Efficient deadlock avoidance for 2D mesh NoCs that use OQ or VOQ routers

Philippos Papaphilippou, Thiem Van Chu

Abstract—Network-on-chips (NoCs) are currently a widely used approach for achieving scalability of multi-cores to many-cores, as well as for interconnecting other vital system-on-chip (SoC) components. Each entity in 2D mesh-based NoCs has a router responsible for forwarding packets between the dimensions as well as the entity itself, and it is essentially a 5-port switch. With respect to the routing algorithm, there are important trade-offs between routing performance and the efficiency of overcoming potential deadlocks. Common deadlock avoidance techniques including the turn model usually involve restrictions of certain paths a packet can take at the cost of a higher probability for network congestion. In contrast, deadlock resolution techniques, as well as some avoidance schemes, provide more path flexibility at the expense of hardware complexity, such as by incorporating (or assuming) dedicated buffers.
This paper provides a deadlock avoidance algorithm for NoC routers based on output-queues (OQs) or virtual-output queues (VOQs), with a focus on their use on field-programmable gate-arrays (FPGAs). The proposed approach features fewer path restrictions than common techniques, and can be based on existing routing algorithms as a baseline, deadlock-free or not. This requires no modification to the queueing topology, and the required logic is minimal. Our algorithm approaches the performance of fully-adaptive algorithms, while maintaining deadlock freedom.

Index Terms—FPGA, NoC, SoC, deadlock avoidance, VOQ, OQ, NoC router, turn model

1 Introduction

A network-on-chip (NoC) is an interesting and diverse approach for interconnecting a high number of computing entities. With the increase of the number of entities in today's processors and their heterogeneity, NoCs have an increasing presence in research. This includes field-programmable gate arrays (FPGA), where NoCs are applied to prototyping, CGRA implementation [1] or simply for connecting systems of smaller logic components.

One of the fundamental challenges in NoCs are deadlocks, and this is usually solved at the routing level or/and at the flow control level. This work focuses on the former. At the routing level, deadlocks are avoided by ensuring that the paths produced by the routing algorithm do not form any cycles. At the flow control level, deadlocks are avoided by preventing router buffers from being allocated to packets in a way such that a dependency cycle of packets is formed. Note that in practice, this distinction may be less clear, as it can relate more directly to implementation, such as to divide the router logic into pipeline stages [2].

While the NoC research mostly focused on virtual channels (VCs) for buffering, in FPGAs and embedded systems, it is also common to use NoC designs with routers based on output-queued (OQ) switches [3], [4] or input-queued (IQ) switches with virtual output queues (VOQs) [5], [6]. These have a queue for every input-output combination (see figure 2). As a result, each router in the nodes of such NoCs has a queue per pair combination of the 5 directions ({E, S, W, N, C} for east, south, west, north and centre respectively), assuming a 2D-mesh topology.

Limitations in state-of-the-art: While there is a plethora of research on deadlock freedom in NoCs with routers based on virtual channels (VCs), NoCs that have a VOQ-like queue organisation still rely on simpler routing algorithms to achieve deadlock avoidance. This is because there is no theoretical background to increase path freedom for approaching full-adaptivity. While it is possible to adapt some methods from VCs, they are costly to apply, such as by introducing numerous additional queues for implementing/emulating escape channels. This limitation has an impact on routing performance, while the studied queue organisation remains popular among FPGA and embedded applications [3], [4], [7].

Insights: The main idea of this paper is that in NoC routers with VOQ-like queue organisation (one queue per input-output pair), deadlock avoidance can be achieved with more path flexibility than traditional models by *calculating a worst case occupancy of specific queues*. This is based on the observation that *this queue topology already implies turn information, which can be exploited to relax the turn model* for building more flexible deadlock-free routing algorithms.

Motivation: In order to demonstrate the importance of path flexibility, figure 1 presents simplified results from a 8 × 8 NoC simulation with output-queues under 3 routing algorithms. The first algorithm is dimension order routing (DOR) which limits 4/8 of the possible turns, the second is "north last" which forbids 2/8 of the available turns. The third one, however, does not have any turn limitation, but has a potential for creating deadlocks. The traffic model for this example was purposely selected to produce lower probability for deadlocks, in order to show the potential

- *Philippos Papaphilippou is with the School of Computer Science & Statistics, Trinity College Dublin, Ireland (E-mail: papaphip@tcd.ie).*
- *Thiem Van Chu is with Tokyo Institute of Technology, Japan (E-mail: thiem@artic.iir.titech.ac.jp).*

impact of path flexibility on performance.

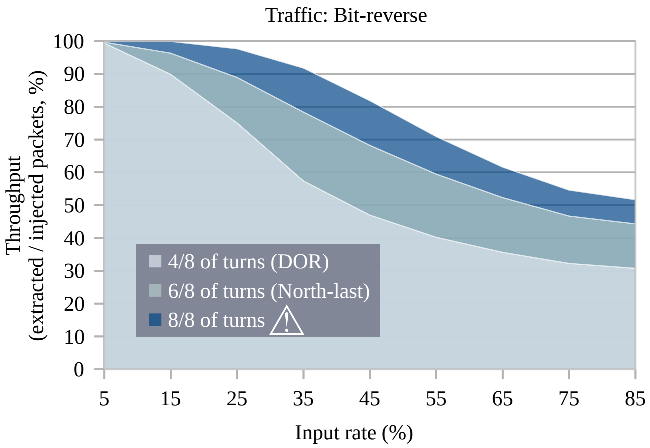

Fig. 1. NoC throughput under different quantities of turn restrictions

For instance, for a 55% input rate of bit-reverse traffic [2], DOR yields a throughput (extracted over injected packets) of 40%, while the heuristic-based "north-last" and the full-freedom ones yield 59% and 72% respectively. The goal is to approach the performance of the latter, but with deadlock freedom. Note that these numerical values are case-specific. The throughput metric is simplified for the introduction here. See section 5.1 for more detailed discussion and simulations from which this example is based on.

Contribution: The proposed solution is a hybrid approach that solves the deadlock problem more efficiently than traditional routing algorithms. It gives the illusion of full freedom to any routing algorithm, deadlock-free or not, by using it by default, while intervening with a fallback algorithm when needed. A fallback algorithm such as XY or YX dimension order routing (DOR) is activated only locally and when deemed necessary according to the freedom condition. The freedom condition requires minimal information to decide if a packet can perform an originally restricted-turn. This is achieved without assuming additional queues (such as escape channels [8], interface buffers [9]), deadlock detection [10] and recovery routines [9], misrouting (non-minimal path), packet reordering and global knowledge that are usually found in the literature on NoCs with input-queued VC routers. An intentional but indirect outcome of this paper is also the increased routing performance of example hybrid algorithms exploiting the additional flexibility of the formally proven deadlock-avoidance technique. The evaluation framework is open source.[1]

Following is background information (section 2) on the related router architectures and the baseline deadlock avoidance model. Section 3 introduces the proposed algorithm, while section 4 is a proof for the correctness of the proposal. The evaluation (section 5) includes simulations of example routing algorithms based on the proposed approach, as well as a study on the router resource utilisation and implementation efficiency. Finally, the paper concludes with related work (section 6), discussions on future work (section 7) and a conclusion (section 8).

2 Background

2.1 Network-on-chip routers

NoC routers are essentially switches, forwarding packets from port to port. The most common NoC routers are based on an input-queued switch with virtual channels (VCs). On each port ({E, S, W, N, C} for east, south, west, north and centre), the input is connected to a demultiplexer splitting the packets/flits into a fixed number of VCs (buffers). A virtual channel allocation scheme is one of the first decisions a NoC router makes on packet arrival, which can also relate to quality of service (QoS).

Each group of VCs are then demultiplexed into the assumed crossbar (functionally a superset of permutation networks) connected to the outputs (ports), resulting in a 5×5 switch. The buffering is necessary to facilitate the temporary storage of incoming packets until a passable matching is achieved between the output ports and all virtual channels. A matching is calculated on every fixed period of time and is used by the crossbar. See figure 2 (top left) for an illustration of a router with a number of VCs near each of its inputs.

2.1.1 Input-queued routers with VOQs

Instead of virtual channels (VC), virtual output queues (VOQs) can be used to implement switches, and are especially common in network switches. There is a queue for every input-output port combination. Thus, there are P groups of P VOQs, resulting in P^2 total buffers for a P-port switch (in 2D-mesh NoCs, $P = 5$).

A disadvantage of virtual output queues over virtual channels in NoCs is that they work as a static virtual allocation scheme in a switch with 4 or 5 virtual channels. This essentially means that the queue utilisation may be less efficient, and the buffer space is generally higher. It is also more challenging to reduce the number of those queues (one way is to restrict routability, see the discussion of section 5.3). Additionally, supporting quality of service (QoS) can challenge scalability, due to the increased queuing requirements.

However, as both VCs and VOQs support memory sharing per queue group, their differences can become less significant according to the implementation, also considering that 4 VCs may already be a norm in modern processors [11], [12].

2.1.2 Output-queued (OQ) routers

Another approach to NoC router implementation is output-queues. They eliminate the use of a crossbar in favour of simpler logic. In this case, the queue organisation is the same with virtual output queues, and is illustrated in figure 2 (bottom).

On FPGAs, this is a prominent NoC router architecture [13], as with the split-merge switch [3], [4]. The split-merge switch is an output-queued switch meaning that, upon arrival, the incoming packets are immediately split across queues according to the destination port. Those queues are then grouped based on their destination port, and are multiplexed only once. This results in low-complexity and/or highly-pipelinable logic, the split and merge units, which

[1] Source available: https://philippos.info/deadlock

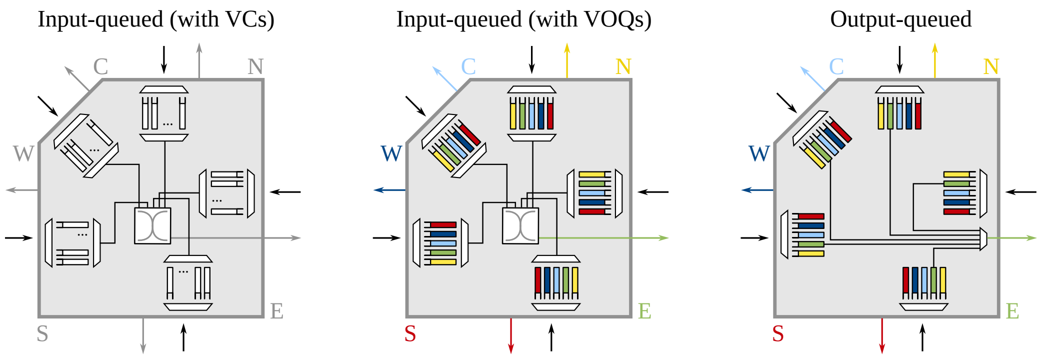

Fig. 2. Input and output-queued NoC routers. Colour-coded queues feed the output port of the corresponding colour. The non-coloured queues rely on virtual channel (VC) allocation, hence each of their packets can be destined to different output ports. The outputs to ports other than E are abstracted for simplicity.

are here equivalent to demultiplexers and multiplexers respectively.

There is no need for an expensive scheduling algorithm, such as by featuring an iterative approach to perform well. There is only arbitration near the output, for every output port, which can be fulfilled by only using priority encoders. This also results in high scheduling performance, as it naturally provides more connectivity combinations than input queued for the same queue topology. This is because of the absence of additional arbitration steps near the inputs, which would otherwise serialise dequeuing from the output queues coming from the same input port.

A potential disadvantage of output-queued switches as NoCs routers is the limited scope for memory sharing, the associated cost of which also relates to QoS support. Both memory sharing and QoS are seemingly less popular on FPGAs, as simpler designs are preferred [4], [7].

Our proposal provides the theoretical foundations to provide more flexibility in routing for such VOQ-like queue topologies, instead of relying on worse-performing traditional routing.

2.2 Turn Model

The turn model can be used to create routing algorithms based on turn restriction to avoid forming any possible cyclic dependency [14]. The resulting set of possible algorithms can be summarised by the rules illustrated in figure 3. There are clockwise and counterclockwise turns for which at least one turn from both must be forbidden. However, for each diagonal direction there must be at least one way for packets to travel, hence the "≥ 1" sum rule per column in the figure. This model is a superset of the older "dimension ordered routing" (DOR), i.e. XY DOR for first traversing along the x-axis (no "↰, ↱, ↲, ↳" turns) and YX DOR (no "⬑, ⬐, ⬏, ⬎" turns).

When reducing the restrictions as much as allowed by this model, there needs to be one clockwise and one counterclockwise turn restriction, out of which they are not on a common path, as explained above. This results in 12 possible combinations for the forbidden turns (permutations of 2 from 4 (i.e. $P_{(4,2)}$), as repetition from the table columns would lead to forbidden destinations). In other words, the turn model gives 12 turn-restriction-based algorithms. However, if a rotated mesh is considered equivalent, this reduces the number of algorithms to only 3: "west-first" (no "↰, ↲" turns), "north-last" (no "↰, ↱" turns) and "negative-first" (no "↰, ⬎" turns) routing algorithms.

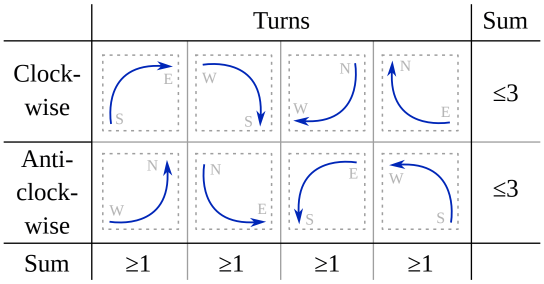

Fig. 3. Turn model for deadlock avoidance

2.3 Deadlocks

The deadlocks that happen in NoC routers with a VOQ or OQ queueing topology are a bit less trivial than the classical example paths on a simple 2×2 mesh [15]. In such a deadlocked 2 × 2 mesh with a single queue (of length 1) per link per node, the remaining packets all have 2-hop path, as 1-hop paths would have been consumed directly by the destination, and 3-hop paths would have been U-turns (this paper only studies minimal path routing). In this case, there is a cyclic dependency on the four buffers. In the OQ and VOQ case, however, it is impossible to create a deadlock on a 2 × 2 mesh, as all of the four different 2-hop paths never pass from the same queue.

Figure 4 illustrates an extended deadlock state (i.e. also including blocked queues not necessarily from the cyclic dependency) that is observed in a simulation of 3 × 4 mesh NoC with OQs. The OQs are of length 2 and the simulation uses a routing algorithm without deadlock avoidance. The numbered nodes represent the NoC nodes, while the blue arrows represent the filled output queues. The start and target positions of the blue arrows denote the source and destination of single turns (e.g. "↱" for the SE turn), as (V)OQs are associated here with turns. The dashed arrows represent queue dependencies on filled queues. They do not necessarily show which queues are involved (the heads of two queues can target the same destination queue), though here it is more apparent, given that this simulation is in a saturated state.

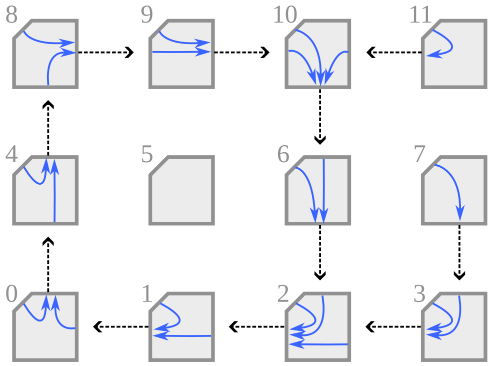

Fig. 4. Extended deadlock state example from a simulation of a 3×4 NoC with OQs.

3 Proposed algorithm

The proposed algorithm is bimodal, and avoids deadlocks by selecting between a base algorithm and a fallback algorithm based on the freedom condition F (section 3.2) for every individual next hop decision. The fallback routing algorithm in our case shall follow the turn model, and can be for example XY DOR, "west-first", etc. The algorithm 1 presents the distributed algorithm among the NoC's nodes that relates to the routing decisions of OQ or VOQ-based NoC routers.

```
in $inp = from(p)$;      ▷ the input port receiving $p$
out $sel$;               ▷ the output port selection
while forever do
    receive ($p$);
    if $F(p)$==True then
        $sel \leftarrow base\_algorithm(p)$;
    else
        $sel \leftarrow fallback\_algorithm(p)$;   ▷ Turn model
    end
    OQs[$inp$][$sel$].$enqueue(p)$;
end
```

Algorithm 1: Proposed routing algorithm for (V)OQs

As the goal of this algorithm is to provide path flexibility, such as for better routing performance, the base algorithm is expected to allow a superset of the turns allowed by the fallback algorithm. The base algorithm can be any arbitrary routing algorithm, deadlock-free or not, which is also very useful for adopting algorithms that would otherwise require a specialised queue organisation for achieving deadlock freedom. Some examples include O1-Turn and LEF [16] that assert virtual channel allocation requirements. Like with turn model algorithms, the flexibility of the base algorithm is also appropriate for adaptive routing, where the routing decisions are based on heuristics such as for estimating network congestion.

Sections 3.2 and 3.3 provide a detailed description of the proposed freedom condition.

3.1 Assumptions

The proposed approach is studied under the assumptions listed in table 1. The table classifies these assumptions based on whether they are a requirement of the proposed algorithm ("fundamental"), or if they are design choices serving simplicity or practicality ("adaptable"). The discussion of section 7 elaborates on some trivial cases for generalising some of the "adaptable" attributes.

TABLE 1
Assumptions

Attribute	Value	Comments
Fundamental:		
NoC topology	2D mesh	
Queue organisation	OQ or VOQ	up to 5×5 virtual/physical queues from (N, W, S, E and C)[1]
Variable-size packets	supported	head flit carries the packet size (elaborated in sections 3.2 and 7)
Adaptable:		
Multi-flit packets	supported	
Allowable paths	minimal (shortest)	Manhattan distance (i.e. $\lvert x_1 - x_2 \rvert + \lvert y_1 - y_2 \rvert$)
Misrouting	no	no temporary redirections
Interface queues	no	
Flow control	ready signal per port	5-bit based on queue occupancy per (N, W, S, E and C) turn[1]
Re-allocation scheme	WPF [12]	non-empty channels can receive new packets, if a tail flit arrives
Processing latency	1	router implementation-specific

[1] fewer in practice, as minimal path contains no U-turns etc.

The series of events when a node receives a packet is the following: routing (proposed algorithm), virtual channel allocation (fixed because of the (V)OQ organisation), scheduling (including for the whole crossbar when using VOQs [7]) and finally switch traversal. The routing decisions of an upstream node are assumed not to influence the routing decisions of a downstream node.

The proposed approach supports multi-flit packets. The paper focuses mostly on single or few-flit packets, a common case for NoCs in processors [17]. This is also the case with NoCs working as system interconnects inside FPGAs. Hence the use of whole packet forwarding (WPF) that avoids unnecessary blocking of the buffers (see table 1). A simple scheduling assumption for the output arbiters is required to give priority to flits of the same packets (e.g. before applying round-robin) to avoid packet overlapping.

In the nomenclature, the term "turn" is also sometimes used for forwardings from and to a port that is on the opposite side (going straight), as well as to the node's centre ("C"). These are considered legal in accordance with the turn model. The centre in this case is a "sink", being able to consume packets directly [18], [19]. A node's centre as a consumer cannot contribute to a deadlock, as every inserted packet is eventually consumed in a dedicated (virtual) output queue. Also, the turn model does not consider straight forwardings for deadlocks, though different models can also break circles in straight sections of a packet's path.

3.2 Freedom condition

The freedom condition (F) is a sufficient condition for the avoidance of deadlocks, and is applied on packet arrival into any non-sink node (sink is the final destination).

The main idea of F goes as follows. If an incoming packet p can do a clockwise or a counterclockwise turn on the next hop, and those turns are considered restricted based on the fallback algorithm (e.g. "north-last"), we check the worst case occupancy for that queue of the next hop router. This is achieved by summing up the contents of all queues that feed into that queue (only the packets that can go into that queue), plus its own contents, and checking whether the addition of the incoming packet would cause an overflow in the worst case. The worst case is for all packets in the sum to end up in the restricted-turn queue and that for any reason it stops being consumed in the meantime. The time frame for the worst case occupancy is for until packet p manages to be enqueued into the restricted turn/queue. Whenever $F(p) = False$, p takes an alternative queue/output port/direction, which ensures that p will not be able to make a turn not following the turn model of the fallback algorithm in the routing step of the next hop.

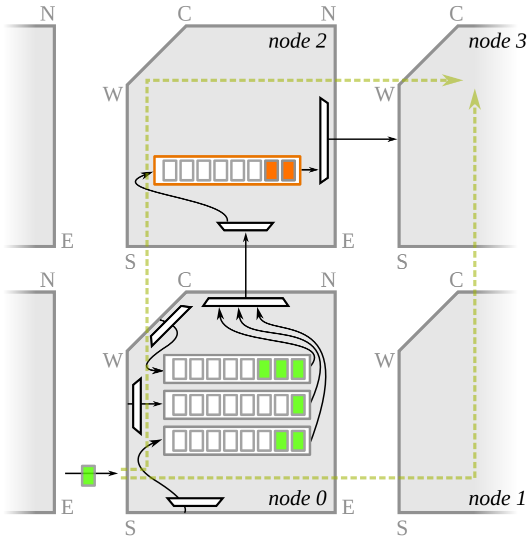

Fig. 5. Example NoC state with single-flit packets and OQ (of depth 8)

Figure 5 illustrates the main idea with an example NoC state that can benefit (avoid a deadlock) from the use of the freedom condition. An incoming packet to node 0 has node 3 as a destination. This packet can be routed to two possible queues, one for going through node 2 (middle queue in node 0) and one for going through node 1 (not shown). The depicted queues in node 0 with green packets are all the queues that feed node 2 from its south port. Given that the fallback routing algorithm is based on the turn model, and the originally-forbidden turn is on the path through 2 ("↷" queue/turn in node 2, with red hint), we notice that the packet may not be able to be accommodated, if the forbidden path is followed. Assuming all packets (green) in those queues target the red queue, i.e. based on packet destination, the worst case occupancy for the red queue is 8 (2+1+3 from node 0 and 2 from node 2). This will happen, for example, under the current queue states, if the red queue stops being consumed in the meantime. Thus, the incoming packet could cause a stall, if it follows the path through 2, and it would be safer to follow the path through node 1. The receiving queue in node 1 ("⤴" turn) is not forbidden by the turn model, as it is the counterclockwise direction-equivalent of the forbidden one (≥ 1 requirement in section 2.2), and therefore will not cause a cyclic dependency on its own.

Let p be a packet or flit received from any port $from(p) \in \{E, S, W, N, C\}$. Let $sel \in \{E, S, W, N, C\}$ be the selected direction of the proposed routing algorithm. based on the result of the freedom condition.

The freedom condition is only useful whenever the next hop implied in sel could make p land in a queue of the next hop that would be a restricted turn based on the turn model. $next(p)$ is the set containing all possible queue destinations of p inside the next-hop's router, and in practice here it also follows no-misrouting. When p arrives at the next hop, according to its designated final destination it can go straight ("$\nearrow$"), to the centre, clockwise ("$\circlearrowright$") or counterclockwise ("$\circlearrowleft$"). Based on our assumptions the latter two are the only ones that can be forbidden turns, and are mutually exclusive.

The piecewise function of the freedom condition is as follows: $F(p) \equiv$

$$\begin{cases} size(p) + occp(q'_{\circlearrowright}) + \sum_{d \in \{C, \nearrow, \circlearrowleft\}} occp(q_d) \leq cap(q'_{\circlearrowright}), & \\ \qquad q'_{\circlearrowright} \in next(p) \land \neg turn(q'_{\circlearrowright}) & \\ size(p) + occp(q'_{\circlearrowleft}) + \sum_{d \in \{C, \nearrow, \circlearrowright\}} occ'(q_d) \leq cap(q'_{\circlearrowleft}), & \\ \qquad q'_{\circlearrowleft} \in next(p) \land \neg turn(q'_{\circlearrowleft}) & \\ True, & Otherwise \end{cases}$$

where $cap(q)$ is the capacity of queue q, q_d is a queue of the current node (receiving p) that receives packets from direction d, while q'_d is a queue of the next hop as pointed by sel. $turn(q)$ indicates that the queue q corresponds to a legal turn based on an algorithm derived by the original turn model.

The occupancy of queue q is calculated as $occp(q) = \sum_{p' \in q} size(p')$, which counts multi-flit packets rather than the physical occupancy. The $size(p)$ function indicates the number of flits a packet p consists of. This is useful in the cases where multi-flit packets are allowed, where this function is only useful on the head flits of packets. The availability of this information is assumed. For multi-flit packets the head flit shall mention the number of flits for the whole packet. For non-head flits of a packet p, $size(p) = 0$, since the final packet occupancy for multi-flit packets should be known by the time their first flit is accommodated.

3.3 Freedom condition adaptation

In order to simplify the presentation of the freedom condition also based on implementation practicality, this subsection provides an adaptation example. The section provides a simplified sufficient condition F' based on system implementation assumptions and a higher degree of approximation for the queue occupancy, but with the same worst case (i.e. $F'(p) \rightarrow F(p)$).

The size of the packets ($size(x)$ function) is replaced by 1, as only single-flits are considered for brevity. A related modification is to replace the packet/flit counts having a potential queue target inside the next hop with the entire (physical) occupancy of each of the corresponding queues (denoted by $occ(x)$ for each queue x). Although there can be an overhead in the path flexibility the algorithm provides, it could also be implemented more efficiently as the queue length circuitry is likely to be already existent.

Another adaptation is that the "north last" routing algorithm is selected as the fallback condition (or a restriction subset that still follows a turn model). By having the same output port ("N") as the potential direction for the start of both forbidden turns for the next hop (SW, SE, i.e. "$\curvearrowleft$, $\curvearrowright$"), there needs to be less logic for the queue capacity checks in total. This is because $q_C, q_{\nearrow}, q_{\circlearrowright}$ and $q_{\circlearrowleft}$ represent the same queues for the turns SW and SE, happening at $q'_{\circlearrowleft}$ and $q'_{\circlearrowright}$ respectively in the next hop across the N direction. Under this assumption, the counterclockwise, straight and clockwise of the current node in F become the WN, SN and EN queues ("⤴, ↑, ⤣") and for the next hop the SW, SN, SE ("$\curvearrowleft$, ↑, $\curvearrowright$") respectively.

The link overhead in this case is two southerly wires between each consecutive neighbouring nodes in the y-axis, of bit widths equal to $\lceil \log_2(cap(q'_{\curvearrowright})) \rceil$ and $\lceil \log_2(cap(q'_{\curvearrowleft})) \rceil$ correspondingly. This metric makes the simplifying assumption that the occupancy of the next hop can be provided within the same cycle. A practical implementation with a credit system, such as with pipelined router implementations, is also likely to use fewer wires between the nodes.

$$F'(p) \equiv$$

$$\begin{cases} 1 + occ(q'_{\curvearrowright}) + \sum\limits_{d \in \{C, \uparrow, \text{⤴}\}} occ(q_d) \leq cap(q'_{\curvearrowright}), & q'_{\curvearrowright} \in next(p) \\ 1 + occ(q'_{\curvearrowleft}) + \sum\limits_{d \in \{C, \uparrow, \text{⤣}\}} occ(q_d) \leq cap(q'_{\curvearrowleft}), & q'_{\curvearrowleft} \in next(p) \\ True, & Otherwise \end{cases}$$

Figure 6 illustrates the potential paths that involve non-allowable turns/queues by "north-last" as the fallback routing algorithm, which restricts SW and SE turns. When a packet can make such turns in the next hop (associated with the queues $q'_{\curvearrowleft}$ and $q'_{\curvearrowright}$), then exactly one of the first two pieces of the piecewise function F' is used.

For the sake of notation simplicity both F and its variation F' are only valid when applied "serially" on the set of incoming packets per cycle (but still being a combinational circuit operating in the same cycle). In practice, their computation can be implemented in parallel for each input port, but there needs to be a simple arbitration step for synchronisation, such as with a priority encoder among the input ports. This is for whenever two packets can compete for the same potential queue of the next hop (packets from different ports/directions are still placed in different queues).

3.4 Novel routing algorithms

Two example routing algorithms are provided for the purposes of evaluation, and are based on the proposed bimodal algorithm. The first one is "XY/Adaptive" and uses XY DOR as a fallback algorithm. This fallback algorithm is equivalent to the restrictions as found in "north-last" in our arrangement (adaptation of section 3.3), as $F' = True$ for south ports, since they cannot form a forbidden turn as per our base turn model. The "Adaptive" component

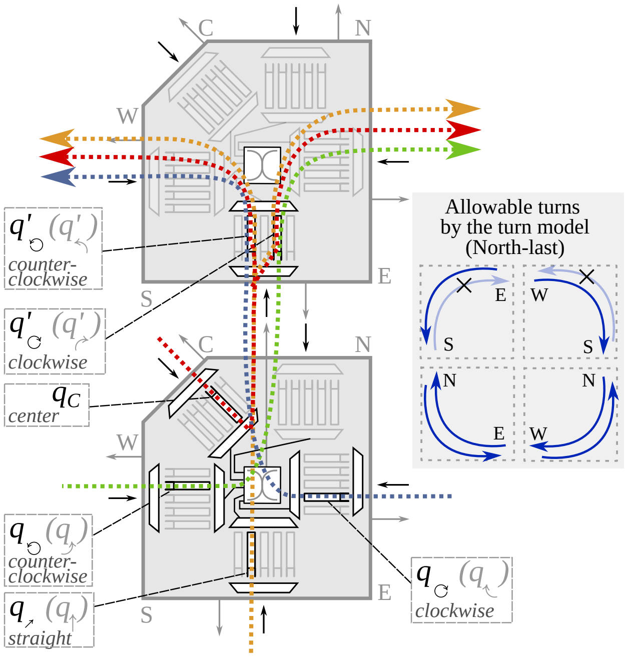
C
N
W
E
S
C
N
W
E
S
$q'_{\circlearrowleft}$ $(q'_{\curvearrowleft})$
counter-clockwise
$q'_{\circlearrowright}$ $(q'_{\curvearrowright})$
clockwise
q_C
center
$q_{\circlearrowleft}$ $(q_{\text{⤴}})$
counter-clockwise
$q_{\nearrow}$ $(q_{\uparrow})$
straight
$q_{\circlearrowright}$ $(q_{\text{⤣}})$
clockwise
Allowable turns
by the turn model
(North-last)
E
W
S
S
N
N
E
W

Fig. 6. Packet paths involving originally-forbidden turns on the next hop

of "XY/Adaptive" is full-freedom, i.e. there is no turn restriction and the routing decisions follow a heuristic.

As adaptive routing algorithms like "north-last" introduce a degree of turn freedom, heuristics are used to avoid congestion. Throughout the entirety of this study, the adaptiveness heuristic is consistent for all experiments. It is enabled where applicable, including for the novel "XY/Adaptive". The higher priority is given to the turn for which the direction maps to the queue with the least occupancy. This is conventional, but in this way, the information used by the heuristic is also local to the node. The occupancy of each local queue is still indirectly associated with the congestion in the corresponding next hops.

The second proposed algorithm based on the freedom condition F' is "XY/O1-Turn", which alternates between XY DOR and O1-Turn [20] per packet arrival. O1-Turn randomly selects XY or YX DOR for the whole path of a packet, and does not provide adaptiveness. O1-Turn is originally designed for routers with virtual channels and achieves deadlock freedom by using separate queues/channels for XY and YX packets [16]. Therefore, this is an example where the proposed technique is used to adapt an algorithm to a VOQ-like queue organisation, without the need to double the storage requirements. In this case, the aforementioned heuristic is never consulted, as XY DOR acts as a fallback algorithm when O1-Turn's decision involves the north port and F' considers it unsafe.

4 Proof

In order to prove that the algorithm in F is always deadlock-free, a proof by contradiction is provided.

Let c be a cyclic dependency [21] between n buffers that has been allowed by the routing algorithm at time t.

$$c = \{q_0, q_1, \ldots, q_{n-1}, q_0 = q_n\} \tag{1}$$

As this is a deadlock, all queues in the cycle are full, with each head packet only able to be served by queues of the subsequent node (including the subsequent queue in the cycle). That is

$$occ(q_i) = cap(q_i) \forall i \in \{0, 1, ..., n-1\}, \tag{2}$$

where $occ(q_i)$ and $cap(q_i)$ is the occupancy and capacity of queue q_i respectively. Additionally,

$$q_{j+1} \in next(head(q_j)) \tag{3}$$

for every j rotation of i, where $next(head(q_j))$ is the set containing all possible buffer destinations of the head of queue q_j. Note that as long as a packet is in a queue, the direction for the next hop (corresponding to a specific node) is already determined, but the queue placement in the next-hop node will still be decided upon arrival at time $t+1$.

For every (q_j, q_{j+1}) pair, a head packet proceeds from q_j to q_{j+1} if and only if the algorithm considers the packet making a legal forwarding to $node(q_{j+1})$ (for deadlock avoidance), and the $node(q_{j+1})$ notifies that it will be able to accommodate it on cycle $t+1$.

As each queue represents a turn (a permutation of 2-selection from {E, S, W, N, C}), based on the turn model, it is impossible for the cyclic dependency to be based only out of turn/queues being inline with the turn model. That is $\neg(\forall i, q_i \in c \wedge turn(q_i))$, where $turn(q_i)$ denotes that q_i follows the turn model. In other words, there is at least one queue/turn not following the turn model. That is, for c to be able to be formed,

$$\exists i,\ q_i \in c \wedge \neg turn(q_i). \tag{4}$$

Similarly, based on the turn model, as summarised in figure 3, the forbidden turns alone are also not able to form the circle c either, as they will consist of a strict subset of the 4 turns in the clockwise direction and a strict subset of the 4 turns in the counterclockwise direction. Therefore,

$$\neg(\forall i, q_i \in c \wedge \neg turn(q_i)) \tag{5}$$

$$\leftrightarrow \exists i,\ q_i \in c \wedge turn(q_i). \tag{6}$$

From (4) and (6), there is at least one consecutive queue pair that consist of one queue following the turn model and one that does not, i.e. $\exists l = (i+k) \bmod n, k \in \mathbb{Z}$,

$$turn(q_l) \wedge \neg turn(q_{l+1}) \tag{7}$$

As this pair also denotes a dependency ($q_{l+1} \in next(head(q_l))$), at the time of insertion of the now $head(q_l)$ to q_l, the F condition has been met, i.e.

$$size(p) + occp(q_{l+1}) + \sum_{d \in \{C,l,l'\}} occp(q_d) \leq cap(q_{l+1}), \tag{8}$$

where $q_{l'}$ is the other queue in the $node(q_l)$ than q_l that can feed q_{l+1}. One of them (q_l and $q_{l'}$) is for the straight movement ($q_{\nearrow}$, or $q_{\uparrow}$ for F'). The other, based on F, if q_{l+1} is clockwise ($q'_{\circlearrowright}$, or $q'_{\curvearrowright}$ for F'), it is counterclockwise ($q_{\circlearrowleft}$, or $q_{\Rsh}$ for F'), and vice-versa (if $q_{l+1} = q'_{\circlearrowleft}$ (or $q'_{\Lsh}$ for F'), then it is $q_{\circlearrowright}$ (or $q_{\curvearrowleft}$ for F')). There is also q_C that starts from the centre of $node(q_l)$. These queues (q_l, $q_{l'}$ and q_C) are the only queues that can feed q_{l+1} based on the assumptions, such as with no misrouting.

First, in section 4.1, the proof is done under the assumption that there is only a single flit per packet. Then, section 4.2 elaborates on the multiple-flit case.

4.1 Single-flit packets

As F is always followed, at the time $head(q_l)$ was enqueued into q_l, the value of F was *True*, i.e.

$$1 + occ(q_{l+1}) + \sum_{d \in \{C,l,l'\}} occ(q_d) \leq cap(q_{l+1}). \tag{9}$$

Following the worst case, it is assumed that the queue q_{l+1} stops being consumed in the meantime for any reason, such as congestion. Any packet p that arrived to $node(q_l)$ after (the now) $head(q_l)$ would have been one of the following cases based on the turn options allowed by the minimal path:

A) p can land in q_{l+1}: this is the case when p avoids the "forbidden" turn (q_{l+1}), since the freedom condition causes it to be routed it to its alternative direction. As $F(p)$ = *False* based on the current occupancies, p will follow the fallback algorithm which will place it in a queue other than q_l, q_C and q'_l, as it will target a different node to $node(q_{l+1})$. Since the turn model always provides full-routability without relying on non-allowable turns, p will simply follow the fallback routing algorithm, which will ensure a next hop without a queue violating the turn model.

B) p cannot land in q_{l+1}, but still passes through q_C, q_l or $q_{l'}$: in this case, the fallback condition will always hold (i.e. $F(p)$ = *True*). Since it passes through one of the queues that can feed q_{l+1} (q_C, q_l or $q_{l'}$), the only effect with respect to the deadlock condition is an earlier falsification of the freedom condition in future packets that actually land in q_{l+1} (as in case A). This happens when:

– p's destination is the next hop ($node(q_{l+1})$). The landing queue will be q'_C, which does not cause a deadlock (being a sink).

– p can only go straight for the remainder of its path. The current x or y-axis (according to the topology of q_l and q_{l+1}) coordinate is the same as the one of the packet destination. This would lead to $q'_{\nearrow}$ ($q'_{\uparrow}$ for F'), which already follows the traditional turn model.

– p can go to the other forbidden turn $q'_{l+1} \in \{q'_{\circlearrowright}, q'_{\circlearrowleft}\}$ or $\{q'_{\curvearrowright}, q'_{\Lsh},\}$ for F'. In this instance, it cannot also have q_{l+1} as a potential destination (being mutually exclusive to q'_{l+1}), as U-turns are not allowed in minimal routing. Thus, this remains orthogonal to the studied deadlock condition, though such packets can still go straight, with equivalent outcomes as the instance above. Note that this is subject to the followed turn model, as with F' that places the forbidden turns on the same downstream router (at North for F').

C) p cannot be placed in q_C, q_l or $q_{l'}$: this case p includes all other instances when it cannot go towards $node(q_l+1)$ (at North for F'). As these are the only queues that feed the queue of the forbidden turn q_l+1, they are unrestricted by following the turn model.

In all the aforementioned cases, $head(q_l)$ would have been the last packet of all queues q_l, $q_{l'}$ and q_C that could land in q_{l+1} (for which $\neg turn(q_{l+1})$) on the next hop. As $F(head(q_l))$ = *True* (the now-head) at the time of arrival, it would have been able to be forwarded before q_{l+1} became full. This packet would have been able to become the tail of

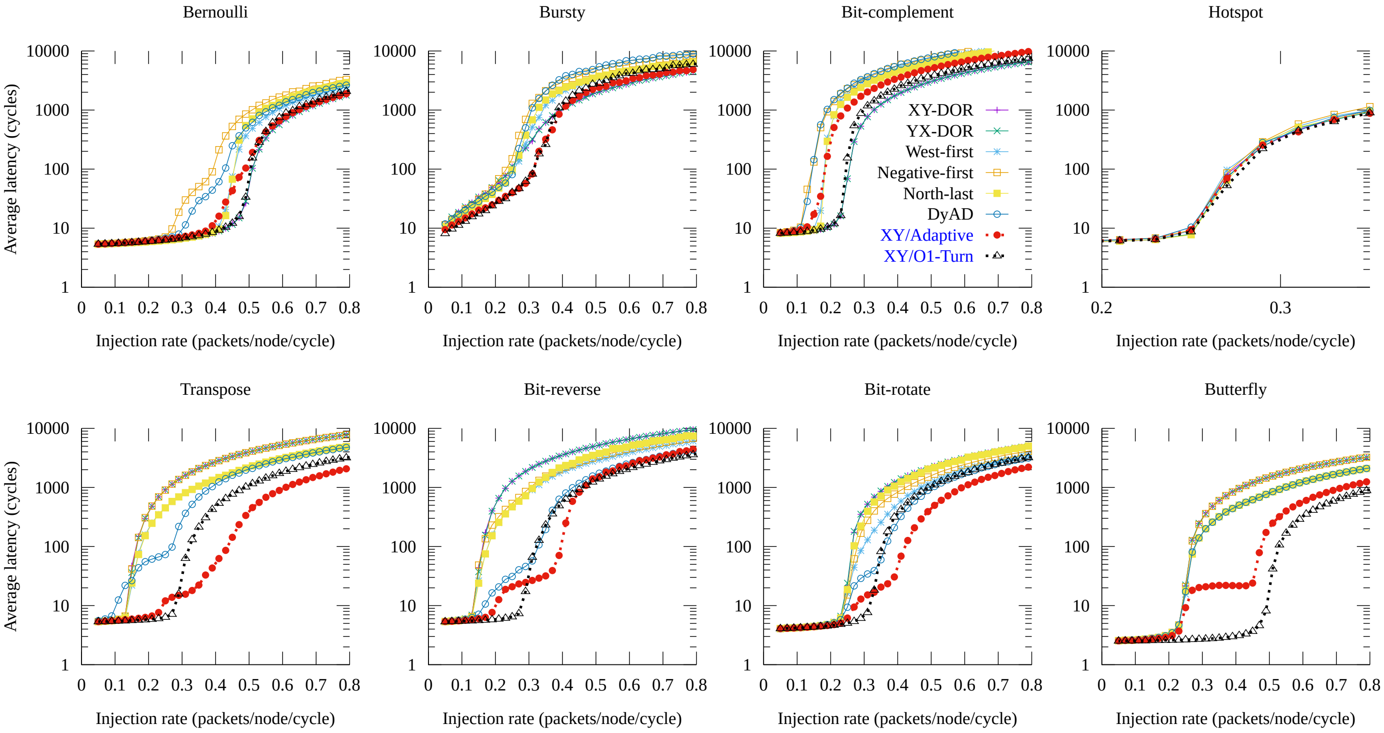

Fig. 7. Comparison of average packet latency using different routing algorithms under different traffic patterns.

q_{l+1} at least. Thus, when q_{l+1} is full, the head of q_l cannot have a dependency to q_{l+1}, i.e.

$$q_{l+1} \notin next(head(q_l)). \tag{10}$$

From (3) ($q_{l+1} \in next(head(q_l))$) and (10), this leads to a contradiction.

4.2 Multiple-flit packets

When taking into account the existence of multi-flit packets, the cases presented at section 4.1 can be adapted accordingly. Based on the assumptions, each head-flit contains the packet's length information, which can be used to calculate the worst case occupancy for F for the entirety of the packets. In other words, the decisions happening at the time a head-flit arrives are enough to ensure the packets fit inside their assigned queues (concerning q_{l+1}). As this assignment lasts for the entirety of a packet, when p is not a head flit, it does not consult F again. It also does not impact the other instances of F (e.g. from other ports), as its size is considered 0 by the $occp()$ occupancy function (see section 3.2).

As an example, packet p' is another packet (head-flit) that arrives midway through the arrival of a packet/flit p to a queue from the set $\{q_C, q_{l'}\}$ (excluding q_l, as output arbiters are required to extract full packets). In this case, all flits from p are guaranteed to fit inside q_{l+1}, as the freedom condition counts for the whole size of packets. At the same time, p' is only granted with *F=True* if there is space for it, including all flits of both the packets of p' and p.

5 Evaluation

The proposed theory is explored by evaluating the proposed novel algorithm examples of section 3.4. First, section 5.1 uses high-level simulation to comment on the algorithmic performance of the freedom condition adaptation (F'). Then, sections 5.3 and 5.4 provide implementation-related results based on an example NoC as described with Verilog, a hardware description language (HDL). The goal is to isolate and compare the routing algorithm behaviour as found in the state-of-the-art (see related work in section 6).

5.1 Routing performance – synthetic

The performance of the proposed routing algorithm methodology is studied under a variety of synthetic traffic models in simulation. The presented results are for an 8×8 2D mesh NoC, the routers of which use output queues. The simulation framework builds upon our earlier open source framework for a study on FPGA switches [7].

The traffic models used in this evaluation are selected with diversity in mind, but are also synthetic which is considered the norm for this level of system design at NoC routing. First, uniform Bernoulli arrivals and uniform bursty traffic are the most common models for interconnection circuits and relate to system interconnect use cases [7], [16]. Then, bit complement, bit reverse, bit rotate [2] and butterfly produce destination permutations based on the corresponding bit manipulation operations on the destination address. These and the transpose model are based on applications, such as the latter for FFT accelerators [2]. Finally, hotspot is the Bernoulli model modified for the central node to receive requests with four times higher probability than the rest of the nodes, modelling system-on-chip behaviour [2], [16].

Figure 7 presents the performance results from this experiment with respect to the average packet latency. Each output queue has a depth of 16 flits-packets. There are only single-flit packets and the forwarding from a node can

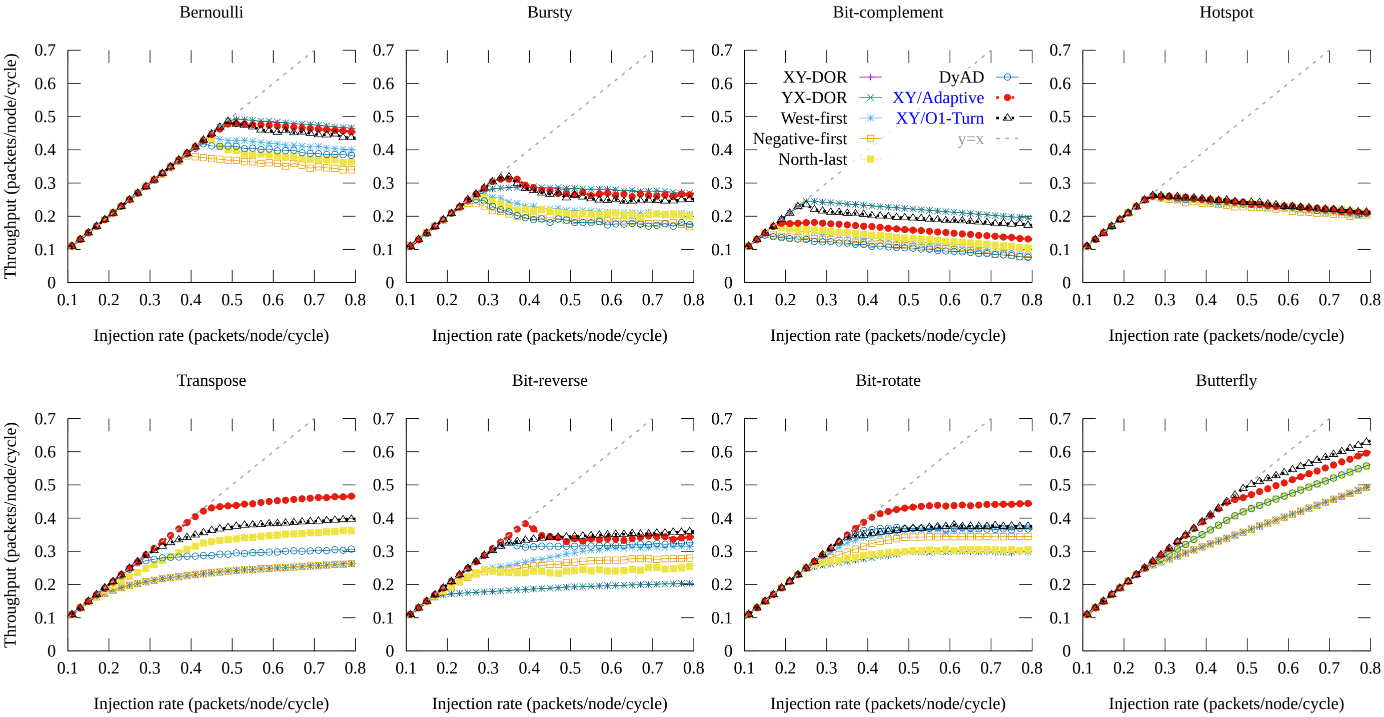

Fig. 8. Comparison of throughput among different routing algorithms under different traffic patterns

happen under a cycle, also according to the queue states. Each simulation has a region of interest of 5,000 cycles. warmup period. When the average latency is predicted to become above 1,500 cycles, the simulation stops early and the series stops to save simulation time. Each data point is an average of 5 runs. The observations for the equivalent experiment with virtual output queues are similar but not shown for brevity.

A general observation is that the novel "XY/O1-Turn" and "XY/Adaptive" are the winners in the majority of the traffic patterns, achieving the lowest average packet latency for almost any presented injection rate. Two noticeable exceptions are the uniform Bernoulli and bit-complement cases. In the first, both fully-adaptive algorithms are marginally worse than with dimension order routing (DOR). In the second case, "XY/Adaptive" comes third, but it is not a close third, so it could be said that 'XY/O1-Turn" is a more balanced solution.

The second-class performance of the full-adaptiveness examples under certain traffic cases is expected, but it is not a limitation of the deadlock avoidance model. There are traffic patterns where additional turn-freedom is not always beneficial, at least when the adaptiveness heuristic has a more-local scope [22]. Hence, instead of a fine-tuned routing algorithm, the main focus is the theoretical model that allows a superset of potential packet paths than is currently feasible. For instance, as XY and YX DOR use a subset of the allowable turns, future adaptiveness heuristics could still use the proposed model while also reverting to XY or YX DOR where deemed beneficial to performance.

Another observation is that overall "XY/O1-Turn" and "XY/Adaptive" are more well-rounded than the alternative algorithms in this selection, always being among the top performers. For example, DyAD routing [23] sometimes performs the worst, as under high bursty or bit-complement traffic, whereas for bit reverse and transpose traffic is the next best alternative to the proposed ones.

Figure 8 presents the throughput results from the same experiment for numerical comparison examples. The throughput is defined as the average number of extracted packets per node per cycle. On average for all the traffic models, under a 35% injection rate, "XY/Adaptive" provides 1.23, 1.22, 1.17, 1.28, 1.19 and 1.19x the throughput of XY, YX DOR, west-first, negative-first, north-last and DyAD respectively. At 35% the corresponding numbers for "XY/O1-Turn" remain very similar at 1.23, 1.22, 1.17, 1.29, 1.19 and 1.17 times.

5.2 Routing performance – traces

Alongside synthetic workloads, we also conduct a similar experiment with the studied routing algorithms using real-world benchmarks. Specifically, we employ the PARSEC [24] traces provided by Hestness et al. [25], which are derived from full-system simulations on the M5 simulator [26]. In our experiments, the packet generation is based on the injection timestamps and traffic patterns (that is, patterns of source-destination pairs) contained in these traces.

Figure 9 presents the results of this comparison. Under this traffic, the general observation is that the proposed routing algorithms remain competitive throughout all traces, with "XY/O1-Turn" taking the lead with up to 8% lower average latency over the best non-proposed alternative.

5.3 Resource utilisation

In order to study the hardware utilisation of the proposed deadlock avoidance methodology, an 8×8 NoC is implemented in System Verilog and synthesised using yosys [27].

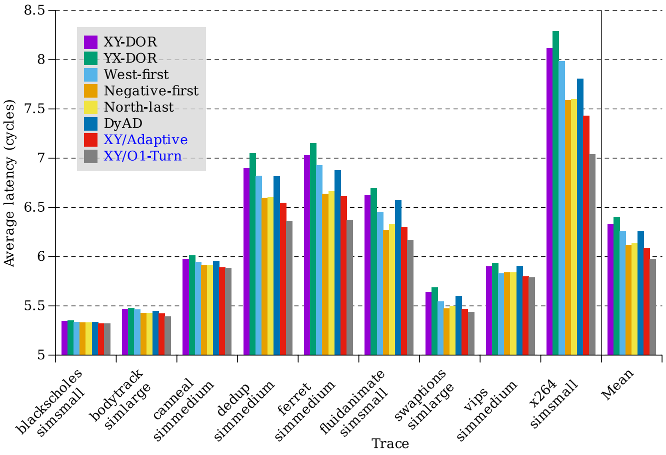

Fig. 9. Packet latency using different algorithms under traffic from Netrace

Each NoC router has a 5 × 5 output-queued switch, with 25 queues in total. The router queues are based on a rather popular formally-verified synchronous queue [28]. The size of each queue is indicatively set to 8, and each packet is 64 bits wide. As building an optimised and specialised implementation is outside the scope of this paper, the router designs do not feature pipelining and the NoC nodes do not perform a useful task (random packet generation). The results are presented for both 6-input lookup tables (LUT6) that are found in most modern FPGAs [29], and a standard cell library ("cmos_cells.lib") for ASICs.

Table 2 presents the synthesis results for 4 variations of the NoC router based on their routing algorithm. From left to right, each approach is expected to use more resources. North-last uses queue information in its heuristic, while DOR does not. "Full-freedom" (deadlock-prone) uses a superset of this information, as its heuristic is responsible for the turns from the north as well. This is the equivalent of "8/8 of turns" from figure 1. Finally, the proposed example routing algorithm "XY/Adaptive" includes the signals required for the freedom condition, hence also the two extra wires in the y-axis carrying occupancy information (for queues $q'_{\circlearrowleft}$ and $q'_{\circlearrowright}$). This is reflected in the addition of 4 public wires, two as inputs in the north direction and two as outputs of the downstream router for monitoring the two restricted turns.

It is also worth mentioning that for the turn-restriction based (XY DOR and north-last) yosys was not able to optimise unused queues away, as 25 queues were used in all cases. As the proposal is based on the observation that (V)OQs imply turn information, we know that the first two cases could result in (4 for XY DOR, and 2 for north-last) fewer queues. The no-misrouting assumption is also not exploited without manual intervention, as it could yield 5 fewer queues per router (for when sending to oneself). Additionally, as synthesis is also based on heuristics, small variations are expected, such as in the distribution of logic cells in the standard library.

As can be observed, "XY/Adaptive" being a superset of the "full-freedom" in terms of logic complexity, it has a small but measurable overhead on the wire and cell utilisation. For the FPGA case, the proposed approach only uses 21 more LUTs (<2% increase). By using the standard cell library, the most noticeable change is in the NAND gate count that increases by 7%.

TABLE 2
Routing algorithm impact on router resource utilisation

	XY DOR	North-last	Full-freedom (deadlock-prone)	XY/Adaptive
		Look-up table-based (FPGA)		
Wires	885	957	977	1002
Public wires	182	182	182	186
Cells	1132	1204	1224	1245
6-LUT	1092	1164	1184	1205
		Standard cell library-based		
Wires	10707	10985	11230	11458
Public wires	183	183	183	187
Cells	5630	5797	5962	6062
NAND	3476	3803	3566	3823
NOR	1455	1344	1649	1518
NOT	659	610	707	681
		Common		
DFF	15	15	15	15
Sync. FIFOs	25	25	25	25

5.4 Performance overhead

This part of the evaluation demonstrates that the performance overhead of including the proposed deadlock avoidance mechanism is minimal. Although the paper mostly provides a theoretical background rather than implementation insights, an indicative NoC implementation verifies our expectations on the impact on the operating frequency.

The last two router implementations from the resource utilisation study (section 5.3) are used as a building block to build a 4x4 NoC on an FPGA. The first of these routers is the deadlock-prone version that enables "full-freedom" by having no turn restriction and follows the adaptiveness heuristic to achieve high scheduling performance (see section 3.4). The second one is the "XY/Adaptive", which is essentially the same, but with the proposed deadlock avoidance mechanism enabled. The idea is to focus on the impact of adding this additional logic and wires in the NoC's routers, while having all other aspects common, including the adaptiveness heuristic.

The target device is Xilinx UltraScale+ ZU3EG using Avnet's Ultra96 board. The toolchain is Vivado 2020.2 and is used to place-and-route the logic onto the device. The implementation directives are set to "ExploreWithRemap" for design optimisation, and "Explore" for post-place and post-route physical design optimisation. The idea is to have relatively aggressive optimisation to minimise the effects of place-and-route heuristic-related discrepancies when reporting the maximal operating frequency (f_{max}).

The resulting worst negative slack is -1.69 ns for "full-freedom", and -1.706 ns for "XY/Adaptive", for a target clock of 375 MHz. This results in a sub-1 MHz drop when extending "full-freedom"'s logic to become "XY/Adaptive", as the f_{max} of the two is about 230 MHz and 229 MHz respectively.

As mentioned, there can still be small variations based on the nature of place-and-route tools. Still, the numerical results show that the proposed method can be supported efficiently in implemented designs. Note that the NoC design space is very narrow, but it also represents a worse case with respect to how the approach can be used. This is since no NoC function is pipelined and all router logic tries to fit inside a single FPGA cycle. See future work (section 7) on a further discussion on implementation.

6 Related Work

Deadlock avoidance, or detection and recovery in NoCs is still an active and thorough research topic. A great majority of the related literature is about NoCs based on virtual channels (VCs) instead of (virtual) output queues.

The proposed algorithm is applicable to NoC routers that feature a VOQ-like (VOQ or OQ) queuing topology. On FPGAs, this includes the commonly used split-merge router [4] and variations or adaptations based on the same basic switch primitives [3]. For instance, the Hoplite NoC [30] uses the split-merge router, but with less buffering and variations for different NoC topologies [31]. All aforementioned examples are classified into the output-queued category, and utilise routing algorithms following the turn model (DOR and west-first). Such switches with static virtual channel allocation by the use of OQ or VOQ queues are focused on implementation efficiency, given certain constraints. Routers based on OQ seem to be the best option for FPGA implementation because there is no need for an FPGA implementation of a crossbar. A crossbar's scheduling algorithm is better represented by hardened circuits, as it is more challenging to include clock domain crossing and iterative algorithms on a unified FPGA logic [7]. Hence the natural decision for FPGA-optimised NoCs to use OQ routers. Our research aims to close the knowledge gap on modern deadlock avoidance techniques on these types of NoCs.

In general, deadlock freedom can be achieved by focusing on the routing algorithm or the flow control protocol. A routing algorithm ensures deadlock freedom, if the paths produced by it do not form any cycles. On the other hand, a flow control protocol ensures deadlock freedom if it allocates buffers to packets in a way such that a dependency cycle of packets cannot be formed. Note that this is achieved even in the case the employed routing algorithm is not deadlock-free, that is, it may produce paths that form a cycle.

Glass and Ni conducted a foundational study of deadlock-free routing algorithms for 2D meshes [14]. They observed that there are eight possible turns in a 2D mesh and found that cycles of paths can be prevented by prohibiting only two of these eight turns. The details have been described in Section 2.2. Chiu [32] pointed out that the flexibility provided by the routing algorithms proposed by Glass and Ni is not even for all source-destination pairs. Specifically, at least a half of the source-destination pairs have only one minimal path while the remaining pairs have more. Chiu proposed to use different sets of prohibited turns for nodes in odd and even columns, called odd-even routing. In this way, the routing flexibility can be improved while deadlock freedom is still guaranteed.

Our proposed algorithm as illustrated in algorithm 1 is inspired by the DyAD routing algorithm [23]. DyAD is a bimodal algorithm as well, as it selects between two component algorithms. One of those components is set as odd-even routing. This is provided indicatively, since as with our proposal, it originally models a routing model rather than a fine-tuned algorithm. Nevertheless, the algorithm selection in DyAD only relates to routing performance, and still has permanently forbidden turns.

There are a number of works also focusing on approaching fully-adaptive routing, but these are still with VCs [8], [17], [33]. Initially, this could be achieved with additional virtual channels called "escape" channels, where certain packets could resort to in order to avoid deadlocks. Conditional forwarding [34] is a deadlock avoidance mechanism and has similar aspects to our proposed approach. It also uses a function ("conditional forwarding flow control") to determine if a packet can follow a restricted path. It aims to increase flexibility in VC allocation, however. This is achieved by eliminating the need to have separate escape channels, which is a common aspect with our approach on (V)OQs.

7 Discussion and Future Work

In this section, we discuss the usage aspect of the proposed approach, especially with respect to future adaptations and improvements. The discussion is divided into the themes of *implementation efficiency*, *adaptation* and *improvements*.

Implementation efficiency: There could be additional and/or different simplifications to F' (see section 3.3) based on restrictions or architectural assumptions that still satisfy F. One such example could be a quantisation of the queue length measurements for reducing the related signals. Another example is to force the 1st hop of all packets to follow the turn model so that q_C will not need to be checked alongside the other queues that contribute to the contents of $q'_{\circlearrowleft}$ and $q'_{\circlearrowright}$. The trade-offs between the decisions in such a design space, especially with respect to circuit complexity and algorithm performance would be interesting to explore.

As with other routing algorithms, the model can also be used in pipelined implementations for hardware scalability and performance. With respect to flow control, a credit system can be used to replace the presented ready signals denoting FIFO space availability [2].

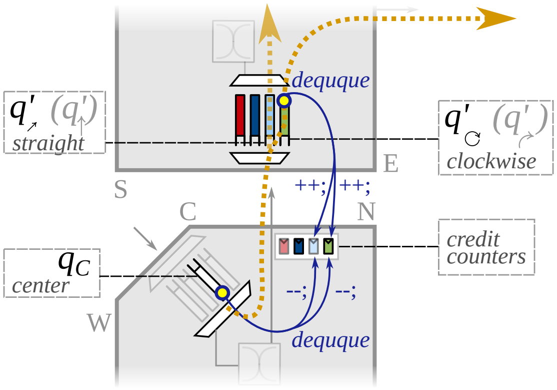
dequque
$q'_{\nearrow}$ $(q'_{\uparrow})$
straight
$q'_{\circlearrowright}$ $(q'_{\curvearrowright})$
clockwise
E
S
++; ++;
C
N
q_C
center
credit counters
--; --;
W
dequque

Fig. 10. Flow control credits can be reused for pipelining F (shown F')

Similarly, as shown in figure 10, some of the flow control credit counters can be reused to estimate the freedom condition. The example shows a packet from the upstream router that can pass through $q'_{\uparrow}$ and the originally-forbidden $q'_{\Rsh}$ of the downstream router. The upstream router already maintains a credit counter for each buffer of the downstream router (the buffers in the channel connected to the upstream router) by using a credit-based flow control. When it is dequeued, both of the counters of the corresponding queues shall be greater than 0, and are both decremented by 1. In the downstream router, when the packet is dequeued from either $q'_{\uparrow}$ or $q'_{\Rsh}$, both corresponding counters are incremented by 1. For the purposes of F, the only useful external queue occupancies are for the forbidden turns ($q'_{\Rsh}$ for this example), hence only consulting the corresponding credit counters. Equivalent arrangements can be made for the multiple-flit case, but it may still be worth duplicating the credit registers for the two purposes (flow control and F) according to the pipelining requirements. Earlier notifications are also possible for restoring the non-taken paths, since this information is known at the time of enqueuing.

Adaptation: The set of system assumptions (section 3.1) is partly due to brevity and practicality, such as for FPGA use. Some of them can be straightforward to remove. Misrouting can be supported at the expense of additional checks accounting for the additional movements that are now possible. These are the U-turns for when a packet does two consecutive clockwise or counterclockwise turns, as well as when a packet goes back and forth through the same port. This results in additional monitoring in the queue occupancies, which are the supersets of the clockwise and counterclockwise directions plus the backward movement. This results in checking all queues of the upstream router that feed the same port, instead of only including q_d for either $d \in \{C, \nearrow, \circlearrowleft\}$ or $d \in \{C, \nearrow, \circlearrowright\}$ in $F(p)$. Similarly, when incorporating interface queues, the formulas can be modified to take the occupancy of additional queues into consideration for calculating the worst case occupancy of the queues of the "forbidden" turns.

The approach can also be used meaningfully in systems supporting both packets of unknown and known size. This is achieved by assuming $F(p) = \mathit{False}$ on packets p of unknown size, while still benefiting the others, since the fallback is localised. Such a scheme would be beneficial as the majority of NoC packets are single or few-flit [17], while streaming traffic is commonly split into fixed packets rather than flits, at least according to the protocol. Similarly, as with WPF [12], it is not necessary for the queues to fit all the flits of a packet, but larger packets will tend to follow the fallback routing more easily.

Improvement: The F condition can be extended to provide additional flexibility. For the queues of the upstream router that feed into originally forbidden turns (e.g. $\{q_C, q_{\nearrow}, q_{\circlearrowleft}\}$ for $q'_{\circlearrowright}$), the notion of the occupancy could also be restricted to only count for the packets whose destination includes the forbidden turn/queue q'_f (e.g. $q'_{\circlearrowright}$), i.e. $occp(q, q'_f) = \sum_{p \in q \wedge q'_f \in next(p)} size(p)$. Future work includes the evaluation of this extension, as the additional checks might lead to implementation complexity tradeoffs.

Finally, the provided routing algorithms "XY/Adaptive" and "XY/O1-Turn" are example uses of the approach. A routing algorithm-focused study could propose fine-tuned variants that combine the best performances that are achieved here per benchmark, for instance.

8 Conclusions

This paper relaxes the turn model for building fully-adaptive routing algorithms on NoCs with an output-queued or virtual output-queued router architecture. These two queueing topologies are useful in applications such as in FPGAs, where deadlock avoidance is traditionally achieved using non-fully-adaptive algorithms. The proposed algorithm is a hybrid algorithm, with a fallback component being activated only locally based on the proposed freedom condition. The provided example routing algorithms "XY/Adaptive" and "XY/O1-Turn" generally outperform older algorithms significantly in simulation. These presented examples are relatively well-rounded across the traffic model selection, though the proposed approach can be used to build other novel routing algorithms and is easily generalisable. An example implementation of "XY/Adaptive" is shown to have minimal overhead on resource utilisation and operating frequency over when not featuring deadlock avoidance for full routing freedom.

Acknowledgement

The support of the JSPS, Japan KAKENHI Grant Number 21K17720 and the Human Capital Initiative (HCI) of HEA, Ireland is gratefully acknowledged.

References

[1] B. Adhi, C. Cortes, Y. Tan, T. Kojima, A. Podobas, and K. Sano, "The Cost of Flexibility: Embedded versus Discrete Routers in CGRAs for HPC," in *2022 IEEE International Conference on Cluster Computing (CLUSTER)*, 2022, pp. 347–356.

[2] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[3] N. Kapre, N. Mehta, M. Delorimier, R. Rubin, H. Barnor, M. J. Wilson, M. Wrighton, and A. DeHon, "Packet switched vs. time multiplexed fpga overlay networks," in *2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*. IEEE, 2006, pp. 205–216.

[4] Y. Huan and A. DeHon, "Fpga optimized packet-switched noc using split and merge primitives," in *2012 International Conference on Field-Programmable Technology*. IEEE, 2012, pp. 47–52.

[5] J. Wang, Y.-b. Li, and Q.-c. Peng, "A performance analytical model for noc with voq router architecture," in *The 2nd International Conference on Information Science and Engineering*. IEEE, 2010, pp. 924–927.

[6] A. B. Ahmed and A. B. Abdallah, "Graceful deadlock-free fault-tolerant routing algorithm for 3d network-on-chip architectures," *Journal of Parallel and Distributed Computing*, vol. 74, no. 4, pp. 2229–2240, 2014.

[7] P. Papaphilippou, K. Sano, B. A. Adhi, and W. Luk, "Experimental survey of fpga-based monolithic switches and a novel queue balancer," *IEEE Transactions on Parallel and Distributed Systems*, pp. 1–14, 2023.

[8] F. Verbeek and J. Schmaltz, "On necessary and sufficient conditions for deadlock-free routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 12, pp. 2022–2032, 2011.

[9] H. Farrokhbakht, H. Kao, K. Hasan, P. V. Gratz, T. Krishna, J. San Miguel, and N. E. Jerger, "Pitstop: Enabling a virtual network free network-on-chip," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2021, pp. 682–695.

[10] A. Ramrakhyani, P. V. Gratz, and T. Krishna, "Synchronized progress in interconnection networks (spin): A new theory for deadlock freedom," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 699–711.

[11] Y. Dai, K. Lu, S. Ma, and J. Chang, "Full-credit flow control: a novel technique to implement deadlock-free adaptive routing," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2022, pp. 1041–1046.

[12] S. Ma, N. E. Jerger, and Z. Wang, "Whole packet forwarding: Efficient design of fully adaptive routing algorithms for networks-on-chip," in *IEEE International Symposium on High-Performance Comp Architecture*. IEEE, 2012, pp. 1–12.

[13] P. Papaphilippou, K. Sano, B. A. Adhi, and W. Luk, "Efficient queue-balancing switch for fpgas," in *2021 International Conference on Field-Programmable Technology (ICFPT)*, Dec 2021, pp. 1–5.

[14] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *ACM SIGARCH Computer Architecture News*, vol. 20, no. 2, pp. 278–287, 1992.

[15] P. Lopez, J.-M. Martínez, and J. Duato, "A very efficient distributed deadlock detection mechanism for wormhole networks," in *Proceedings 1998 Fourth International Symposium on High-Performance Computer Architecture*. IEEE, 1998, pp. 57–66.

[16] T. Van Chu and K. Kise, "Lef: An effective routing algorithm for two-dimensional meshes," *IEICE TRANSACTIONS on Information and Systems*, vol. 102, no. 10, pp. 1925–1941, 2019.

[17] S. Ma, Z. Wang, N. E. Jerger, L. Shen, and N. Xiao, "Novel flow control for fully adaptive routing in cache-coherent nocs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2397–2407, 2013.

[18] R. Holsmark, "Deadlock free routing in mesh networks on chip with regions," Ph.D. dissertation, Linköping University Electronic Press, 2009.

[19] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE transactions on parallel and distributed systems*, vol. 4, no. 12, pp. 1320–1331, 1993.

[20] D. Seo, A. Ali, W.-T. Lim, and N. Rafique, "Near-optimal worst-case throughput routing for two-dimensional mesh networks," in *32nd International Symposium on Computer Architecture (ISCA'05)*, 2005, pp. 432–443.

[21] J. Duato, "A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 10, pp. 1055–1067, 1995.

[22] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, "Catra-congestion aware trapezoid-based routing algorithm for on-chip networks," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2012, pp. 320–325.

[23] J. Hu and R. Marculescu, "Dyad: smart routing for networks-on-chip," in *Proceedings of the 41st annual Design Automation Conference*, 2004, pp. 260–263.

[24] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, 2011.

[25] J. Hestness, B. Grot, and S. W. Keckler, "Netrace: dependency-driven trace-based network-on-chip simulation," in *Proceedings of the Third International Workshop on Network on Chip Architectures*, 2010, pp. 31–36.

[26] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt, "The m5 simulator: Modeling networked systems," *Ieee micro*, vol. 26, no. 4, pp. 52–60, 2006.

[27] C. Wolf, "Yosys open synthesis suite," 2016.

[28] J. Bush, N. Taherinejad, E. Willegger, M. Wojcik, M. Kessler, J. Blatnik, I. Daktylidis, J. Ferdig, and D. Haslauer, "Nyuzi: An open source gpgpu for graphics, enhanced with opencl compiler for calculations."

[29] Xilinx, "Versal architecture prime series libraries guide (ug1344)," 2022. [Online]. Available: https://docs.xilinx.com/r/2022.1-English/ug1344-versal-architecture-libraries/Primitive-Groups

[30] N. Kapre and J. Gray, "Hoplite: Building austere overlay nocs for fpgas," in *2015 25th international conference on field programmable logic and applications (FPL)*. IEEE, 2015, pp. 1–8.

[31] K. Helal, S. Attia, H. A. Fahmy, T. Ismail, Y. Ismail, and H. Mostafa, "Dual split-merge: A high throughput router architecture for fpgas," *Microelectronics Journal*, vol. 81, pp. 51–57, 2018.

[32] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on parallel and distributed systems*, vol. 11, no. 7, pp. 729–738, 2000.

[33] V. Puente, C. Izu, R. Beivide, J. A. Gregorio, F. Vallejo, and J. M. Prellezo, "The adaptive bubble router," *Journal of Parallel and Distributed Computing*, vol. 61, no. 9, pp. 1180–1208, 2001.

[34] Z. Yu, X. Wang, and K. Shen, "Conditional forwarding: simple flow control to increase adaptivity for fully adaptive routing algorithms," *The Journal of Supercomputing*, vol. 72, no. 2, pp. 639–653, 2016.

Philippos Papaphilippou received his PhD from Imperial College London in 2021. His PhD was funded by dunnhumby (Tesco) for researching novel accelerators to improve the performance of big data analytics. He has recently joined Trinity College Dublin as an Assistant Professor for contributing to the Human Capital Initiative (HCI). His research topics include FPGAs, sorting algorithms, network switches, multiprocessor architectures and data science.

Thiem Van Chu completed his Ph.D. at Tokyo Institute of Technology in 2018. Upon graduation, he joined the School of Information Science, Japan Advanced Institute of Science and Technology as an Assistant Professor. In 2020, he moved to Tokyo Institute of Technology. His research interests lie at the intersection of computer architecture, reconfigurable computing, and machine learning.