Average and Worst-Case Latency Improvements in Mixed-Criticality Wormhole Networks-on-Chip

Leandro Soares Indrusiak
Department of Computer Science,
University of York, UK.
Email: leandro.indrusiak@york.ac.uk

James Harbin
Department of Computer Science,
University of York, UK.
Email: james.harbin@york.ac.uk

Alan Burns
Department of Computer Science,
University of York, UK.
Email: alan.burns@york.ac.uk

***Abstract*—Mixed-criticality applications executing over a multiprocessor platform based on Network-on-Chip (NoC) exchange packets of different criticality levels through the same communication infrastructure, and transmission of a packet has potential impact over the latency of all the others. This paper presents NoC architectural improvements to output port arbitration and mode change signalling. The first aim is to improve the average latency of low-criticality packets following a mode change by allowing NoC arbiters to service them during cycles in which no high-criticality flows can be transmitted. The second aim is to reduce the worst-case latency of high-criticality packets transmitted by the NoC. The former objective improves the system's responsiveness, while the latter contributes to increased resource efficiency. The achieved improvements are evaluated, respectively, by cycle-accurate simulation and by schedulability analysis, showing full delivery of low-criticality packets following a criticality change, and achieving full schedulability in 8.2% more flowsets than the state of the art.**

I. Introduction

Mixed-criticality systems (MCSs) are characterised by two potentially conflicting objectives: the need for logical separation between application components of different levels of criticality, and the need for resource sharing for increased utilisation and efficiency. In such systems, for instance, safety-critical components might share the same hardware infrastructure with non-critical functionality. As the timing behaviour of the latter is often less well-understood than that of the former [8], which is more thoroughly analysed and tested, special care should be given to prevent a temporal anomaly on the execution of the non-critical component from negatively impacting the performance of the safety-critical part of the system, which could have catastrophic consequences.

When it comes to MCSs running over Network-on-Chip (NoC) platforms, this usually means that the NoC should make sure that traffic of different criticality levels should not interfere with each other (i.e. have latency impact) while efficiently sharing its infrastructure: routers, links and buffers. We assume a less restrictive interpretation of this trade-off: as long as the NoC can enforce that interference between traffic of different criticality levels is predictable and bounded, and can it be shown that it will not violate any timing requirement of the system, it can be considered suitable for mixed-criticality systems.

In this paper, we review the existing work on NoCs supporting mixed-criticality traffic and extend the state-of-the-art WPMC by proposing WPMC-FLOOD with the following improvements:

1) Reduce the average latency of low-criticality (LO-crit) traffic by allowing it to carefully use idle NoC resources even when the system has detected a high-criticality situation.
2) Reduce the worst case latency of high-criticality (HI-crit) traffic by improving the mode-change signalisation protocols and thus preventing interference from LO-crit traffic once the system has detected a high-criticality situation.

The first improvement is validated through cycle-accurate simulation of a NoC platform under mixed-criticality traffic scenarios, while the second is validated through the application of a novel, albeit incremental, schedulability analysis over a large set of synthetic applications.

II. Related Work

A number of NoC architectures and protocols were published recently, aiming to address the trade-off between resource sharing and separation of different criticality levels. All of them follow common NoC design choices such as mesh topology and wormhole switching, and differ mostly on link arbitration policies.

CompSOC [5] is an approach based on the AEthereal NoC [4] , aiming to support the joint execution of applications with different time criticality, and written under different models of computation (MoCs) such as Kahn Process Networks or Synchronous Dataflow. Their ultimate goal is to allow designers to individually validate each application according to the guarantees that are inherent to their underlying MoC, while transparently sharing a virtualised NoC platform. The key to the approach is time-division multiplexing (TDM), which allows the temporal isolation of components of different criticality level by granting them exclusive access to platform components for predefined time slots. While enabling a straighforward flavour of composability, TDM makes it harder to handle change or uncertainty regarding application behaviour, as most of the time slot allocation must be done at design time, and it tends to under-utilise the NoC resources due to the fact that applications don't always need all the time slots reserved to them.

IDAMC [7] is a NoC architecture that includes virtualisation and monitoring mechanisms to enable functional and non-functional isolation between applications of different criticalities. It supports address translation at the network interface (NI), allowing applications to be developed as if in isolation and using only locally-available resources. The access to remote resources is transparent and enabled by the NoC services themselves, which use virtual channels (VCs) to isolate traffic of different criticalities. Two approaches are supported: static isolation, when the arbitration of virtual channels is done at design time in order to guarantee a fixed amount of bandwidth to each VC in a TDM fashion, like CompSOC; or dynamic isolation, which uses a *back suction* technique [3]. Back suction tries to maximise the bandwidth given to low (or non) critical traffic by monitoring the progress of high-criticality traffic and making sure that all packets arrive by their deadlines, and not unnecessarily earlier. In other words, if a critical packet is arriving at its destination too early, the routers along its path will gradually reduce the amount of bandwidth given to its VC, therefore reducing the interference that it can cause to packets going through other VCs of lower criticality and thus improving their latencies in a best-effort fashion (i.e. no guarantees).

WPMC [10] is a protocol applied to NoCs with flit-level priority-preemptive VC arbitration, aiming to provide hard real-time guarantees to all criticality levels (i.e. all packets will arrive by their deadlines even in the worst-case scenario) and supporting sporadic as well as periodic traffic patterns. It follows Vestal's assumption [8] that application components of high criticality will be given more generous upper bounds for their timing behaviour, e.g. due to more strict analysis or to larger safety margins; and that components of low criticality, which are likely to be analysed with less strict techniques or which are given smaller safety margins, will have tighter upper bounds for their timing behaviour. The intuition behind that approach is the fact that highly critical application components must cope even with very unlikely timing behaviour (e.g. video frame compression will take more than 2 ms only once in a billion frames), thus the system must be dimensioned accordingly so that it can cope with those scenarios as well. In the case of non-critical components, those cases can be ignored, and the system can be dimensioned to cope only with the other e.g. 99.999999999% of the cases. In line with that approach, WPMC assumes that high-criticality traffic is likely to have potentially larger packets, or having packets injected more often into the NoC, as this would be a safer upper bound on the load it may impose to the NoC.

A key idea of WPMC, which was also used in the AMC scheduling algorithm [1], is that traffic of high criticality could also be analysed with the same techniques and safety margins used to profile low criticality traffic, and thus be given tighter upper bounds to its timing behaviour. The tight upper bounds can be used to dimension the NoC in such a way that all packets will always meet their deadline, as long as they don't exceed their low criticality upper bounds (i.e. maximum packet size, minimum packet inter-arrival interval). WPMC uses runtime monitoring to check whether all high-criticality traffic stays within their low-criticality upper bounds. The moment one of them exceeds that bound, the system is said to change into a high-criticality mode. To guarantee the timely delivery of all high-criticality packets under that mode, the NoC is allowed to drop all low-criticality traffic (as a way to achieve graceful degradation). Thus, to ensure the system is dimensioned to cope with the high-criticality mode, it must be able to support only the high-criticality traffic, but considering their more generous upper bounds. In [10], the authors define the NoC mechanisms to perform the runtime monitoring, signalise mode change, and to change the NoC arbitration policies to drop low-criticality traffic. They also provide schedulability analysis to evaluate whether a given NoC is properly dimensioned to cope with the traffic produced by a given (set of) application(s) under the default low-criticality mode, as well as during and after a change to the high-criticality mode is detected.

III. System Model

Let us now precisely define the models of mixed-criticality applications and NoC platforms used in this paper to describe our approach. To allow for a fair comparison with WPMC, which will be the baseline in our experimental work introduced here as WPMC-FLOOD, we follow mostly the same models and notation.

A. Application Model

Following WPMC, we take a communication-centric view of the system and focus on the traffic load imposed by the application on the NoC platform. Thus, a mixed-criticality application Γ comprises n real-time traffic-flows (or just *flows* for short) $\Gamma = \{\tau_1, \tau_2, \ldots \tau_n\}$. Each flow τ_i gives rise to a potentially unbounded sequence of *packets*. The flow has a set of properties and timing requirements which are characterised by a set of attributes:

τ_i = (P_i, C_i, T_i, D_i, J_i^D, J_i^I, IP^s, IP^d). All the flows which require timely delivery are either periodic or sporadic. The lower bound interval on the time between releases of successive packets is called the period (T_i) for the flow. The maximum basic network latency (C_i) is the maximum duration of transmission latency when no flow contention exists.

Each real-time flow also has a relative deadline (D_i) which is the upper bound restriction on network latency, assumed to be $D_i \leq T_i$. Any flow can suffer two forms of release jitter; J_i^D is direct jitter and denotes the maximum deviation of successive message releases from the flow's period. That is, a packet from τ_i will be released for transmission at most J_i^D time units after its periodic tick, e.g. due to the time it takes for its generating task to execute. The other form of jitter, J_i^I, is the indirect interference τ_i may suffer [6]. Here, even if some flow τ_k does not share any resources with τ_i, it can affect its behaviour by delaying another flow τ_j that directly interferes with τ_i (e.g. causing back-to-back interference of two τ_j packets). As shown in [6], a safe upper bound of J_i^I is R_i - C_i.

Each flow is also assigned, at design time, a criticality L_i, which can be high (HI) or low (LO). Three of the flow parameters have (potentially) HI-crit and LO-crit values, according to the strictness of the analysis and safety margins used to obtain them (as discussed in Section II): $C_i(LO)$, $C_i(HI)$, $T_i(LO)$, $T_i(HI)$, $J_i^I(LO)$, $J_i^I(HI)$, with the natural constraints: $C_i(LO) \leq C_i(HI)$, $T_i(LO) \geq T_i(HI)$, $J_i^I(LO) \leq J_i^I(HI)$.

In addition to these parameters, each flow has a priority P_i; the value 1 denotes the highest priority and larger integers denote lower priorities. It also has a source and destination IP core on the NoC (IP^s and IP^d).

B. Platform Model

We assume a NoC platform with deterministic routing (e.g. XY), wormhole switching, credit-based flow control and priority-preemptive virtual channels (VCs) [2] implemented as multiple FIFO buffers in the input port (Figure 1). To enable the mixed-criticality support, the NoC routers have two modes of operation: LO-crit (default) and HI-crit.

In the LO-crit mode, the arbitration of the output ports will always enable the transmission of the flit of the input VC with the highest priority which has credits (i.e. buffer space on the respective VC on the downstream router), regardless of their criticality. We assume that all flows of the same priority level will also have the same level of criticality, but we do not require that flows of higher priority must also have higher criticality. This means that flows of different criticality levels will always use different VCs, but it will not prevent a HI-crit flow from suffering interference from a LO-crit flow. In the example shown in Fig. 1 we have the flows with priorities 1 and 2 as LO-crit and the flows with priority 3 as HI-crit (as indicated in the figure besides each respective VC).

We also assume that the network interfaces (NI) of each NoC router can detect whether any packet injected into the NoC has exceeded any of its low-criticality parameters: $C_i(LO)$ or $T_i(LO)$. If they have, the NI signalises a mode change to its router as it injects the packet header, so the router changes its mode of operation to HI-crit. The mode change triggers two actions by the router: it changes the arbitration of the output ports to prevent HI-crit flows from suffering interference of LO-crit flows, and it signalises the mode change to the neighbouring routers. In the next section, we will provide more details on how those two actions were implemented in WPMC, and will propose improvements to both of them.

IV. Improving Average Latency of Low Criticality Traffic

In WPMC, LO-crit flows are not forwarded to the output port of a router after a criticality mode change. This prevents them from causing interference on HI-crit flows, a key requirement to the flow schedulability analysis proposed in [10]. The implementation of the WPMC protocol described in [10] enforces such behaviour through the output port arbitration mechanism of the NoC router. The output port arbiter decides

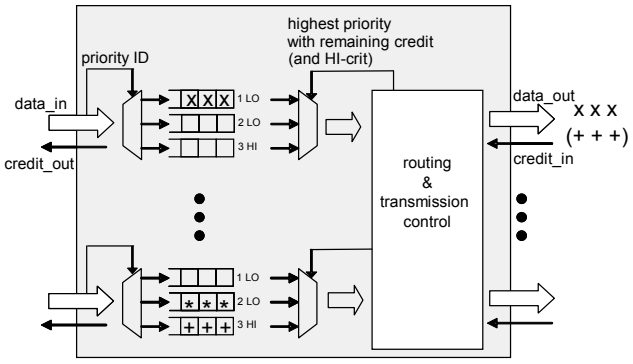


Fig. 1: NoC router with priority-preemptive arbitration and mixed-criticality support (HI-crit mode of operation shown in parenthesis)

which input port and which virtual channel will be the source of the flit forwarded through its respective output port at every cycle. When operating in the default LO-crit mode, the arbiter will, on each cycle, forward a flit from the input virtual channel of the highest priority which has credits, regardless of criticality. After a mode change, the arbitration rules change to prevent the forwarding of flits from LO-crit flows: it will then forward a flit from the input HI-crit virtual channel of the highest priority which has credits. Flits stored in buffers of LO-crit virtual channels will remain where they are after a mode change, as those virtual channels will not be considered ever again for arbitration in WPMC.

Fig. 1 shows a scenario where all packets stored in the input VCs are competing for the output port on the upper-right part of the figure. Under LO-crit mode, the highest-priority VC (the one with flits represented by "X") is given arbitration regardless of its criticality, and its flits are shown leaving the output port. However, under HI-crit mode, the arbitration rules change to consider only HI-crit packets (the additional arbitration rule for the HI-crit mode is shown in brackets). In that case, the VC who would be given arbitration is the one with flits represented by "+", and so its flits are shown leaving the output port (also in brackets).

To improve the service to LO-crit flows, we propose an enhancement to the arbitration of output ports after a mode change. We keep with the initial ordering of arbitration of the original WPMC, serving the HI-crit virtual channels that have credits in order of priority, but then we consider the LO-crit virtual channels that have credits in order of priority. By making sure that LO-crit virtual channels of any priority will only be considered after all HI-crit virtual channels, we can ensure that LO-crit flits will only be forwarded through the output port in the cycles when no HI-crit flows are transmitting. This will happen when the buffers of the HI-crit input virtual channels are all empty (i.e. there are no HI-crit flows to be forwarded during that cycle) or when the buffers of the HI-crit input virtual channels of the downstream router (the one connected to the output port in question) are all full (i.e.

no HI-crit virtual channels have credit, probably because of network congestion further downstream). Thus, our enhanced arbitration will only use links during periods when they would be otherwise left idle by the original WPMC. Consequently, no additional interference will be suffered by HI-crit flows, which means that the schedulability analysis proposed in [10] is also valid for a NoC with the proposed enhanced arbitration.

In the scenario shown in Fig. 1, the arbiter would allow flits of the packet represented by "X" to flow through the output port once the packet represented by "+" is finished, or even before that in case the buffer of the virtual channel 3 of the downstream router is full.

V. Improving Worst-case Latency of High Criticality Traffic

A. Motivation

Criticality mode change is a critical part of WPMC. When HI-crit flows exceed their LO-crit budget, the links which they go through will change their criticality level, hop by hop. Once a router receives a flit from a link in HI-crit mode, it immediately triggers a mode change and designates all its output links as HI-crit. That way, WPMC prevents LO-crit flows from interfering on HI-crit flows after a mode change. However, due to the nature of the protocol, which "piggybacks" the mode change notification on the flits of the flow that caused the mode change, it is possible that a flow that does not cause a mode change to cross two regions of the NoC, each at a different criticality level.

In those cases, the flow in question will suffer increased interference from HI-crit flows that have potentially exceeded their LO-crit budgets when going through the HI-crit region of the NoC, and will not suffer any interference from LO-crit flows in that region because those routers will not transmit LO-crit flows after the mode change. On the other hand, in the LO-crit region of the NoC, the packet will still suffer interference from LO-crit flows and HI-crit flows (within their LO-crit budgets), as those NoC routers are unaware of the mode change.

Figure 2 shows such a scenario. Flow τ_3, which goes from IP core *d* to IP core *p* through routers *4*, *8*, *12*, and *16*, is a HI-crit flow which has exceeded its LO-crit budget and therefore has caused a mode change in those routers as its first packet goes through them one by one. As a result, all their output links have been designated as HI-crit (appearing in bold in the figure). Flow τ_1 is also a HI-crit flow, but one that has not exceeded its LO-crit budget, so it does not cause mode changes in the routers it goes through (i.e. *1*, *2*, *3*, *4* and *8*, from its source IP *a* to its destination *h*). Some of those routers (e.g. 1 and 2) keep forwarding LO-crit traffic, for instance flow τ_2, which in this example has higher priority than τ_1 and therefore keeps causing interference despite of the mode change elsewhere in the NoC.

B. Review of NoC Schedulability Analysis

A standard criticality-unaware schedulability analysis for a set of flows in a wormhole switching NoC is presented in [6]. A set of flows is deemed schedulable if the worst-case response times of each flow is less than their deadline. The worst-case response time R_i of a flow τ_i can be obtained from Equation 1. This equation is defined recursively and iterated until a stable fixed point is discovered.

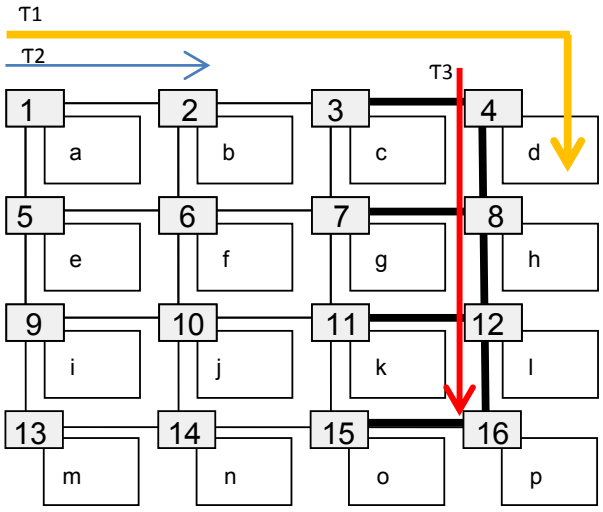


Fig. 2: Scenario with a traffic-flow τ_1 moving into a HI-crit region of the NoC (HI-crit links shown in bold)

$$R_i = C_i + \sum_{\tau_j \in \mathbf{Shp}(i)} \left\lceil \frac{R_i + J_j^D + J_j^I}{T_j} \right\rceil C_j, \quad (1)$$

The set $\mathbf{Shp}(i)$ is the set of higher priority flows which share a link with τ_i. The two jitter terms J_j^D and J_j^I are those described in Section III.

C. Review of Schedulability Analysis in WPMC

The WPMC schedulability analysis presented in [10] extends the schedulability analysis to wormhole mixed criticality NoCs, and considers the different interference patterns of each of the criticality regions of the NoC. Although full details are presented in [10], the fundamental details of this schedulability analysis are presented in this section.

The first goal of schedulability analysis is to ensure the flowset is correctly schedulable in the LO-crit mode, as performed for each flow τ_i in Equation 2. This approach is as defined in Equation 1, using the LO-crit mode parameters throughout.

$$R_i(LO) = C_i(LO) + \sum_{\tau_j \in \mathbf{Shp}(i)} \left\lceil \frac{R_i(LO) + J_j^D + J_j^I(LO)}{T_j(LO)} \right\rceil C_j(LO), \quad (2)$$

where

$$J_j^I(LO) = R_j(LO) - C_j(LO). \quad (3)$$

The HI-crit schedulability analysis considers three cases, with the response time of a flow defined by the worst of the three cases. The first case considered is the schedulability of a HI-crit flow that itself causes a mode change. Its response time is presented in Equation 4. The analysis uses the interference set $\mathbf{ShpH}(i)$; the set of HI-crit flows with higher priority and sharing a link with τ_i. Since it causes the mode change, this flow can suffer no interference from higher priority LO-crit flows. However, to be sure that the worst case is considered, it assumes that all other HI-crit flows have moved simultaneously to their HI-crit parameters:

$$R_i^a(HI) = C_i(HI) + \sum_{\tau_j \in \mathbf{ShpH}(i)} \left\lceil \frac{R_i^a(HI) + J_j^D + J_j^I(HI)}{T_j(HI)} \right\rceil C_j(HI), \quad (4)$$

The second case considers a flow which remains in the LO-crit mode, but which experiences an increase in indirect interference as a result of other flows switching to a HI-crit mode. The response time in this case is presented in Equation 5:

$$R_i^b(HI) = C_i(LO) + \sum_{\tau_j \in \mathbf{Shp}(i)} \left\lceil \frac{R_i^b(HI) + J_j^D + J_j^I(HI)}{T_j(LO)} \right\rceil C_j(LO). \quad (5)$$

The third case assumes that a flow (τ_i) going from a LO-crit to a HI-crit region will suffer continuing interference from LO-crit (high priority) packets from upstream links (i.e. before it enters the HI-crit region), but the interference from similar packets on downstream links, in the worst case, will be capped at $R_i^b(HI)$ as a mode change must have occurred before this time. Once τ_i has entered the HI-crit region of the NoC, all possible future links that τ_i uses will be in the HI-crit mode (because they already were, or because it will cause them to change as it goes through them). The following formulation, which appears as Equation 7 in [10], represents the worst case latency of a flow (τ_i) in such situation:

$$R_i^c(HI) = C_i(LO) + \sum_{\tau_j \in \mathbf{ShpH}(i)} \left\lceil \frac{R_i^c(HI) + J_j^D + J_j^I(HI)}{T_j(HI)} \right\rceil C_j(HI) + \sum_{\tau_j \in \mathbf{ShpUL}(i)} \left\lceil \frac{R_i^c(HI) + J_j^D + J_j^I(LO)}{T_j(LO)} \right\rceil C_j(LO) + \sum_{\tau_j \in \mathbf{ShpDL}(i)} \left\lceil \frac{R_i^b(HI) + J_j^D + J_j^I(LO)}{T_j(LO)} \right\rceil C_j(LO), \quad (6)$$

where $\mathbf{ShpUL}(i)$ is the set of higher priority LO-crit flows that share any link with τ_i and may be upstream of the flow that caused the mode change; and $\mathbf{ShpDL}(i)$ is the set of higher priority LO-crit flows that share any link with τ_i and are guaranteed to be downstream of the flow that caused the mode change. So all interfering LO-crit flows are represented by the union of $\mathbf{ShpUL}(i)$ and $\mathbf{ShpDL}(i)$. The latter set is computed conservatively to be only those LO-crit messages that are forced to be downstream from any HI-crit flow that could cause the mode change.

D. Modifications and Schedulability Analysis For WPMC-FLOOD

In this paper, we propose an enhancement to the criticality change notification mechanism. Upon receiving a flit through a HI-crit link, we propose that a router will actively propagate the mode change to all its neighbours through dedicated control signals instead of simply changing the mode of the local output links. These dedicated control signals operate so that when any incoming criticality control line is activated, the arbiter immediately activates all output criticality control lines leading to its neighbouring arbiters. This is important as it serves to bound the delay of criticality transmission to a minimal value, by decoupling propagation of the criticality change from the transmission of individual HI-crit flits across links. That way, the mode change will be propagated to the whole NoC, and not only downstream following the path of the flow that motivated the mode change. As a result, in a few cycles after the mode change is detected, the whole NoC will change to HI-crit mode and there is no need to consider the case of flows going across two criticality regions. A slightly different schedulability analysis can be used instead, since there is no need to take into account the LO-crit interference from upstream flows after a mode change is notified. In the worst case, the LO-crit interference will last for the time it takes for the notification to go across the NoC, which in the case of a 2D-mesh is the network diameter (in hops) over the operating frequency of the NoC.

With the proposed mode-change signalling, worst case latency of a flow (τ_i) becomes:

$$R_i^c(HI) = C_i(LO) + \sum_{\tau_j \in \mathbf{ShpH}(i)} \left\lceil \frac{R_i^c(HI) + J_j^D + J_j^I(HI)}{T_j(HI)} \right\rceil C_j(HI) + \sum_{\tau_j \in \mathbf{ShpUL}(i)} \left\lceil \frac{R_i(LO) + \alpha + J_j^D + J_j^I(LO)}{T_j(LO)} \right\rceil C_j(LO) + \sum_{\tau_j \in \mathbf{ShpDL}(i)} \left\lceil \frac{R_i^b(HI) + J_j^D + J_j^I(LO)}{T_j(LO)} \right\rceil C_j(LO), \quad (7)$$

where α is the maximum time needed to propagate a mode change to the whole NoC (which is the product of the NoC clock period and its diameter).

The proposed improvement on the signaling mechanism will require additional wires between routers. For only two criticality levels, as addressed in this paper, only one additional wire is needed on each link of the NoC, which has minimal overhead on energy dissipation and on the chip floorplanning

(each NoC link typically has 40-140 wires, depending on the flit width and the number of VCs, so this means an overhead between 0.7 and 2.5%). In the general case, the overhead percentage will be the same, and the number of additional links required for a MxN 2D mesh NoC is given by $(M-1).N+(N-1).M$ (which is the number of edges of an MxN lattice graph) multiplied by 2 (as NoCs have bidirectional links between routers), plus $2.M.N$ (to account for the bidirectional links between each router and its local core's NI). Notice that the overhead does not increase with the number of VCs of each router. It increases, of course, with the number of supported criticality levels $NCRIT$ by a factor of $\lceil \log_2 NCRIT \rceil$.

VI. Experimental Work

This section discusses the experimental work performed to validate the experimental protocol WPMC-FLOOD, consisting of an evaluation of the analytic equations with synthetic flowsets, and a cycle-accurate simulation executed using a real application case study.

A. Analytic Evaluation

The intent of this section is to assess the schedulability advantage of the worst-case latency improvements generated by the mode-change signalling protocol WPMC-FLOOD. Schedulability is assessed by producing test synthetic flowsets from a flowset generator, configured by default using the parameters specified in Table I. These flowsets are then evaluated comparing four different approaches for scheduling equations and priority assignments, to determine the proportion of schedulable flowsets in each. The first approach is a baseline case without criticality awareness. A second approach is a baseline using the previous WPMC equations defined in [10]. Thirdly, the experimental case is the WPMC-FLOOD mixed-criticality scheduling equations proposed in Section V. These first three evaluation cases use deadline monotonic priority assignment. A fourth approach is the baseline case with criticality monotonic priority assignment. For criticality monotonic priority assignment, all HI-crit flows are ensured to be higher priority than all LO-crit flows, and deadline monotonic priority assignment is used within each criticality level.

Compared to the standard WPMC protocol [10], the protocol proposed in this paper is able to suppress interference from LO-crit flows globally in the event of a mode change, allowing it to schedule additional flowsets which would not be schedulable under standard WPMC. However, it is not strictly the case that the WPMC-FLOOD equations dominate WPMC due to the inclusion of the additional time delay term α for propagation of the mode change requests (in Equation 7). Nevertheless, under the practical scenarios produced in which α is of the order of NoC cycles while flow periods are of the order of milliseconds, then the protocol provides an advantage in the practical scenarios evaluated. Therefore, the improvement that the protocol provides to worst-case latency of HI-crit flows can be assessed firstly by ensuring that it can schedule at least as many flowsets as WPMC together with an additional advantage, and secondly by a stress test validation showing that given certain characteristics in the flowset interference structure, it can produce a more significant advantage.

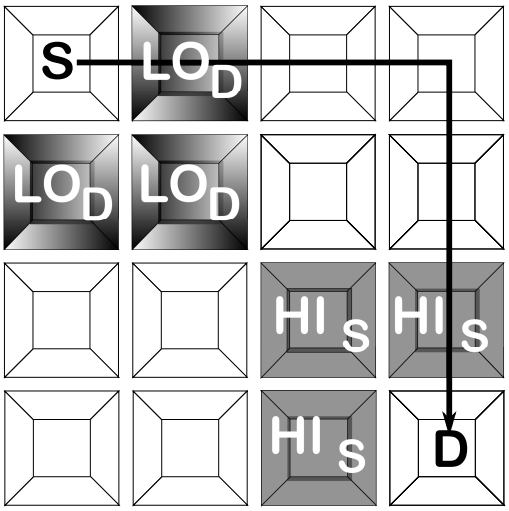


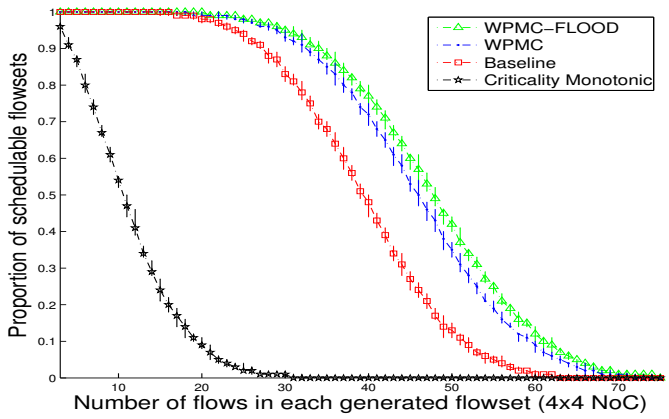
Fig. 3: The structure of stress testing flowsets

Two flowset generation modes are used in performance evaluation, to examine protocol performance in a general randomised source/destination case, and in a case in which a specific structure to the flowsets exists. In the *standard flowset* mode, a fixed number of flows are assigned to a randomly chosen source and destination node, with each node equiprobable for selection as a source or destination. Each flow is also assigned a randomly chosen criticality (aiming to produce an equal proportion of high and low-criticality flows). Flow periods are assigned according to a log-uniform distribution in which latencies are a random proportion of the period. Deadline monotonic priority assigment is used to assign priorities to the flows. Since the deadlines are equal to the flow period, shorter period flows have smaller deadlines.

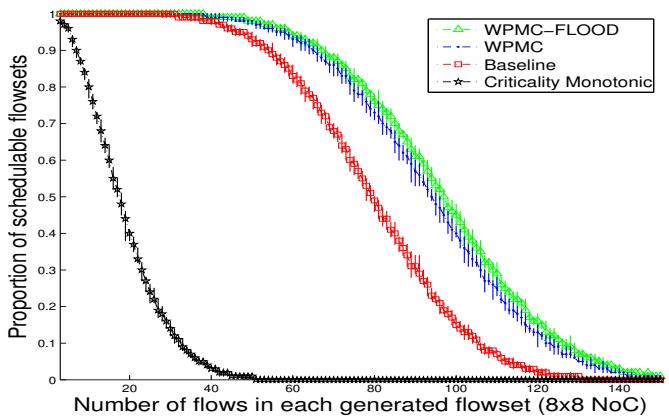
In the *stress test* flowset generation mode, a specific structure to flow interference relationships exists, as illustrated in Figure 3. A single long HI-crit flow exists which crosses the NoC from the top left node S to the bottom right corner node D. Interfering upstream LO-crit flows are clustered around the top left corner, all distributed with their source at S and their destination as one of the nodes in the LO_D set. HI-crit flows which suffer interference from this flow are clustered around the bottom right, with their source at one node in HI_S and their destination at D.

The WPMC analytic equations from [10] are compared with the WPMC-FLOOD equations, which compute $R_i^c(HI)$ via Equation 7 from Section V. As described in Equation 7 an additional time delay is added to $R_i(LO)$ in order to account for the delay of propagating the criticality change. The maximum delay in cycles α is equal to the length of the interfering route that triggers the mode change, to account for possible delays inherent in propagating the criticality change along the signalling wires between two arbiters.

The results of the equation evaluation for the standard and stress testing flowset structures are shown in Figures 4 and 5 respectively. For the standard flowset structure, two distinct NoC sizes of 4x4 and 8x8 are used. For the stress testing structure, only the 4x4 case is used since the stress testing structure produces similar interference set relationships regardless of NoC size. The results show that the modified analytic equations for WPMC-FLOOD achieve higher schedulability than either the criticality-unaware baseline case or WPMC. The peak improvement in the proportion of flowsets successfully scheduled over WPMC is 8.2% in the standard flowset generation case. In the stress testing flowsets, in which the physical arrangement ensured upstream LO-crit flows causing indirect interference upon HI-crit flows, the magnitude of peak schedulability improvement is increased to 19.5%. The highest number of flows possible in schedulable flowsets is lower in the stress testing flowsets than in the standard flowset structure, since the stress testing concentrates interference, effectively producing additional congestion around the single long HI-crit flow.



(a) 4x4 NoC



(b) 8x8 NoC

Fig. 4: Flowsets schedulable under scheduling equations (standard flowset structure)

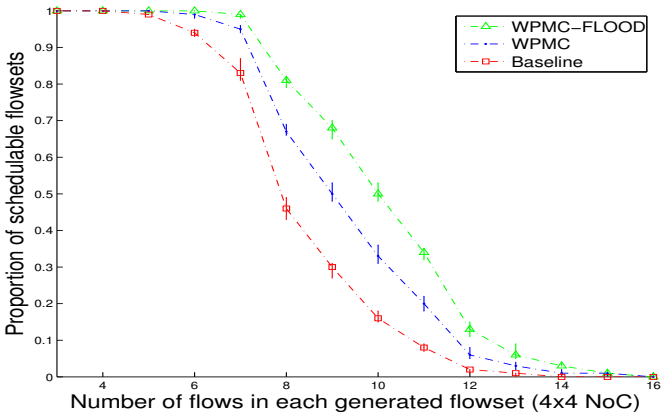


Fig. 5: Flowsets schedulable under scheduling equations (stress testing flowset structure)

Parameter	Default flowsets	Stress testing flowsets
Flowsets tested per trial	1000	1000
Trials per flowset size	10	10
Period range (ms)	1-1000	1-1000
HI-crit probability	0.5	0.5
Max. ratio C_{LO} to period	0.15	0.15
$C(HI)$ to $C(LO)$ ratio	2	2

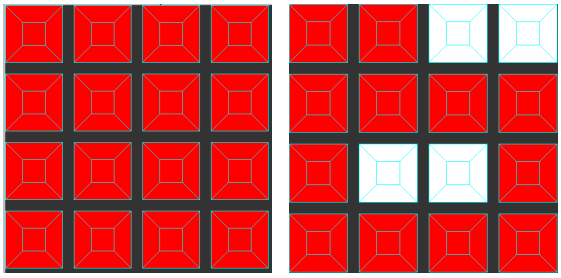
TABLE I: Parameters used in the protocol evaluation for default and stress testing flowsets

B. Cycle-Accurate Simulation

1) Simulation Introduction: The simulation execution examples validate WPMC-FLOOD via cycle-accurate simulation. The cycle-accurate simulation model provides an implementation of NoC routers incorporating buffering, arbitration and interconnecting links, together with abstract task schedulers and packet generators modelling the IP cores. Three alternative NoC designs are contrasted in the simulation results. A criticality-unaware baseline is provided in which arbitration decisions are always made upon flow priority, so no special status is afforded to HI-crit flows. A second baseline comparison is with the WPMC protocol [10]. In WPMC, criticality changes are not propagated globally throughout the NoC but are inherited through the propagation of HI-crit flows, and routers in HI-crit mode never forward LO-crit traffic following a mode change. WPMC-FLOOD is the experimental protocol.

2) Scenario Definition: Simulations are performed using a real application case study, the autonomous vehicle (AV) application [9]. The AV application models the complete execution involving navigation, video processing and object database management for an autonomous vehicle. The application periodically transmits 38 communicating flows, with HI-criticality status assigned to flows 23, 24, 34, 31, 36 and 37. A full definition of the communication patterns, flow periods and task mappings used is given in [10].

3) Simulation Results: During simulation execution of the experimental protocol, the communication latency of each packet is recorded. These latencies are categorised and grouped for each priority level, with maximum, minimum and mean computed per priority level. These max-min-mean



(a) WPMC-FLOOD protocol GUI (b) WPMC protocol GUI

Fig. 6: Criticality mode status of individual arbiters during execution (after criticality changes)

distributions are compared against a baseline implementation of the WPMC protocol [10], and to a further baseline cycle-accurate simulation which implements a simulation model of a conventional priority-preemptive NoC without any awareness of criticality modes. If a packet does not complete transmission under a particular protocol (because criticality-aware arbitration rules prevent transmission) then the packet receives an undefined latency.

The criticality mode of individual arbiters is recorded and displayed on a GUI during simulation execution. A red color indicates that the core has changed to HI-crit mode, while white indicates that it is still in the LO-crit mode. Figure 6a illustrates the simulation GUI in the experimental protocol, indicating that several cycles after the transmission of the first HI-crit packet exceeding its LO-crit budget the entire NoC has successfully changed to HI-crit mode. In contrast, in the WPMC case (Figure 6b), the criticality changes are only propagated forwards along the advancing routes of HI-crit flows causing or propagating the criticality change, so not all arbiters change to the HI-crit mode even after the final criticality change occurs.

Figure 7 presents the maximum, minimum and mean latencies per flow for the AV application in both baseline and experimental cases. For all application cases, the time interval considered is that following the final criticality change in the system. The HI-crit flows are emphasised by a grey background surrounding them. Although the application consists of 38 flows, flows with priority 37 and 38 have a very long period and are not transmitted by the application after the criticality change, and therefore they are not present on the graph for any protocol. In the WPMC baseline as specified in [10], very few of the LO-crit flows are successfully delivered to their destinations following the criticality change, with the majority being unable to progress and held up within the buffers. They therefore exhibit undefined latency and are not illustrated on the graph. Flows that experience undefined latency under WPMC are marked with a `u` symbol above their flow priority index. This is intended and necessary behaviour for WPMC, in that these LO-crit flows must not progress through the NoC in case their transmissions disrupt the schedulability of HI-crit flows.

For the current experimental protocol WPMC-FLOOD, the HI-crit flows are demonstrated to exhibit identical average latency compared to the WPMC baseline. In WPMC-FLOOD, several LO-crit flows experience slightly higher latencies than in the criticality-unaware baseline, due to the preferential selection of HI-crit traffic for arbitration in the case of contention. In WPMC-FLOOD, several LO-crit flows (such as priority 22) experience significantly higher latency than WPMC, since WPMC causes interfering LO-crit higher priority flows to be interrupted.

However, the most important contribution is that in the experimental protocol WPMC-FLOOD, all flows are delivered successfully, since no LO-crit traffic is dropped but must merely wait behind HI-crit traffic in case of contention. Therefore, this amounts to a reduction in the average latency across all LO-crit traffic, since many LO-crit flows would otherwise receive a large and undefined latency.

Figure 8 shows the statistics of the number of packets delivered to their destination under the various protocols throughout simulation execution (that is, before and after criticality changes). The HI-crit flows are marked with asterisks. This figure demonstrates that all application messages are delivered successfully regardless of criticality under the experimental protocol. Although WPMC successfully delivers all HI-crit flows, aggregate statistics show that only 20.5% of total application messages were successfully delivered to the destination.

VII. Conclusions and Future Work

This paper has proposed WPMC-FLOOD as an improvement over state-of-the-art mixed-criticality NoC protocols, detailing changes to NoC output port arbitration and mode-change signalling protocols. In the previous state of the art WPMC [10], many LO-crit flows would not be serviced following a mode change and would be held in buffers potentially indefinitely. Therefore, in practical terms the effect of the arbitration policy alteration is the reduction of average latency of all LO-crit traffic. Although individual LO-crit traffic flows may have longer latencies than WPMC, WPMC-FLOOD permits LO-crit traffic to still be forwarded towards its destination even in the case of a criticality mode change, enabling a degradation of service that is more graceful than the original specification of WPMC. It would however be possible to modify WPMC from its original definition to include this improvement in arbitration policy, improving the service of LO-crit traffic without any alteration to the WPMC mode change signalling protocol.

The outcome of the new mode-change signalling protocol is a potential reduction in the worst-case latency of HI-crit traffic-flows, which in turn can allow for increased resource utilisation: a platform configuration that was not deemed to be safe with the previous approach could be made safe simply by implementing the proposed signalling mechanism. The evaluation of analytic scheduling equations has demonstrated that the new mode-change protocol can successfully schedule

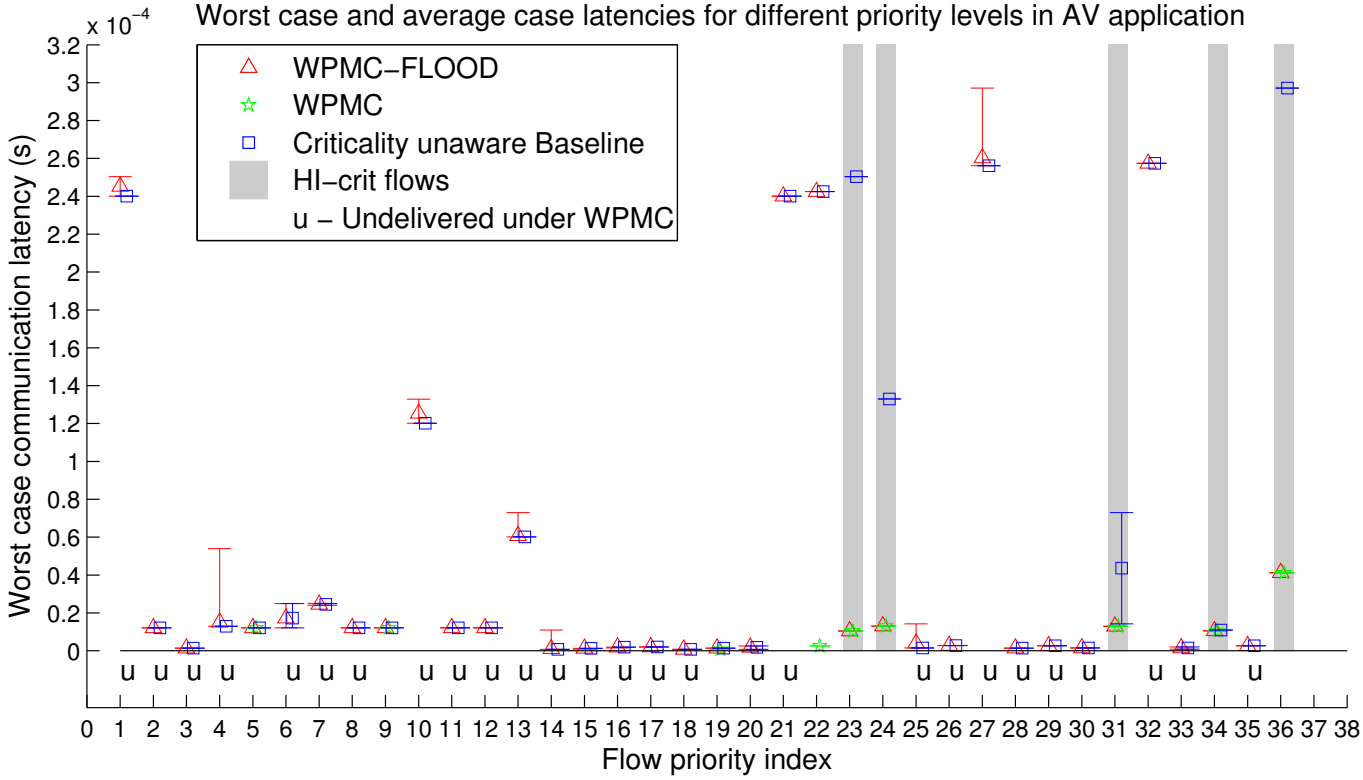


Fig. 7: Latencies for AV application under the experimental and baseline cases (following final criticality change)

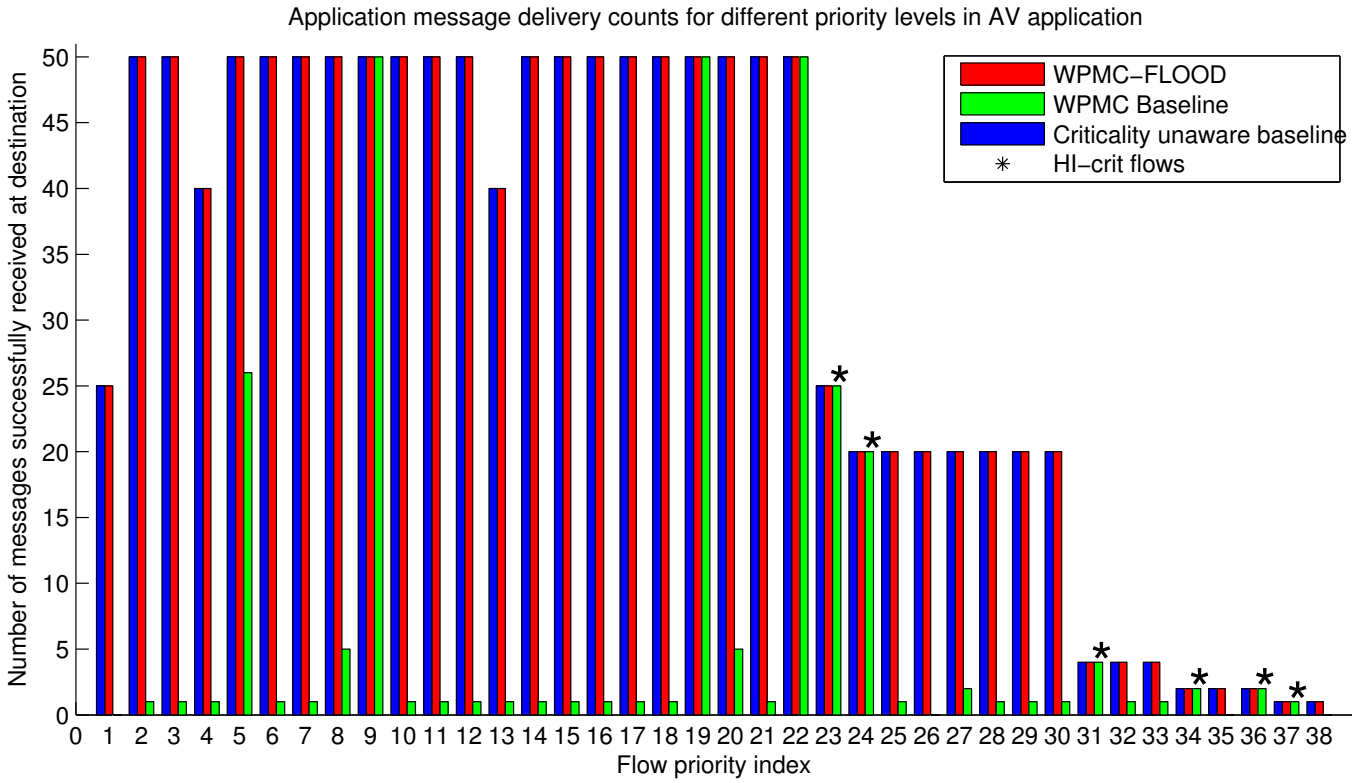


Fig. 8: Packet delivery counts for the AV application under the experimental and baseline cases

all flowsets that can be scheduled by the state-of-the-art protocol WPMC, together with a peak improvement of 8.2%, and in an additional improvement of 19.5% in a stress testing flowset structure. In addition, a cycle-accurate simulation has demonstrated that the new mode-change protocol successfully delivers all LO-crit flows to their destinations, unlike the previous state-of-the-art in which they must be supressed to guarantee schedulability.

Future work will include support of more than two levels of criticality, the study of mixed-criticality end-to-end latency analysis (i.e. considering task execution as well as traffic-flows), and the protocols supporting a mode change from HI-crit back to LO-crit.

Acknowledgements

The research described in this paper is funded, in part, by the EPSRC grant, MCC (EP/K011626/1). No new primary data were created during this study.

REFERENCES

[1] S.K. Baruah, A. Burns, and R. I. Davis. Response-time analysis for mixed criticality systems. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 34–43, 2011.

[2] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. QNoC: QoS architecture and design process for network on chip. *J. Syst. Archit.*, 50(2-3):105–128, 2004.

[3] J. Diemer and R. Ernst. Back suction: Service guarantees for latency-sensitive on-chip networks. In *Proc. 2010 Fourth ACM/IEEE Int. Symp. Networks-on-Chip*, Proc. NOCS '10, pages 155–162. IEEE Computer Society, 2010.

[4] K. Goossens, J. Dielissen, and A. Radulescu. Aethereal Network on Chip: concepts, architectures, and implementations. *IEEE Des. Test. Comput.*, 22(5):414 – 421, 2005.

[5] K. Goossens, A. Azevedo, K. Chandrasekar, M.D. Gomony, et al. Virtual Execution Platforms for Mixed-time-criticality Systems: The CompSOC Architecture and Design Flow In SIGBED Rev. 10(3):23–24, 2013.

[6] Z. Shi and A. Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Proc. of the 2nd ACM/IEEE International Symposium on Networks-on-Chip(NoCS)*, pages 161–170, 2008.

[7] S. Tobuschat, P. Axer, R. Ernst, and J. Diemer. IDAMC: A NoC for mixed criticality systems. In *Proc. RTCSA*, 2013.

[8] S. Vestal. Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In *Proc. of the IEEE Real-Time Systems Symposium (RTSS)*, pages 239–243, 2007.

[9] Z. Shi, A. Burns, and L.S. Indrusiak. Schedulability analysis for real time on-chip communication with wormhole switching. *Int Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, 1(2):1–22, 2010.

[10] A. Burns, J. Harbin and L.S. Indrusiak. A Wormhole NoC Protocol for Mixed Criticality Systems In *Proc. of the IEEE Real-Time Systems Symposium (RTSS)*, pages 184–195, 2014