energy required by a full packet transmission. The connection-oriented transmission of data packets without arbitration once a circuit has been successfully established should limit energy waste. To introduce a suitable energy model in the SystemC NoRC model constitutes part of our future work. We will also investigate the suitability for asynchronous realisation of the stochastic router using the Balsa synthesis toolset [22] and extract data in terms of latency and energy consumption to further refine the initial model presented in this paper.

8 References

[1] FURBER S., BAINBRIDGE J.: 'Future trends in SoC interconnect'. Proc. 2005 Int. Symp. System-on-Chip, Tampere, Finland, November 2005, pp. 183–186

[2] MARCULESCU R., MARCULESCU D., PILEGGI L.: 'Toward an integrated design methodology for fault-tolerant, multiple clock/voltage integrated systems'. Proc. IEEE Int. Conf. Computer Design (ICCD), San Jose, CA, October 2004

[3] WOODS J.V., DAY P., FURBER S.B., *ET AL.*: 'AMULET1: an asynchronous ARM microprocessor', *IEEE Trans. Comput.*, 1997, **48**, pp. 375–398

[4] BAINBRIDGE J., PLANA L.A., FURBER S.B.: 'The design and test of a smartcard chip using a CHAIN self-timed network-on-chip'. Design, Automation and Test in Europe Conf. and Exhibition Designers' Forum (DATE'04), 2004, p. 30274

[5] BURSKY D.: 'We must hold the line on soaring ASIC design costs', *Electron. Des.*, 2002, **50**, PART 21, pp. 22–25, www.elecdesign.com

doi: 10.1049/iet-cdt:20060175

[6] BENINI L., DE MICHELI G.: 'Networks on chips: a new SoC paradigm', *IEEE Comput.*, 2002, **35**, (1), pp. 70–80

[7] BERTOZZI D., BENINI L., DE MICHELI G.: 'Error control schemes for on-chip communication links: the energy-reliability tradeoff', *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2005, **24**, (6), pp. 818–831

[8] BHOJWANI P., SINGHAL R., CHOI G., *ET AL.*: 'Forward error correction for on-chip networks'. Proc. Workshop for Unique Chips and Systems (UCAS-2), March 2006

[9] MURALI S., THEOCHARIDES T., VIJAYKRISHNAN N., *ET AL.*: 'Analysis of error recovery schemes for networks on chips', *IEEE Des. Test Comput.*, 2005, **22**, (5), pp. 434–442

[10] LI L., VIJAYKRISHNAN N., KANDEMIR M., IRWIN M.J.: 'Adaptive error protection for energy efficiency'. 2003 Int. Conf. Computer Aided Design (ICCAD'03), San Jose, CA, USA, 9–13 November 2003, pp. 2–7

[11] LEHTONEN T., LILJEBERG P., PLOSILA J.: 'Online reconfigurable self-timed links for fault tolerant NoC', *VLSI Des.*, 2007, **2007**, p. 13

[12] SHIVAKUMAR P., KISTLER M., KECKLER S.W., *ET AL.*: 'Modeling the effect of technology trends on the soft error rate of combinational logic'. Proc. Dependable Systems and Networks (DSN), 2002, pp. 389–398

[13] PARK D., NICOPOULOS C., KIM J., *ET AL.*: 'Exploring fault-tolerant network-on-chip architectures'. Proc. 2006 Int. Conf. Dependable Systems and Networks, June 2006, pp. 93–104

[14] ALI M., WELZL M., ZWICKNAGL M., *ET AL.*: 'Considerations for fault-tolerant network on chips'. Proc. 17th Int. Conf. Microelectronics (ICM), 2005

[15] DUMITRAS T., MARCULESCU R.: 'On-chip stochastic communication'. Proc. Design Automation and Test in Europe (DATE), March 2003

[16] PIRRETTI M., LINK G.M., BROOKS R.R., *ET AL.*: 'Fault tolerant algorithms for network-on-chip interconnect'. Proc. ISVLSI, 2004

[17] COPPOLA M., CURABA S., GRAMMATIKAKIS M.D., *ET AL.*: 'OCCN: a NoC modeling framework for design exploration', *J. Sys. Archit.*, 2004, **50**, pp. 129–163

[18] WIKLUND D., LIU D.: 'SoCBUS: switched network on chip for hard real time embedded systems'. Int. Parallel and Distributed Processing Symp. (IPDPS'03), France, 2003, p. 78a

[19] HEO S., ASANOVIC K.: 'Replacing global wires with an on-chip network: a power analysis'. Int. Symp. Low Power Electronics and Design (ISLPED'05), San Diego, CA, August 2005

[20] OGRAS U.Y., HU J., MARCULESCU R.: 'Key research problems in NoC design: a holistic perspective'. Int. Conf. Hardware–Software Codesign and System Synthesis, September 2005

[21] BAINBRIDGE J., FURBER S.B.: 'Chain: a delay-insensitive chip area interconnect', *IEEE Micro*, 2002, **22**, (5), pp. 16–23

[22] EDWARDS D.A., BARDSLEY A.: 'Balsa: an asynchronous hardware synthesis language', *Comput. J.*, 2002, **45**, (1), pp. 12–18

doi: 10.1049/iet-cdt:20060175