Embedded Transitive Closure Network for Run-Time Deadlock Detection in Networks-on-Chip

Ra’ed Al-Dujaily, Student Member, IEEE, Terrence Mak, Member, IEEE, Fei Xia, Alex Yakovlev, Senior Member, IEEE, and Maurizio Palesi, Member, IEEE

Abstract—Interconnection networks with adaptive routing are susceptible to deadlock, which could lead to performance degradation or system failure. Detecting deadlocks at run-time is challenging because of their highly distributed characteristics. In this paper, we present a deadlock detection method that utilizes run-time transitive closure (TC) computation to discover the existence of deadlock-equivalence sets, which imply loops of requests in networks-on-chip (NoC). This detection scheme guarantees the discovery of all true-deadlocks without false alarms in contrast with state-of-the-art approximation and heuristic approaches. A distributed TC-network architecture, which couples with the NoC infrastructure, is also presented to realize the detection mechanism efficiently. Detailed hardware realization architectures and schematics are also discussed. Our results based on a cycle-accurate simulator demonstrate the effectiveness of the proposed method. It drastically outperforms timing-based deadlock detection mechanisms by eliminating false detections and, thus, reducing energy wastage in retransmission for various traffic scenarios including real-world application. We found that timing-based methods may produce two orders of magnitude more deadlock alarms than the TC-network method. Moreover, the implementations presented in this paper demonstrate that the hardware overhead of TC-networks is insignificant.

Index Terms—Networks-on-chip, deadlock detection, dynamic programming, transitive closure computation, performance analysis.

1 INTRODUCTION

Networks-on-Chip (NoC) emerged as an on-chip communication architectural paradigm that applies networking theory and methods to VLSI systems incorporated on a single silicon chip [1], [2]. Such an architecture consists of a network constructed from multiple point-to-point data links, namely channels, interconnected by routers. The routers are connected to a set of distributed Intellectual Properties cores (IPs). Communication among these distributed IPs usually utilizes a packet-switching method where messages are divided into suitably-sized blocks, called packets and flits. The packets can be relayed from any source-destination communication pairs over several data links, by making routing decisions at routers. NoCs bring remarkable improvement in performance, flexibility, scalability, and power efficiency over conventional bus interconnections. However, in NoCs communication deadlocks may appear and cause an impasse in the network, and hence lead to performance degradation or system failure.

1.1 Generalities on Deadlock

A deadlock is a situation in NoCs wherein two or more packets are waiting for one another to release channels, and are unable to make progress. The network is deadlocked if a chain of waiting packets forms a cycle. As a simple deadlock example, consider the dependency cycle of four packets shown in Fig. 1. Each of the four packets in the figure holds some channels but it cannot proceed further until it acquires another channel currently held by one of the rest of the packets. However, no packet can release the channel needed by an other until it acquires its own requested channel. The packets are deadlocked and will remain in this state halting all the occupied channels until some intervention brings them back to life by breaking the dependency cycle. Deadlocks can paralyze network communications by halting the resources occupied by the deadlocked packets which in return could increase other packets blocking [3]. Hence it is crucial to remove deadlocks. There are two strategies to deal with deadlocks: avoidance and recovery [4], [5].

In deadlock avoidance, resources are granted to packets in a way that the overall network is deadlock free. This can be based on a turn-model which prohibits the routing algorithm from making certain turns in the network [6], [8] or based on the strict ordering of virtual channels [4]. In general, avoidance techniques require restricted routing functions or additional resources to prevent deadlocks. Due to its simplicity the turn-model is popular in NoCs, even though it limits the routing alternatives and diminishes fault tolerance capabilities. Also the turn-model is not applicable to an arbitrary network topology [12].

Alternatively, deadlock recovery implies that resources are granted to packets without any routing restrictions,
potentially outperforming deadlock avoidance [7], [9], [10], [11]. Deadlocks may occur and efficient detection and recovery mechanisms are required to intervene. However detecting deadlocks in a network is challenging because of their distributed nature. Heuristic approaches, such as timeout mechanisms, are often employed to monitor the activities at each channel for deadlock speculations. These techniques may produce substantial false detections, especially with the network close to saturation where blocked packets could be flagged as deadlock. Several techniques have been proposed for reducing the number of false detections in general computer networks [10], [11], [14] but all of them are based on the timeout idea and finding best threshold values for different network settings is not trivial.

To recover from deadlock, there are two kinds of deadlock recovery schemes: regressive and progressive. A regressive recovery is based on the abort-and-retry mechanism [9], which kills the suspected packets. A progressive recovery uses additional hardware in each router node (central buffers) to bypass the suspected packets to their destination sequentially [7] or concurrently [13]. The supplementary document to this paper contains a RELATED WORK section which covers the existing work on deadlock recovery strategy. Also it contains Table S1 which summarizes the advantages and disadvantages of the existing deadlock handling strategies.

In this work and in line with [14], we present the effectiveness of our proposed detection method and compare it with existing work in deadlocks detection [7], [10], [11]. Our emphasis is not on comparing with avoidance techniques. Hence, a simple abort-and-retry approach is employed in this study. Thus our primarily target is analyzing the deadlock detection rate and not the overall behavior of a deadlock recovery strategy (detection + recovery), which depends on both detection and recovery.

1.2 Contributions of the Paper

Unlike general computer networks, where internode information can only be exchanged through packets, on-chip networks can take advantage of additional dedicated wires to transmit control data between routers.

This paper exploits this NoCs-specific capability and proposes a new deadlock detection method, which guarantees true deadlock detection. A run-time transitive closure (TC) computation scheme is employed to discover the existence of deadlock-equivalence sets, which imply loops of requests. Also, the proposed detection scheme can be realized using a distributed architecture, which closely couples with the NoC infrastructure, to speed up the necessary computation and to avoid introducing traffic overhead in the communication network. The major contributions of this paper are:

- Introducing a deadlock-equivalence set criterion for detecting loops of packet requests.
- Presenting a new deadlock detection scheme to discover the existence of deadlock-equivalence sets based on TC computation. In addition a distributed architecture, TC-network, is proposed to implement the detection technique.
- Evaluating the proposed deadlock detection scheme performance through experimental studies and comparisons with the timing-based detection schemes [7], [10], [11] using various traffic scenarios.
- Evaluating the hardware area and power overhead of the TC-network.

Preliminary results of this study and a sketch of the proposed architecture were presented in [16]. In here, a complete theoretical framework of the transitive closure computational approach is presented. Hardware architecture for the framework realization is detailed and new results on real-life applications are included.

2 Methodology

2.1 Background and Assumptions

The network of concern in this work is a collection of router nodes connected by channels. Each router is connected to a single IP core that can inject/consume packets via the router. A fully adaptive routing algorithm with minimal paths is used; minimal in this context means that the routing algorithm always chooses the shortest path between sender and receiver. A wormhole flow control technique [17] is employed which has been widely used in NoCs. The packets in wormhole flow control are divided into smaller flits. A header flit is routed first and the rest will follow it in a pipeline manner and that will allow a packet to occupy several channels simultaneously. Thus, wormhole reduces the number of buffers required in each router which is a desirable feature for NoCs. However wormhole makes networks more prone to blocking and deadlock [3], [18].

In this paper we study NoCs without any deadlock avoidance techniques, i.e. not imposing any restriction on routing function and not using virtual channel ordering, but adopting deadlock recovery with an accurate deadlock detection method (proposed in the next Section). In line with existing work on deadlock recovery [7], [10], [11], [14], we assume that a channel buffer...
cannot contain flits belonging to different packets and a packet arriving at its destination is eventually consumed.

2.2 Equivalence Set Criterion for Deadlock

To introduce our proposed method for deadlock detection in NoCs, we first introduce a Deadlock-Equivalence Set (DES) criterion for detecting loops of packet requests. Then we use the transitive closure computation to determine whether there is a set of channels in the NoCs forming a DES.

A network $I(O,V)$ is a strongly connected directed graph, where $O$ is a set of vertices that represent the set of router nodes and $V$ is a set of ordered pairs of edges represent the set of channels that connect the routers. The network resource ownership and request at any particular time can be expressed as a Channel Wait-for Graph (CWG) [4] $G = (V,E)$ where $V$ the set of channels and $E$ the set of the edges. The graph is readily translated to an $n \times n$ Adjacency Boolean matrix $A$ as follow: either 1) $A_{ij} = 1$ if and only if there exists an edge in $G$ between vertex $i$ and vertex $j$ or 2) $A_{ij} = 0$ otherwise (including when $i = j$, thus no channel has the same network node as both its source and destination).

Given a directed graph $G$ it is possible to answer reachability questions, i.e. can one get from node (vertex) $u$ to node $v$ in one or more hops, using the concept of Transitive Closure (TC). The TC of $G$ is a derived directed graph, $G^+ = (V,E^+)$, which contains an edge $(u,v)$ if and only if there is a directed path from $u$ to $v$ in one or more hops. Now let $V = \{v_1,v_2,\ldots,v_n\}$ be the channels from network $I$ and consider a subset $M = \{v_1,v_2,\ldots,v_m\}$ of channels for some $m < n$. $M$ is a Deadlock-Equivalence Set (DES) if and only if in all its channels there are flits waiting for one another in a cyclic manner to progress to their respective destinations, i.e. $v_1$ occupied by a flit and it requests $v_2$, $v_2$ occupied by a flit and it requests $v_3$, $\ldots$, $v_m$-1 occupied by a flit and it requests $v_m$ and a flit in $v_m$ requests $v_1$.

The TC computation can then be used to determine whether there is a set of channels in the network forming a DES. Suppose we have channel “a” and channel “b”, where $a, b \in V$. If these two channels form a deadlock-equivalence set $\Omega(a,b)$, their corresponding TC will be $T_{ab} = T_{ba} = 1$ and $T_{aa} = T_{bb} = 1$. This can be extended to a subset $M = \{v_1,v_2,\ldots,v_m\}$ of channels for some $m < n$, such that all pairs of elements in $M$ meet the self-requesting condition:

$$\Omega(a,b) = \begin{cases} 1, & \text{if } T_{ab} = T_{ba} = T_{aa} = T_{bb} = 1, \forall a, b \in M \\ 0, & \text{otherwise} \end{cases}$$ (1)

The supplementary file to this paper contains a more elaborated example about DES detection (Example S1). A channel can appear in a DES only once and cannot appear in multiple DES’s at the same time. Thus, members of a set of simultaneous DES’s, $S = \{S_1\}$, are pairwise disjoint, that is $S_i$ and $S_j$ are subsets of $S$ and $i \neq j$ implying $S_i \cap S_j = \phi$.

To recover from a deadlock situation the dependency cycle formed by the DES should be resolved. A deadlock can be resolved if one or more of the packets forming the deadlock is removed from the network [9], [10], [11]. Hence in this work a packet is removed from the network once it detected as deadlocked.

2.3 Equivalence Set Computational Complexity

The DES provides a simple criterion for deadlock detection. The technique is to generate the CWG from the network and then derive the TC of the CWG and identify the channels that satisfy the criterion. Fig. 2 illustrates this idea. Given a network at any particular state (Fig. 2a) the CWG can readily be drawn (Fig. 2b). The derived TC graph (Fig. 2c) clearly shows four vertices (channels) with self-reflexive paths and all pairs of these satisfy the condition of the DES (Eq.1).

The supplementary file to this paper contains Algorithm S1 which illustrates the TC computation. The computation of the TC is expensive. In software the algorithm requires a computational complexity of $O(n^3)$ [19], where $n$ is the number of vertices in the CWG. In hardware, however, there are many proposed previous works to speed up the transitive closure computation time to $O(n)$ using a systolic array of $O(n^2)$ processing elements [20] [21]. In this work, we used the NoC distributed architecture to simplify the complexity. We present in the next section a distributed architecture, TC-network, to realize the TC computation in NoCs.

3 TC-NETWORK ARCHITECTURE

3.1 Transitive Closure Computation with TC Networks

Dynamic programming (DP) can yield the solution for the transitive closure [19]. DP provides an opportunity for solving the computation using a parallel architecture with improved computation performance. Mapping TC computation to a parallel computational platform can be achieved with the introduction of a DP-network. The network has a parallel architecture, and can be used to compute the TC solution through the simultaneous propagation of successive inferences. Lam and Tong [22] introduced DP-networks to solve a set of graph optimization problems with an asynchronous and continuous-time computational framework. This new class of inference networks is inherently stable in all cases and has been shown to be robust with an arbitrarily fast convergence rate [22]. A parallel computational network for solving the dynamic routing problem for NoCs is also proposed in [23]. In this work we introduce a TC-network to discover the existence of deadlock-equivalence sets at run-time. Similar to the DP-network [22], [23] the proposed TC-network can be realized using a distributed architecture, which closely couples with the NoC infrastructure.

The TC-network is constructed by the interconnection of autonomous computational units. The structure of a
Such realization. Such network architecture encompasses node and a link represents a communication channel. A way as the NoC structure. Each unit represents a router computational units will be interconnected in the same unit which resolves the binary relation in Fig. 3. Each unit represents a binary relation between two objects and there are S
\[ S_k(i, j) = g(i, k) \land g(k, j) \] (2) \[ g(i, j) = v_{\forall k}S_k(i, j) \] (3) where \( \land \) (AND function) is the inference for the site function and the conflict-resolution operator for the unit function. The operator \( \lor \) (OR function) denotes the unit which resolves the binary relation \((i, j)\). The computational units will be interconnected in the same way as the NoC structure. Each unit represents a router node and a link represents a communication channel. A distributed network can readily be implemented using such realization. Such network architecture encompasses the advantage of simplicity and parallelization, which presents a great opportunity to be applied for on-chip network management.

The delay of TC-network convergence to deadlock-detection depends on the size of the DES and the network topology, which both determines the delay of information propagation within the network, and the delay of each computational unit. As it can be seen that each unit involves \( O(|A|) \) AND operations and one OR operation where \(|A|\) is the number of adjacent edges. Hence, the solution time is \( O(k|A|) \) where \( k \) is the number of iterations evaluated by each unit. In a software computation, \( k \) equals to the number of nodes in the network which guarantees that all nodes have been updated. However, in the hardware implementation with parallel execution, \( k \) will be determined by the network structure and \(|A|\) AND operations can be executed in parallel. Each computational unit can simultaneously compute the new expected output for all neighbor nodes. The delay of TC-network convergence to deadlock detection is directly proportional to the size of the DES, which determines the delay of information propagation within the TC-network and the delay of TC computational units. To investigate the worst case delay for the TC-network to converge to the deadlock detection we analyzed several popular network topologies in the NoCs literature. After excluding the topologies that inherently are deadlock free, e.g., tree, bus, star, butterfly, etc. Table 1 shows the longest DES for several network topologies as a function of the network size \((k)\). Consider the \( k \)-ary 1-cube (ring) topology. The maximum number of channels in such a network is \( 2k \) while the longest and the only possible DES is \( k \) as this will be the longest dependency cycle. Likewise for \( k \)-ary 2-mesh network topology (2D mesh) of \( k^2 \) nodes with \( k \) rows and \( k \) columns the shortest DES in this network will span over four nodes while the longest DES will be \( 3k^2 - 3k - 2 \) and thus, the detection time will depend on the network topology and on how deadlocks are distributed over the network nodes.

3.2 Coupling TC-Networks to NoCs

An on-chip communication network itself defines the graph vertices of its TC-network. This provides an opportunity for TC computation embedding a TC-unit at each node. The TC-network shown in Fig. 4 consists of distributed computational units and links between them. The topology of the network resembles the defined

---

**Fig. 2.** (a) A network scenario of deadlock formation; (b) the CWG of the network; (c) the transitive closure of the CWG, the set of channels \( \{ch_1, ch_3, ch_7, ch_9\} \) satisfy the DES definition (Eq.1).

**Fig. 3.** Unit interconnection in a general TC-network where \( 1 \leq i, j, k \leq n; k \neq i, j \) [22]
Performing exhaustive TC calculation for the entire network will discover concurrently all the channels in the network that are members of the same DES and will add recovery overhead by requesting to release all channels. Back to Fig. 2c, the TC graph exhibit four self-reflexive paths around $ch_1, ch_5, ch_7$ and $ch_9$ and all of them are members of the same DES (satisfy Eq. 1). To reduce the hardware complexity of TC-networks we can solve the problem by finding a path in multiple-source single destination network. In this case, it is possible to employ a light-weight network, a few logic gates and a few wires, based on mutual exclusion units to implement the TC-network to discover the existence of a self-reflexive path for each channel. The mutual exclusion circuit ensures that only one channel can use the TC-network at any time and channels are checked sequentially using a simple token-ring protocol.

One possible token-ring path for mesh and torus networks is the one presented in [7], a Hamiltonian cycle. The token-ring could be implemented as an asynchronous circuit [7] or could be clocked using router clocks. However the synchronous implementation is not a desirable option as it requires the same clock to span the whole chip. In this work we implement asynchronous token ring protocol circuits however this choice requires a synchronizer circuits to achieve the synchronization with the TC-unit in each router node.

Each TC-unit seizing the token will initiate checking of the corresponding router channels. The rest of the units will implement a transitive property that passes the test signal to neighbor units if and only if there is a chain of channel dependencies between its input and output.

This makes the TC-network self-pruning and will keep switching power to minimum. The function performed in the TC-unit that seizes the token is described in Algorithm 1. It starts checking each channel in the router node. In case a self-reflexive path is detected for a particular channel, line 10, it means the channel is part of a DES and that channel corresponding deadlock flag is set (line 11) which may be used by the router node to trigger a recovery. Otherwise the deadlock flag is reset and the next channel will be tested. Once the TC-unit finishes checking all the channels of the corresponding router, it passes the token to the next neighbor unit. The supplementary file augmented this paper with the following: (1) Example S2 which illustrates, with high level description, how to detect the deadlock configuration presented in Fig. 2a using Algorithm 1, (2) HARDWARE IMPLEMENTATION section which details the hardware architecture realization.

4 RESULT AND DISCUSSIONS
4.1 Evaluation Methodology
We evaluate the TC-network approach by comparing the percentage of detected deadlocks with the heuristic timeout mechanism [7] for different network traffics and different Packet Injection Rate (PIR). The percentage of detected deadlocks is obtained as the ratio between the number of packets detected as deadlock over the total number of packets (received and detected). Once a packet is presumed as deadlocked, it is removed from the network to resolve the deadlock situation. We evaluate the consumed energy to resolve all detected
deadlocks using the proposed method and the timing-based methods. In the result section the energy consumed to resolve deadlocks is presented as a percentage to the total communication energy (energy consumed by normal transmissions + energy consumed to resolve deadlocks). The energy model adopted to measure the communication energy in this paper is similar to the one used in [25] which segregate the energy dissipated by the NoC into two parts, the router circuit and the inter-router wire links. Specifically, the NoC energy dissipated is divided into the following categories: 1) Routing energy which depends on the routing type; 2) Selection energy which refers to the type of selection if the routing algorithm returns more than one option; 3) Forwarding energy which is used in sending a flit; 4) Receiving energy which is used in receiving a flit and 5) Waiting energy which is related to the time the header flit remains waiting until a successful routing take place. Hence, we define the consumed energy to resolve detected deadlocks \(E_{\text{dd resolve}}\) as:

\[
E_{\text{dd resolve}} = h \times (E_{\text{forward}} + E_{\text{receiving}}) + Flit_{\text{age}}
\]

where \(E\) denotes energy, \(h\) is the number of hops the flit passed before being aborted, \(Flit_{\text{age}}\) is the number of clock cycles the flit lived in the network (either moving or waiting resources to be freed) and \(f\) is a Boolean flag that adds the last term to the equation if the flit type is head, i.e. if the blocked flit is head it will continue consuming energy by trying to reserve an output channel in each clock cycle.

One of the main problems of timeout schemes is that they strongly depend on message length. However, some of the previous work, like [11], shows that their detection scheme is less dependent on the message length. In this paper to demonstrate that the proposed TC-network based detection method is completely independent of the message length we evaluated them with random message lengths. However this choice makes the timing-based deadlock detection schemes look worse and the reader should be aware of that.

The performance evaluation was carried out using a modified version of Noxim [24]. In particular, we modified Noxim by introducing the TC-network and the timing-based deadlock detection techniques [7], [10], [11]. We chose a mesh with four port architecture, a fully adaptive routing with random selection function, no virtual channels, and a crossbar switch - the last three can be used in a wide range of NoC configurations. Each input channel consists of four flit buffers and one clock cycle is assumed for routing and transmission time across the crossbar and a channel. The result are captured after a warm up period of 10,000 clock cycles. The overall simulation time is set to 300,000 clock cycles. To ensure the accuracy of results captured with a higher confidence, the simulation at each PIR is repeated many times with different seeds and their mean values taken.

4.2 Evaluation Result

Fig. 5 shows the performance results for a 4 × 4 2D mesh NoC, and a uniform distribution of packet destinations\(^1\) with different injection rates. The packet lengths are randomly generated between 2 and 16 flits. The majority of detected deadlocks using the timeout mechanism [7] are false alarms. For instance, 22% of the packets injected in the network at higher injection rates are detected as part of deadlocks with the threshold value set to 32 (Fig. 5a). The TC-network instead detected that less than 1% of packets are in true deadlocks, consistent with literature [3] which stated that deadlock is an infrequent event. Fig. 5b shows the network average delay versus the throughput over the entire load range. It shows that smaller threshold values used in the timeout mechanism improve these two important network metrics because

\(1\) This kind of traffic pattern is the most commonly used traffic in network evaluation [5] although it is very gentle because it naturally balances the load all over the network.

---

Algorithm 1 Pseudo code of the TC-unit computation

**Input:** \(tc_{\text{rx}}[n]\) TC signals from upstream routers, 
**Input:** \(ch_{\text{oc}}[m]\) channels occupation from local routers, 
**Input:** \(ch_{\text{reg}}[m][n]\) channels request from local routers, 
**Input:** \(ch_{\text{sel}}[m]\) channel to check for a self-reflexive loop; input from channel selection circuit. 
**Output:** \(tc_{\text{tx}}[m]\) TC signals to downstream routers, 
**Output:** \(dd_{\text{flags}}[m]\) deadlock detection flags to local routers.

**Definitions:** 
par: denotes parallel operations, 
\(n\): number of input channels to the local routers, 
\(m\): number of output channels to neighbor routers, 
\(tp[m]\): temporary buffers.

1: \(\text{par: for } i = 1 \to m \text{ do} \)
2: \(\text{if } ch_{\text{sel}}[i] \text{ then} \)
3: \(tc_{\text{tx}}[i] = 1 \)
4: \(\text{end if} \)
5: \(tp[i] = 0 \)
6: \(\text{par: for } j = 1 \to n \text{ do} \)
7: \(tp[i] = tp[i] \lor (tc_{\text{rx}}[j] \land ch_{\text{oc}}[j] \land (ch_{\text{reg}}[j][i] = 1)) \)
8: \(\text{end par: for} \)
9: \(\text{if } ch_{\text{sel}}[i] \text{ then} \)
10: \(\text{if } tp[i] \text{ then} \)
11: \(dd_{\text{flags}}[i] = 1 \)
12: \(\text{else} \)
13: \(dd_{\text{flags}}[i] = 0 \)
14: \(\text{end if} \)
15: \(\text{else} \)
16: \(dd_{\text{flags}}[i] = 0 \)
17: \(tc_{\text{tx}}[i] = tp[i] \)
18: \(\text{end if} \)
19: \(\text{end par: for} \)
it detects more false deadlocks and by dropping them
will alleviate network congestion that may lead to con-
gestion. Moreover, resolving the detected deadlocks by
merely dropping the detected packets without retrans-
mission increases the NoC consumed energy but adds
no significance in terms of the network throughput and
latency. In the future we are planning to integrate our
proposed deadlock detection method with an efficient
recovery mechanism to evaluate the NoC performance
under different deadlock handling strategies, i.e. avoid-
ance and recovery.

which repeatedly try to route the head flit till the time
out value has elapsed in the case of the timeout method
(see Eq.4). For instance, the Timeout-128 detects 5.3%
at saturation and it waste 8.8% of the total consumed
energy by aborting these packets while the Timeout-512
detects 2.6% at saturation and wastes 9.1% energy. There
are two reasons behind that: the first one is that the
network with bigger threshold value is delivering fewer
flits at the given simulation time (Fig. 5b). The second
reason is that the Flit_age in Eq.4 is directly proportional
to the threshold value, in the case when a packet is
detected as deadlocked.

To investigate different traffic scenarios and network
sizes, Fig. 6 (a, b and c) shows the performance results
with the bit-reversed traffic pattern\(^2\). The network size
is 8 × 8 mesh and the packet sizes are randomly chosen
between 32 and 64 flits. The results in general show a
similar trend to the previous example. The TC-network
method detects around 0.07% of packets as deadlocked
and dropping them consumed energy of less than 0.26%
compared to 4% the least detected using Timeout-1024
[7] and wasting energy of around 10%.

Moreover, we implemented the deadlock detection
techniques proposed in [10] and [11], which were pro-
posed to enhance the accuracy of deadlock detection
over the crude timeout method [7], i.e. to reduce the
false-positive-deadlock alarms. The mechanism pre-
sented in [10] measures the inactivity time of all the
output channels requested by a blocked packet. To fur-
ther reduce the number of false deadlock detections,
the mechanism presented in [11] is intended to identify
only one packet in a sequence of blocked packets as
deadlocked. Our proposed method, TC-network, and the
timing-based deadlock detection methods [7], [10], [11],
are evaluated in the rest of this section. The timeout
threshold for each simulated technique is selected as
16, 32, 64, 128, 256, 512, and 1024 clock cycles, but only
the results for 64 and 256 cycles are presented, since
the only major difference is the measures scaling up or
down.

Fig. 7 (a, b, c and d) shows the performance results
with the shuffle traffic\(^3\). The NoC size is 16 × 16 mesh
and the packet sizes are randomly chosen between 32
and 64 flits. The result of the crude timeout mechanism
is not shown in the figure as it detects more than 70% of
the packets as deadlock and hence the focus is on com-
paring the TC-network performance with the other two
elegant timeout techniques [10], [11]. The results dem-
strate the effectiveness of the TC-network. It drastically
outperforms the most elegant timing-based techniques,
presented in [10] and [11], by eliminating false detections
and, thus, reducing energy wastage in resolving these
detected deadlocks. Fig. 7d shows the total energy con-
sumed by the NoC. It clearly shows that the NoC in case
of employing the timing-based methods consumes more

\(^2\) Each node with binary address \(\{a_{n-1}, a_{n-2}, \ldots, a_0\}\) sends a
packet to the node with address \(\{b_0, b_1, \ldots, b_{n-1}\}\).

\(^3\) This kind of traffic might be induced by sorting applications [5].
A network starts saturating when an increase in injection rate does not result in a linear increase in throughput [25].

2 illustrates the efficacy of the proposed run-time deadlock detection method over the three different existing timing-based methods [7], [10], [11]. The TC-network method significantly surpasses timing-based deadlock detection mechanisms by avoiding false detections and, as a result, reducing energy wastage to resolve all detected deadlocks for all synthetic traffic scenarios presented in the table. It should be noted that the quantitative analysis presented in the last row of Table 2 as “TC-improvement” would vary according to injection rate (PIR) and it is used to summarize, in average, by what factor the TC-network reduces the detected deadlocked packets compared to the timing-based schemes [7], [10], [11]. Therefore, the TC-improvement magnitude is not necessarily the same in case the evaluation incorporated the entire load range. The results presented are for 8 × 8 mesh with packet sizes randomly chosen between 32 and 128. For a real-world application, the supplementary file to this paper contains experiment results for a Generic Multi-Media System.

4.3 Area and Power Estimation

It is crucial in designing NoCs that routers should not consume a large percentage of silicon area compared to the IP core blocks. We have designed in Verilog two fully adaptive routers based on the timeout and the TC-network methods. These are then synthesized using Synopsys Design Compiler and mapped onto the UMC 90nm technology library. Confirming well-known findings from, for instance, [25] and [26], our hardware synthesis result shows that the First In First Out (FIFO) buffer area significantly dominates the logic of the router. The buffer area mainly depends on the flit size. For a flit size of 64 bits and FIFO buffers with a capacity of four flits, we found that the TC circuit adds only 0.71% area overhead to the total router area compared to 2.9% for the timeout [7] implementation with 10 bit threshold counter. Thus the proposed method yields an area gain of more than 2% in each network router circuit compared to the timeout implementation.

Power consumption is also an important system performance metric. We determine the power dissipated in each router design by running Synopsys Design Power on the gate-level netlist of the router with different random input data streams as test stimuli. We found that the power dissipated by the TC-unit is 0.44% of the entire router power compared to 0.97% dissipated by the crude timeout circuit, with 10 bit threshold counter. This gives a power saving of 0.53%. In the Intel TeraFlop 80-tiles chip implementation [27] the communication power (Router + Links) estimation is 28% of the tile power profile. The Router power, however, is 83% of the communication power. Taking these numbers in consideration and the power result from proposed router synthesis will suggest that the TC-unit will dissipate less than 0.01% of the total tile power in similar NoC implementations.
Moreover we investigated the area and power contributions of different timing-based deadlock detection methods circuits with different time-threshold values and compared them to the TC-unit circuit implementation. The area gain and the power saving with the TC-unit implementation compared to different timeout implementation can be observed from Table 3. The table clearly shows that the techniques used in [10] and [11] introduce more area and power overhead compared to the crude timeout implementation and this is due to the extra hardware requirements to implement these techniques. The TC-network implementation however, not only improve the performance of the deadlock detection method but also minimize the area and power overheads as can be seen in Table 3.

However, the implementation of the TC-network requires some extra control wires, as shown in Fig. S1 in the supplementary file. The \( tc_{rx}[n] \) input wires and \( tc_{tx}[m] \) output wires to/from each TC-unit are coming-going to TC-units in neighbor router nodes. The \( token_{in}, token_{ack\ out}, token\_{out} \) and \( token\_{ack\ in} \) signals are used to propagate asynchronously the token, \( i.e. \) changing the destination node, in the token ring protocol implemented in this work. Overall, the total number of wires required to support the deadlock detection based on TC-network is

\[
TC\_Network\_wires = n + m + 4.
\]

where \( n \) equal to the number of input channels and \( m \) equal to the number of output channels in each router nodes. Considering the data used in the experiments (2D mesh network where \( n, m \in \{ North, East, South, West \} \) and flit size of 64 bits), we have \( TC\_Network\_wires = 12 \).
TABLE 3
Area and power contributions of the TC-unit and different timeout circuits to the total router area and power

<table>
<thead>
<tr>
<th>Module name</th>
<th>Proposed in</th>
<th>Module area to total router area</th>
<th>Module power to total router power</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC-unit</td>
<td>This work</td>
<td>0.71%</td>
<td>0.44%</td>
</tr>
<tr>
<td>Timeout-1024</td>
<td>[7]</td>
<td>2.93%</td>
<td>0.97%</td>
</tr>
<tr>
<td>Timeout-256</td>
<td>[7]</td>
<td>2.33%</td>
<td>0.81%</td>
</tr>
<tr>
<td>Timeout-64</td>
<td>[7]</td>
<td>1.56%</td>
<td>0.70%</td>
</tr>
<tr>
<td>Timeout-1024</td>
<td>[10]</td>
<td>3.12%</td>
<td>1.16%</td>
</tr>
<tr>
<td>Timeout-256</td>
<td>[10]</td>
<td>2.54%</td>
<td>0.98%</td>
</tr>
<tr>
<td>Timeout-64</td>
<td>[10]</td>
<td>1.76%</td>
<td>0.91%</td>
</tr>
<tr>
<td>Timeout-1024</td>
<td>[11]</td>
<td>3.53%</td>
<td>1.51%</td>
</tr>
<tr>
<td>Timeout-256</td>
<td>[11]</td>
<td>2.94%</td>
<td>1.38%</td>
</tr>
<tr>
<td>Timeout-64</td>
<td>[11]</td>
<td>2.12%</td>
<td>1.24%</td>
</tr>
</tbody>
</table>

4.4 Delay Estimation
In order to study the operation delays, first the TC-unit’s critical path gates delay is calculated using the sdf file generated after synthesizing the circuit using worst case library. Second the interconnect delay calculation assumes the tiles are arranged in a regular manner on the floorplan with 2nm × 1.5nm tile size, similar to Intel TeraFLOPS chip [27]. The maximum interconnect length between routers is 2 mm. The load wire capacitance and resistance are estimated using the Predictive Technology Model (PTM) [28] to be 0.146 fF and 1.099 Ohm per micron respectively. The wire delay between TC units, TC-interconnect, can be readily calculated based on the distributed RC model [29]. In reference to Section 3.2 and Table 1 in the main paper the worst and average convergence times of the TC-network for different network topologies can be estimated. Fig. 8 shows the worst and average times to discover a deadlock for different network topologies and sizes. It is expected that the delay required by the TC-network to converge to the desire output will depend on the network topology and the size of the DES.

Fig. 8. TC-network convergence time for different network topologies and sizes

5 CONCLUSION AND FUTURE WORK
This work studies deadlock recovery strategy in NoCs as opposed to deadlock avoidance. We proposed a new deadlock detection method based on computing deadlock-equivalence sets at run-time. Also, transitive closure network architecture is proposed to realize the detection computation in a distributed way. The new method eliminates the need of any kind of timeout mechanism, and delivers true deadlock detections independent of the network load and message lengths rather than approximating with congestion estimation as in the existing methods. The proposed method is rigorously evaluated using a cycle-accurate simulator to demonstrate the effectiveness of the TC-network based detection method compared to the timeout mechanisms. In the future, we plan to investigate the methodology of using TC-network for real-time deadlock detection and integrate it with efficient recovery mechanisms in a large-scale three-dimensional Networks-on-chip.

REFERENCES


Ra'ed Al-Dujaily B98-M01 received both B.Sc. and M.Sc. degrees in control and computer engineering from the University of Technology of Baghdad/Iraq in 1998 and 2001, respectively. He is currently working toward the Ph.D. degree with the School of Electrical, Electronic and Computer Engineering, Newcastle University, UK. His research interests include Network-on-Chip, reconfigurable computing. VLSI circuits design and fault tolerance in VLSI.

Fei Xia is a Senior Research Associate with the School of Electrical, Electronic, and Computer Engineering, Newcastle University, UK. His research interests include the design, modelling, and analysis of microelectronic systems focusing on asynchronous systems and data communication in such systems. Dr Xia holds a BEng in Automation from Tsinghua University, an MSc in Control Systems from the University of Alberta, and a PhD in Electronic Engineering from King's College London.

Alexandre (Alex) Yakovlev received the MSc and PhD degrees from St. Petersburg Electrical Engineering Institute in 1979 and 1982, respectively, where he worked in the area of asynchronous and concurrent systems since 1980 and the DSc degree from Newcastle University in 2006. In the period between 1982 and 1990, he held positions of assistant and associate professor at the Computing Science Department. Since 1991, he has been at the Newcastle University, where he worked as a lecturer, reader, and professor at the Computing Science Department until 2002, and is now heading the Microelectronic Systems Design Research Group (http://async.org.uk) at the School of Electrical, Electronic and Computer Engineering. His current interests and publications are in the field of modeling and design of asynchronous, concurrent, real-time, and dependable systems on a chip. He has published four monographs and more than 200 papers in academic journals and conferences, has managed over 25 research contracts. He has chaired program committees of several international conferences, including the IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), Petri nets (ICATPN), Applications of Concurrency to Systems Design (ACSD), and is currently a chairman of the Steering committee of the Conference on Application of Concurrency to System Design. He is a senior member of the IEEE and a member of the IET. In April 2008, he was general chair of the 14th ASYNC Symp. and 2nd Int. Sympo. on NoCs, and tutorial chair at Design Automation and Test in Europe (DATE) in 2009.

Maurizio Palesi (M06) Maurizio Palesi received the M.S. and Ph.D. degrees in computer engineering from the University of Catania, Italy, in 1999 and 2003, respectively. Since November 2010, he is Assistant Professor at Kore University, Enna, Italy. His research focuses on platform-based system design, design space exploration, low-power techniques for embedded systems, and network-on-chip architectures. His main research interests include design space exploration methodologies, networks-on-chips, and low power design. Dr. Palesi serves on the Editorial Board of very large scale integration (VLSI) design as an Associate Editor since May 2007. He has served as a Guest Editor for the VLSI Design Journal - Special Issue on Networks-on-Chip in 2008, and as a Guest Editor for the International Journal of High Performance Systems Architecture - Special Issue on Power-Efficient, High Performance General Purpose and Application- Specific Computing Architectures in 2009. He serves as the Technical Program Committee Member for the following IEEE/ACM International Conferences: RTAS, CODES + ISSS, ESTI-Media, SOCC, VLSI, ISC, and SITIS. He was also the Co-organizer of international workshops on NoC architectures in 2008, 2009, and 2010.

Terrence Mak (S05-M09) received both B.Eng. and M.Phil. degrees in Systems Engineering from the Chinese University of Hong Kong in 2003 and 2005, respectively, and the Ph.D. degree from Imperial College London in 2008. He joined the School of Electrical, Electronic and Computer Engineering at Newcastle University as a lecturer in 2010. During his Ph.D., he worked as a Research Engineer Intern in the VLSI group at Sun Microsystems Laboratories in Menlo Park, California. He also worked as a Visiting Research Scientist in the Poons Neuroengineering Laboratory at MIT. He was the recipient of both the Croucher Foundation Scholarship and the US Navel Research Excellence in Neuroengineering in 2005. In 2008, he served as the Co-Chair of the UK Asynchronous Forum, and in March 2008 he was the Local Arrangement Chair of the Fourth International Workshop on Applied Reconfigurable Computing. His research interests include FPGA architecture design, Network-on-Chip, reconfigurable computing and VLSI design for biomedical applications.