Highway in TDM NoCs

Shaoteng Liu
liu2@kth.se
KTH Royal Institute of Technology

Zhonghai Lu
zhonghai@kth.se
KTH Royal Institute of Technology

Axel Jantsch
axel.jantsch@tuwien.ac.at
Vienna University of Technology, Austria

***Abstract*—TDM (Time Division Multiplexing) is a well-known technique to provide QoS guarantees in NoCs. However, unused time slots commonly exist in TDM NoCs. In the paper, we propose a TDM highway technique which can enhance the slot utilization of TDM NoCs. A TDM highway is an express TDM connection composed of special buffer queues, called highway channels (HWCs). It can enhance the throughput and reduce data transfer delay of the connection, while keeping the quality of service (QoS) guarantee on minimum bandwidth and in-order packet delivery. We have developed a dynamic and repetitive highway setup policy which has no dependency on particular TDM NoC techniques and no overhead on traffic flows. As a result, highways can be efficiently established and utilized in various TDM NoCs.**

According to our experiments, compared to a traditional TDM NoC, adding one HWC with two buffers to every input port of routers in an 8×8 mesh can reduce data delay by up to 80% and increase the maximum throughput by up to 310%. More improvements can be achieved by adding more HWCs per input per router, or more buffers per HWC. We also use a set of MPSoC application benchmarks to evaluate our highway technique. The experiment results suggest that with highway, we can reduce application run time up to 51%.

I. Introduction

Time Division Multiplexing (TDM) technique is frequently used for guaranteed data transfer in NoCs [1]–[5]. TDM NoC means that a physical link can be shared by different connections, with each connection allocated one or several specific time slots in a finite repeating time window. A connection can span many links from source to destination, by allocating slot(s) at each of the links in a consecutive manner. As illustrated in Fig. 1, connection $v1$ passes link 1 and link 2. If slot 0 and slot 2 of link 1 is allocated to $v1$, then slot 1 and slot 3 of link 2 must be allocated to $v1$. Once a TDM connection is established, packet delivery on the connection is free from contention. It can therefore provide hard guarantees on delay, throughput and in-order delivery. However, in TDM NoCs, quite often the TDM slot utilization is low due to *unused slots* including both unallocated and idle slots.

Firstly, *unallocated slots* commonly exist inside a TDM NoC. TDM NoC requires that reserved slots on the links of a connection must follow a consecutive sequence. Such mandatory sequence makes it difficult to utilize all the slots of links. For example, suppose there is a connection $v4$ wants to use link 3 and link 2 in Fig. 1, although both link 3 and link 2 have unallocated slots, they cannot be allocated to $v4$ since they are not consecutive. Also, when mapping an application onto a TDM NoC, it is common that some links have more bandwidth reservation requirements and some links less. Such unbalanced bandwidth requirements inside a network also cause slots of some links unallocated.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org. NOCS '15, September 28 - 30, 2015, Vancouver, BC, Canada 2015 ACM. ISBN 978-1-4503-3396-2/15/09$15.00 DOI: http://dx.doi.org/10.1145/2786572.2786577

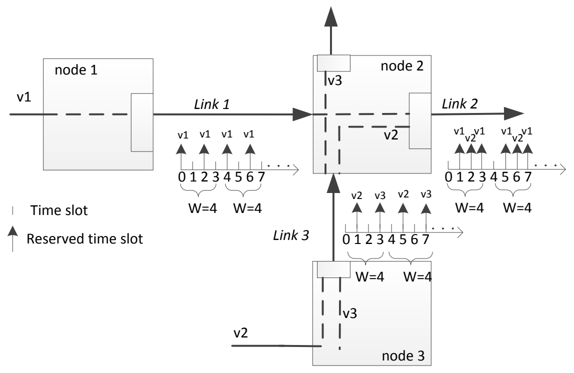


Fig. 1. Illustration of connections in TDM NoC

Secondly, *idle slots* are common for a connection with dynamic fluctuation of network traffic. Idle TDM connections withhold all the pre-reserved slots even if they have no data to deliver. For example, as shown by Fig. 1, suppose connection $v2$ becomes idle, it still occupies one slot on link 1 and link 2, respectively. Also, busy TDM connections can just use their pre-reserved slots for data transfer. For example, no matter how many data flits are waiting on connection $v1$, connection $v1$ can still just use the two reserved time slots in a time window, even if there are free slots available along link 1 and link 2.

In the paper, we develop a new technique called *highway* which can enhance the performance of TDM NoCs by utilizing free slots. With this technique, a TDM connection can dynamically acquire both idle and unallocated time slots to enhance its throughput, while keeping its QoS guarantee on minimum bandwidth and packet order. In particular, our contributions are:

- We develop the concept of highway to efficiently use time slots, which is applicable to many kinds of TDM NoCs.
- We propose a distributed, run-time highway setup and reclaim policy. Whenever the necessary resources for building a highway become available, a TDM connection can make use of it.
- We have made an efficient implementation of our pro-

posed technique. By taking advantage of the contention-free property of TDM NoCs, we can set up a highway with $2D + 2$ cycles and without head flits, where D is the distance between source and destination. Besides, our HighWay Channel (HWC) allocation only needs a very simple allocator, since our highway setup method promises contention free. Thus, our hardware implementation has a relatively short critical timing path.

- We evaluated our highway technique using both synthetic traffic and application benchmarks and suggest how to efficiently use highways.

II. Related Work

Goossens et al. [9]. tried to increase the slot utilization of TDM NoC by introducing best effort (BE) traffic into a TDM NoC (Æthereal), free slots can be utilized to deliver best effort packets. However, the main issue with this mixed guaranteed and BE traffic scheme is that it may cause disorder of arrival packets, as observed in [6]. Eg., suppose a sender sends a BE packet first and sends a guaranteed packet second. At the receiver side, it is possible that the guaranteed packet arrives before the BE packet. This is due to that the transfer delay of a guaranteed packet is bounded, whereas a BE packet has no QoS guarantee. Besides, this solution is not cost-effective, as observed by Goossens et al. themselves [1]. The cost for supporting BE traffic is relatively high due to virtual channels and their allocation. But the service given to BE traffic is low. Since TDM NoC has to prioritize guaranteed traffic, the BE packets can be blocked for a very long time.

To solve the packet disorder problem as in Æthereal, Marescaux et al. [6] proposed a source routing based TDM NoC which provides a new QoS class called SuperGT to increase the slot utilization while maintaining packet order. However, this method suffers from several limitations. Firstly, it is tightly coupled to a source routing based TDM NoC. It forces the traffic flow of a connection to be divided into small packets. The packet size is limited by the number of reserved slots of the connection in a time window. Besides, each packet must have a head flit, since the head flit's information is needed for source routing, virtual channel setup, connection identification, and packet order maintenance. Secondly, it requires that each TDM connection must have at least two reserved TDM slots in a time window, and all reserved slots must be adjacent. Otherwise, the superGT technique cannot be utilized. Eg., connection $v2$ and $v3$ in Fig. 1 cannot use superGT because they only reserve one slot per time window. Connection $v1$ in Fig. 1 cannot use the superGT technique as well, because its two reserved slots are not adjacent. Because of these limitations, superGT can only be utilized in restrictive situations. Moreover, this technique always wastes bandwidth for head flit delivery, due to the limitation on packet size. As an extreme case, suppose a connection has two reserved slots in a time window, the packet size of this connection is limited to 2 flits, which means 50% of the throughput has to be wasted on the head flits.

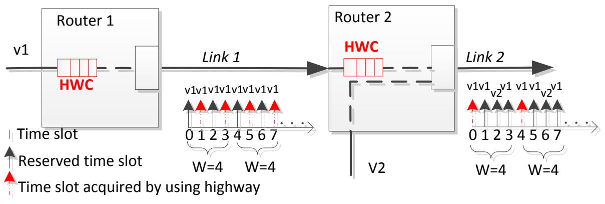


Fig. 2. The function of highway

In contrast, our highway technique can make use of unused TDM slots while keeping the packet order and minimum throughput guarantee. It needs no head flits, has no limitation on packet size, introduces no interference to normal TDM traffic flows, puts no constraint on TDM slot allocation and configuration method, and does not rely on particular routing mechanism or router architecture. As a result, our technique has no architecture dependency and thus can bring benefits to many different kinds TDM NoCs.

III. Highway Concept and Design Considerations

A. Additional Motivation

As we analysed in Section I, low utilization due to unused TDM slots is a common problem in TDM NoCs. Besides, this technique has to be leveraged in order to suit the needs of dynamic and mixed traffic scenarios. Consider the following practical situations: 1) Dynamic traffic: For streaming applications [7], [8], e.g., H.264 decoding, we just need the NoC's promise on the lower bound communication bandwidth. The upper bound of a traffic flow can be dynamically and readily adjusted by applying a flow control mechanism or adding flow regulation components. 2) Mixed traffic: As demonstrated in [6], consider a system with a guaranteed connection between a L1 cache and L2 cache. Since cache misses are not completely predictable, it is common practice to over-allocate bandwidth. Thus we might want a mechanism which can keep the guarantee on minimum throughput and on the predictable traffic flow, while offering additional non-guaranteed bandwidth to enhance the overall system performance or absorb peaks of less predictable traffic flows.

B. The Concept of Highway

Our proposed highway is an express path for a TDM connection. Based on an established TDM connection, it can speed up data transfer by using unused time slots along the links of the connection. This is made possible because a highway consists of one buffer queue at the input port per router. Arbitration is performed to use the free time slots of the output link of a router. We name these buffer queues as *highway channels (HWCs)*.

The function of a TDM highway is illustrated in Fig. 2. The two connections, $v1$ and $v2$ share link 2 in such a way that $v1$ reserves two slots (slot 1 and 3) of link2; $v2$ reserves one slot (slot 2) of link 2. Then, $v1$ also builds up a highway path by occupying one HWC in router 1 and one HWC in router 2. Thus, $v1$ can additionally use slots of link 1 and link 2 whenever they are free. For example, it can acquire slot 1 and 3 on link 1 and slot 0 on link 2. It can also acquire the slot 2 of link 2, if $v2$ does not use it. In contrast, connection $v2$

has no HWC and thus can only use the reserved one slot for data delivery.

A HWC functions like an input queue: if the output link of a router is occupied, incoming flits of a connection with a highway will be buffered in the HWC. When the output link becomes free, or a reserved slot of the connection is coming, the output link can immediately serve the HWC.

C. Design Considerations

The idea of our highway technique sounds simple. However, the details are complicated, especially when we try to make it commonly used for TDM NoCs. The principles and considerations are listed as follows:

1) With or without a highway, a TDM connection is promised to use its pre-reserved slots to offer a guaranteed service. Due to this principle, if a highway is used for a TDM connection, whenever any of the pre-reserved slots arrives, the queue of the corresponding HWC will be served. For example, as Fig. 2 suggests, connection v1 has an established highway. Therefore, at slot 1 or 3, its HWC in router 2 is guaranteed to be served. This is in contrast to normal virtual channels which have no service guarantee. Moreover, this principle also requires that the dynamic highway setup/release process should incur no additional traffic to the guaranteed traffic flow.
2) We must guarantee that reordering of a traffic flow never happens in any situation. This rule sounds easy but it is tricky to follow. In Section IV-D, we will show our solution to keep flits ordered during the highway release process.
3) A highway accelerates data transfer only if one HWC per router is allocated along the path of a connection, all the way from the source to the destination. This is due to that individual HWCs cannot ameliorate performance. For example, suppose connection v1 in Fig. 2 gets a HWC in router 1 but fails to get a HWC in router 2, then link 2 will be the throughput bottleneck, since still only the two reserved slots of link 2 can be used. Thus, during the highway setup process, our HWC allocation follows a *win all or nothing* principle that, if we fail in allocating one HWC in any of the routers along a connection, all allocated HWCs are canceled as soon as possible.

IV. Highway in TDM NoCs

A. Design Overview

An overview of a TDM router with highway is depicted in Fig. 3. Each input link has an input manager, which manages all the incoming flits of that input. We can have one or several HWCs per input manager. With more than one HWC, the input manager needs additional logic for internal arbitration between the HWCs. To support our highway technique, each input /output link needs to have a $flag$ signal and a $credit$ signal coupled with the data path. The $flag$ signal is in parallel to the data path. It is used for highway setup, transfer and release. The $credit$ signal goes in the opposite direction of

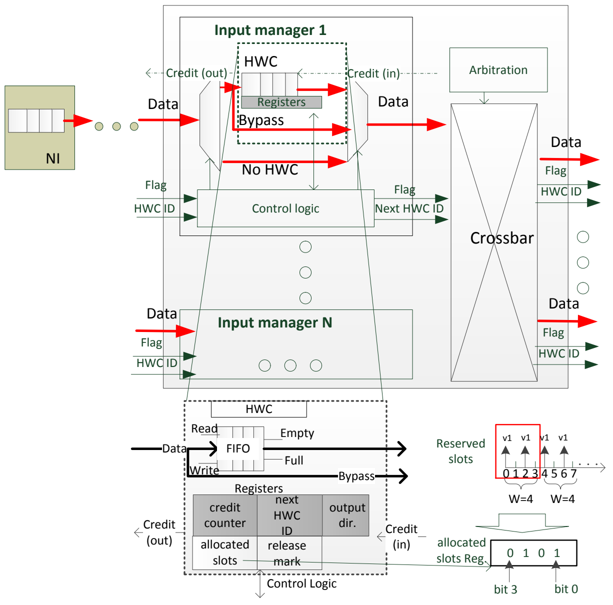


Fig. 3. The micro-architecture of a TDM router with highway

the data path. It is used for flow control in the data transfer phase and for Ack/Nack purpose in the highway setup phase. As Tables I and II suggest, we use 3 bits for the $flag$ signal and 2 bits for the $credit$ signal.

Because of the design considerations listed in Section III-C, we do not use head flits in our highway technique. Instead, we use the 3-bit flag signal for highway setup and release purpose. Unlike the information bits of a flit, a flag associated to a flit can be changed during the transfer process. It does not need to be buffered together with the flit. Instead, it is regenerated when a flit leaves a router, based on the router's current status.

Inside each HWC, there is a set of registers. The "next HWC Id" register stores the id of the HWC in the next router[1]. The "out dir." register stores the output direction to reach the the next router. The credit counter records the available queue size of the downstream HWC. All of these 3 registers are commonly used in virtual channel techniques. The "release mark" register is used in our highway release process. The "allocated slot" register uses a vector of bits to mark which are the reserved slots in a time window of a connection. The size of the bit vector equals the time window size. For example, as Fig. 3 describes, if the reserved slots for a connection are slot 0 and slot 2 inside a time window of size 4, the "allocated slot" register is configured as "0101".

In our design, HWCs are dynamically allocated and reclaimed for connections. Note that, a HWC can only be assigned to one connection at a time.

When a TDM connection has a certain amount of data buffered in the network interface, it may set up a highway for acceleration. The general operation for using a highway in a TDM NoC consists three phases, namely, *setup*, *data transfer*, and *release*.

[1]We do not need this register if each input manager only has one HWC.

TABLE I
USAGE OF THE FLAG SIGNAL

Flag	message	Usage
000	Idle	–
001	Setup	try to book a HWC
010	HWC-RS	Incoming flit has a HWC and sent/received at a reserved slot (RS)
011	HWC-NS	Incoming flit has a HWC and sent/received at a non-reserved slot (NS)
100	No HWC	Incoming flit has no HWC
101	Release	Release the allocated HWC

TABLE II
USAGE OF THE CREDIT SIGNAL

Credit	Usage during highway setup	Usage after highway is built
00	Idle	–
01	–	credit updating
10	Nack (highway setup failed)	–
11	Ack (highway setup success)	–

B. Highway Setup Phase

Fig. 4 illustrates the setup process for a 2-hop TDM connection which has 2 reserved slots inside a time window of 4. From the source node, when a reserved slot (slot 0) is coming, a data flit is sent out with $flag$ signal "setup". This flit travels by using reserved slots of links, at a constant speed of one slot per hop. If it can acquire a HWC when arriving at a router during the traversal, its flag remains "setup". When the destination node receives a flit with flag "setup", it will send back an "Ack" message by using the $credit$ signal of the allocated HWC in the destination node. Such a message will be delivered backward to the source node hop by hop, through the $credit$ signal of each allocated HWC along the forward path of the flit. When the source node receives "Ack" , it can begin to use the established highway for data transfer.

The following is worth mentioning.

1) The HWC allocation performs one hop in advance. For example, as suggested by Fig. 2, router 1 is responsible to allocate the HWCs at the input port of router 2. The allocation decision is put into "next HWC Id" reg. This technique is inherited from the virtual channel allocation technique [12].
2) If the flit with flag "setup" fails to acquire a HWC in a router, the highway setup process stops. The flit continues its traversal with $flag$ changed into "No HWC". Meanwhile, a "Nack" signal will be sent back towards the source node hop by hop through the $credit$ signals of the allocated HWCs in routers. As the "Nack" signal travels, it will release the allocated HWCs.
3) Since "setup" flits are delivered without contention by pre-reserved slots of links, arbitration is not needed in the setup phase. Thus, we can simplify the logic for the HWC allocation inside each input manager. Moreover, the highway setup time is predictable. It is $2D + 2$ cycles (assuming each slot is one cycle), where D is the distance between source and destination.
4) The whole setup process just utilizes $flag$ and $credit$ signals, it generates no additional traffic flow overhead.
5) Our highway setup method is architecture independent. We do not care about how flits of a connection are routed or how connections are established. As long as a flit is delivered without contention to the destination with a $flag$ as "setup", a highway has been established [2].

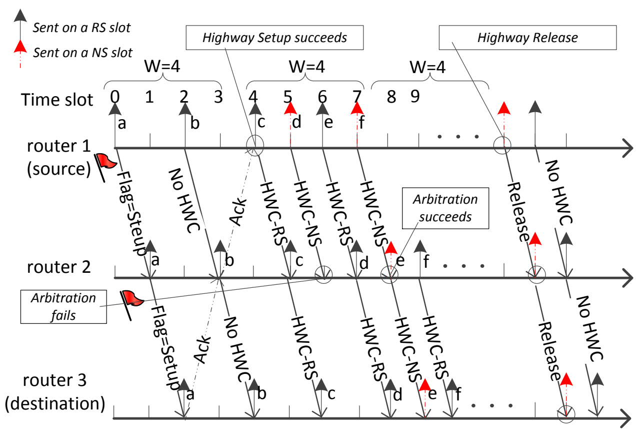


Fig. 4. Highway setup, data transfer, and release process. With highway, the RSs (reserved slots) on a link are still guaranteed to use, while a NS (non-reserved slot) can be acquired by wining an arbitration.

The highway configuration process configures the registers inside each allocated HWC. During the setup phase, the flit with flag "setup" configures the register "next HWC Id" and "out dir." The "allocated slots" register configuration takes place in data transfer phase.

In the case of failed highway setup due to HWC unavailability, we use a simple retry scheme until success.

C. Highway Data Transfer Phase

After a highway is established, flits sent at a reserved slot have the flag "HWC-RS", whereas flits sent at non-reserved slots have the flag "HWC-NS", as illustrated in Fig. 4. Arbitration for a non-reserved slot is needed, if it is requested by multiple input managers.

The "allocated slots" register of a HWC is configured by multiple "HWC-RS" flits arrived after the highway setup process, if the connection has multiple reserved slots. When a flit with flag "HWC-RS" arrives at a HWC, it will set the corresponding bit of the HWC's "allocated slots" register according to the ID of the current slot, if the bit is unset ("0" means unset and "1" set)[3]. For example, if a "HWC-RS" flit arrives at a HWC on slot 0, it will set the bit 0 of the register.

The function of a HWC in the data transfer phase is somehow similar to input queue based virtual channel. Incoming data flit of a HWC is pushed into a queue according to the flit's $HWC\ ID$. If the desired output channel is granted, one flit will be popped from the queue and sent with a valid $HWC\ ID$ pointing to the HWC in the downstream.

One specialty is that, when a reserved slot of an HWC comes, the queue of the HWC is guaranteed to be served. Besides, the transfer of "HWC" flits does not interfere with "No HWC" flits, since our arbitration mechanism prioritizes the data flit sent by using reserved slots.

[2]In source routing based TDM NoCs, we need to add a slot pointer inside each router to obtain the ID of the current slot for the "allocated slots" register configuration.

[3]Due to slots are reserved consecutively along the links of a connection, when an HWC receives a "HWC-RS" flit from its upstream node, it means its own reserved slot is met.

A FIFO is implemented in an HWC for flit buffering. However, we also add a bypass way to the FIFO. If the FIFO is empty and the desired output is ready, an incoming flit can be directly forwarded, without being buffered in the FIFO.

The credit based flow control policy used in our HWCs is similar to that in Chapter 13.3.1 of [11], except that sending data flit at reserved slots neither consumes credits nor generates credit updating signal to the upstream.

D. Highway Release Phase

If a connection no longer requires a highway, it should release all the occupied HWCs.

To release a highway, the source node sends out a ”release” flag. If a HWC is empty when a ”release” flag arrives, the ”release” flag will reset the HWC and get forwarded to the downstream node. However, if a HWC still has buffered data, the ”release” flag will set the ”release mark” register and halt its forwarding until the HWC becomes empty.

As mentioned in Section III-C, it is tricky to maintain flit order of a connection during its highway release process. Let’s consider the following situation: Suppose the source node of a connection has sent out a ”release” flag. After that, the source node continues to send flits with the flag ”No HWC”. The HWC release process is relatively slow, since the ”release” flag may wait inside a HWC until it becomes empty. However, the flit sent by the source node with the flag ”No HWC” travels at a guaranteed speed of one slot per hop. Therefore, a ”No HWC” flit of a connection may arrive at a router which still has an unreleased HWC for that connection. In this situation, this ”No HWC” flit should go into the unreleased HWC to maintain the flit order. However, since all the upstream HWCs have been released, this flit arrives without a valid $HWC\ ID$. How can we find the HWC for this flit?

Unlike [6], we do not use a head flit for carrying the connection ID. Instead, we use the reserved slot information for connection identification. Let us also consider the following facts:

1) ”No HWC” flits are delivered by using reserved consecutive slots of links.
2) ”allocated slots” register of a HWC can be used to claim which slot is the reserved slots on a link of a connection.

Therefore, when a ”No HWC” flit arrives at an input manager of a router, normally it will be directly forwarded to an output. However, if a HWC at the same input manager also claims that it meets a reserved slot and must be served, the incoming flit and this HWC must belong to the same connection and thus the incoming flit should go into the HWC. In this way, we can identify whether or not a ”No HWC” flit has an unreleased HWC.

E. Implementation Cost

Our highway TDM router is built on top of a base TDM router (similar to [2]) used for mesh topology. The additional components are 1) one input manager per input port, 2) an external arbitrator for the arbitration between all the input managers. Note the buffer size per HWC should be large enough to cover the round-trip delay of credit updating (see Chapter 13.3.1 of [11]). Thus, suppose that each slot is one clock cycle, and the credit-updating signal is one cycle per hop. Depending on the implementation details, the round-trip delay is at least two cycles[4]. Thus, the minimum buffer requirement is two flits. We evaluate the additional costs in this section. The synthesis results are reported under 45 nm technology.

TABLE III
SYNTHESIS RESULTS WITH 45 nm TECHNOLOGY WITH FLIT WIDTH OF 128-BIT AND A BUFFER SIZE OF 2 FLITS

Component	Comb.	Non-comb.	Flip-flops	Total (um^2)	Total (gates)
HWC	298	1588	282	1886	2694
Other logic	457	28	5	485	693
Input manager total	755	1616	287	2371	3387

The area costs of an input manager containing one HWC with 2 buffers and with a data flit width of 128-bit and 16-slot window size are listed in table III. In our implementation, each slot represents one clock cycle. As Table III shows, the total area cost of a HWC manager is 2371 um^2, among which 1886 um^2 is spent on the HWC. Inside a HWC, FIFO constitutes about 92% of the area. In our implementation, FIFO is built by using flit-flops. If we use SRAM cells, it can be a factor of 3 or 4 less expensive. The cost of the external arbitrator is only 161 um^2. It does not scale up with the HWCs per input.

The critical timing path of a router consists of 3 components: an input manager, the external arbiter and the crossbar. The latter two components contribute a latency of 0.15 ns. The latency of a input manger varies with the number of HWCs it has. E.g., with one HWC, its latency is 0.36 ns; with 8 HWCs, its latency is 0.54 ns.

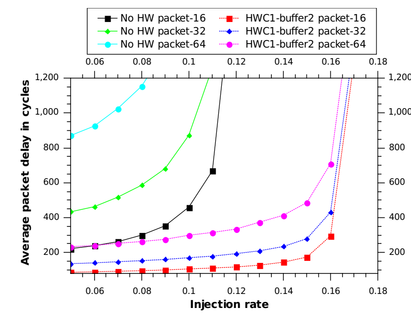
Compared to the base TDM router, our highway TDM router (adding 1 HWC with 2 buffers per inport) causes an area overhead of 12016 um^2. This is the cost of leveraging the static inefficient QoS guarantees.

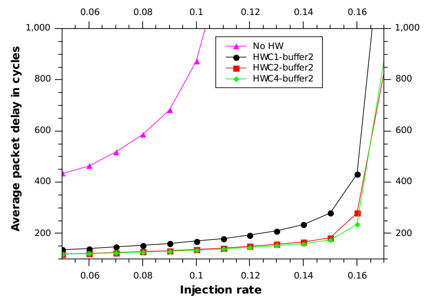
There are also additional costs on the network interface (NI) when applying the highway technique. We need a state machine to control the setup/release of highways, as well as credit based flow control logic for data transfer. If a NI uses more than one highway, arbitration logic is also needed. However, since we do not need to increase the flit buffers inside a NI, the additional costs in total are still relatively small.

V. Performance experiments and results

We present experiments and results concerning the benefits of using highways in different configurations, under different test scenarios, and with different TDM NoC technologies. We utilize a popular mesh topology in our evaluation. To facilitate our evaluation, we assume that each slot is one clock cycle, and each HWC should have more than 2 buffers to cover the credit round-trip delay. The performance enhancements brought about by our highway technique are evaluated with both synthetic traffic patterns and application benchmarks.

[4]In this case, the credit updating signal needs to be sent out as soon as the arbitration succeeds.


(a) Packet sizes


(b) Increase HWCs

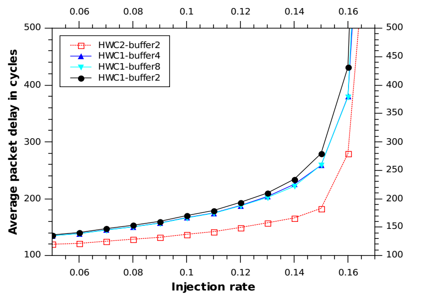
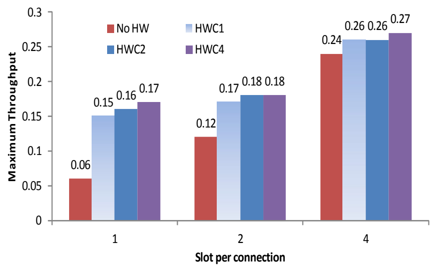
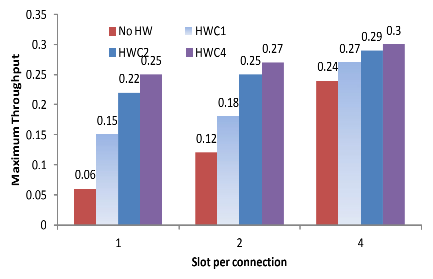

(c) Increase buffers

Fig. 5. Performance evaluation under uniform traffic in an 8x8 mesh, with a window size of 16 slots. Each connection reserves 2 slots in a window. The packet size is 32 flits for (b) and (c). Injection rate is in flit/connection/cycle (fcc)


(a) Uniform random


(b) Tornado

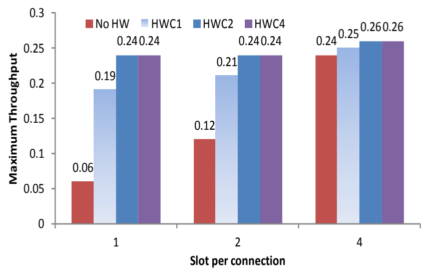

(c) Shuffle

Fig. 6. Throughput evaluation under different traffic patterns and different connection width and a packet size of 32 flits

A. *Performance Evaluation with Synthetic Traffic*

We assume that the static TDM NoC has an architecture similar to [2]. We designed a depth first searching algorithm similar to [10] to do path searching and slot allocation for connections. After all the connections are configured, we can launch our NoC for data transfer by using these statically reserved connections.

For a uniform random traffic pattern, we randomly generated 64 connections and scheduled them in an 8×8 mesh with a window size of 16 slots. If a generated connection cannot be scheduled, we will regenerate it until it can be scheduled. We also use the Tornado and Shuffle traffic patterns for which connections are generated according to the relationship of source and destination nodes as described in the Chapter 3.2 of [11].

Packets generated for each connection obey a Poison distribution. Therefore, the traffic injection rate of a flow can be controlled by adjusting the inter arrival time between packets, while the burstiness of the traffic flow can be controlled by varying the number of flits in a packet. We vary the packet size from 16 to 64 flits, the reserved number of slots per connection per time window from 1 to 4 slots, the HWC per input from 1 to 4 and buffer per HWC from 2 to 4.

In our evaluation, we define *injection rate* as average flits/connection/cycle (fcc). The packet delay includes both the waiting delay in the NI and the transfer delay of the network. The *throughput* results are also given as flits/connection/cycle.

The results under uniform random traffic are shown in Fig. 5a. Our highway technique greatly reduces the average packet delay. Eg. with a packet size of 16 flits, applying one HWC per input with 2 buffer stages can achieve a delay reduction of 77% at injection rate 0.1 fcc. As the packet size grows, the benefits of using highways becomes more prominent. Eg. at packet size of 32 flits, the delay reduction is 80% at injection rate 0.1 fcc, while at packet size of 64 flits the reduction becomes 84%. Besides, the maximum throughput improvement also increases from 50% to 200%, when packet size increases from 16 to 64 flits. This is because when the packet size grows, the network traffic becomes more and more bursty and thus unbalanced. As a result, our highway technique gains more chances to utilize free slots for busy connections.

We also observe that, increasing the number of HWCs per input can further improve performance. As Fig. 5b shows, increasing HWC from 1 to 2 can have a further 35% delay reduction at injection rate 0.15. However, more HWCs do not lead to apparent performance improvements, showing a performance saturation phenomenon.

Compared with adding the buffers per HWC, increasing the number of HWCs per input is more effective on performance. As Fig. 5c suggests, two HWCs per input with 2 buffers in each is far better than one HWC with 8 buffers.

Furthermore, we studied the effect of different number of reserved slots per connection and different traffic patterns. All of these results suggest that using our highway technique can greatly reduce packet delay, which is similar to the delay curves in Fig. 5a. For example, when each connection reserves 4 slots in a time window, we can still have 47% delay reduction at injection rate 0.1 fcc and 70% delay reduction at 0.23 fcc with uniform random traffic. We skip the delay results here due to space limitation.

The effects on maximum throughput are shown in Fig. 6. We find that the same HWC configuration under different traffic pattern and with different reserved slots per connection generates different throughput improvements, ranging from

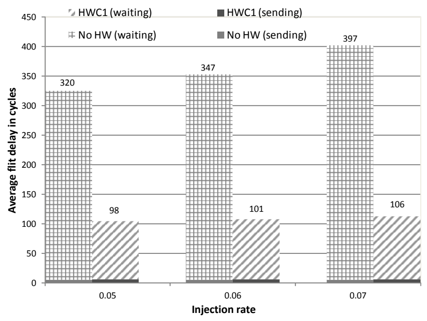


Fig. 7. Delay decomposition under uniform random traffic and a packet size of 32 flits.

417% to 8%. We also observe that, as the reserved slots per connection grows, the throughput improvements decreases. This is due to that given a fixed number of connections, as the reserved slot per connection increases, the unallocated slots decrease.

Finally, in our experiments we separate the total delay into: waiting delay and transfer delay. As shown in Fig. 7, compared with the transfer delay, the waiting delay of a flit is much bigger. For example, without a highway, the average flit waiting delay is more than 300 cycles, which will also increase dramatically when the injection rate increases, while the transfer delay is always 6 cycles. The large waiting delay of a flit is due to two reasons: (1) when a burst of flits arrives, the FIFO order requires the flits waiting in a queue, if the service rate of the output is not enough; (2)Without a highway, a TDM connection mandates a flit to wait for one of the reserved slots. The highway technique can reduce the waiting delay in both scenarios, by increasing the service rate of a connection as well as reducing the waiting time for a slot. As suggested by Fig. 7, the waiting delay is reduced from above 300 cycles to around 100 cycles, when applying 1 HWC with 2 buffers per input. With HWC, the transfer delay of a flit may slightly increased due to buffering delay in the HWCs, for example, at injection rate 0.7 fcc, the average transfer delay with a highway is 7 cycles, which is 1 cycle more than without highway. However, when compared with the waiting delay reduction, such increase can be neglected.

From these results, we find that adding 1 HWC per input with two buffers can have up to 80% packet delay reduction and up to 310% throughput enhancement. Increasing the buffers per HWC does not have significant improvements, whereas using more HWCs can further reduce the packet delay from 10% to 40% and increase the throughput by 4% to 75%. However, considering the doubled, even tripled area cost, as well as the more than the 15% increase on the critical timing path, we think it is not cost-effective to use more HWCs or more buffers per HWC. Applying one or two HWC for each input link, and 2 buffers in each HWC to cover the round-trip delay seems to be a reasonable compromise.

B. Performance Evaluation with MPSoC Benchmarks

We experimented with the NoC benchmarks designed by Pekkarinen et al [14] [15], to confirm the benefits of highways. The NoC benchmark utilizes *task communication graphs* (TCGs) to model MPSoC applications. It contains a set of processor and DSP models. Tasks can be mapped and running on these processor and DSP models. [15] has already given the task mappings. It mapped all the applications to either a 2×2 or a 4×4 mesh based MPSoC platform depending on the size of the TCGs. Users of the NoC bench are required to use their own NoC to connect all the processor/DSP cores to run these applications for evaluation.

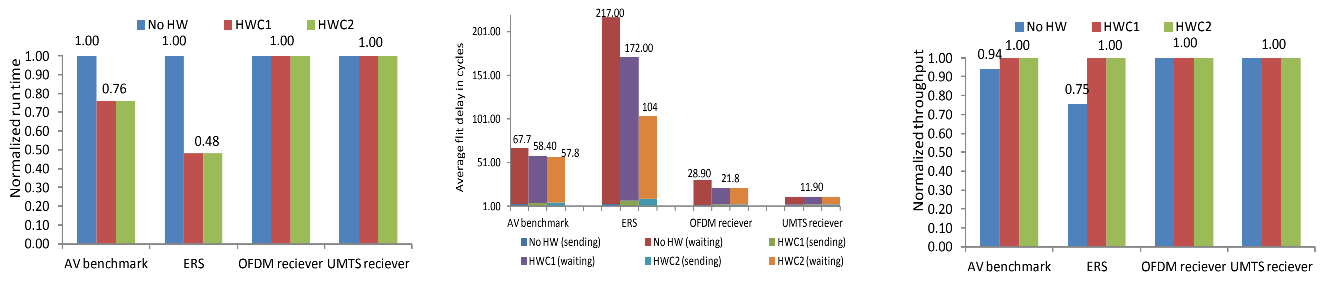
TABLE IV
COMMUNICATION PROPERTIES OF THE APPLICATIONS

	AV bench	ERS	UMTS	OFDM
Connections	57	26	11	12
Avg. Burst size (flits)	531.60	12797.0	14.1	1.43
Total Request (MB/s)	6772	4488	94	196
Max Request (MB/s)	1168	512	246	80
Min Request (MB/s)	0.25	64	2	52

The NoC bench contains four applications which have throughput requirements annotated on the edges of TCGs. The details of the four applications have been described in [14]. Their TCGs can also be found in [15]. Besides, we list the communication properties of each application in Table IV, in which *Total Request* means the total throughput requirement of an application. *Min/Max Request* refers to the minimum/maximum throughput requirement among all the connections' requirements of an application. Note that the AV benchmark and Ericsson Radio System (ERS) are communication intensive, their *Total Request* are 6772 MB/s and 4488 MB/s respectively, whereas the traffic flows generated by the UMTS receiver and OFDM receiver are a magnitude less, which are 96 MB/s and 196 MB/s respectively.

Before running an application on the statically scheduled TDM NoC, we first establish TDM connections between processor/DSP cores to satisfy all the communication needs and throughput requirements described by the TCG of the application. We use a depth-fist search algorithm to search paths and allocate slots to connections. We can statically optimize a TDM NoC for an application by tuning the NoC clock frequency and the TDM window size, in order to avoid too much bandwidth waste. For this reason, we gradually increase the NoC clock frequency and the time window size until reaching a condition where our search algorithm can schedule all the connections of an application. The clock frequency of the NoC ranges from 100 to 1000 MHz. The time window size is between 2 to 32 slots.

As Fig. 8a shows, the highways significantly improve the performance of AV benchmark and ERS. The average application run time is reduced by 24% and 52%, respectively. Meanwhile, as suggested by Fig. 8c, the throughput is enhanced by 6% in AV benchmark and 25% in ERS. Such performance improvement is due to the highways in the TDM NoC. For example, we see 14% (with one HWC per input link) and 15% (with two HWCs per input link) average flit delay reduction with AV bench. We also find about 20% (one HWC per input link) and 52% (with two HWCs per input link) delay reduction in the ERS application. For these two applications, improvements on communication can also help to enhance the overall system performance, which is reflected



(a) Application performance (b) Average flit delay (c) Normalized throughput

Fig. 8. Benchmark evaluation of our highway technique

by the shortening of application run time and the increase of system throughput.

Applications like the OFDM receiver and UMTS receiver have less communication needs and their performance mainly depends on the processor speed. Therefore, from Fig. 8b, we find that for the OFDM receiver, although our highway technique can bring about 25% flit delay reduction, such improvement on communication system does not help to increase the application performance, as suggested by Fig. 8a and Fig. 8c.

We further studied the network performance, as described in Fig. 8b. We find when using highways, applications with larger burst size tend to have more delay improvements. For example, the biggest delay improvement happens with ERS, since its traffic flows has an average burst size of 12797 flits. In comparison, there is no improvement for the UMTS receiver, since it has a low traffic generation rate and the average burst size is only 1.43. In this situation, the highway setup process is seldom initiated because of not enough flits waiting in the network interface. Furthermore, we find that the transfer delay of a flit is very small, ranging from 3 to 7 cycles on average. However, the average waiting delay is relatively large, eg. it can reach up to 213 cycles. Our highway technique can greatly reduces the total delay by shortening the waiting delay, eg. from 213 cycles to 104 cycles.

VI. Conclusions

We have proposed a TDM highway technique to utilize free time slots in TDM NoCs. With the highway technique, the upper bound throughput of a connection is adaptive to link sharing situations, while it still offer QoS guarantees on the lower bound throughput and flit order. The delay of packets are greatly reduced. We can dynamically setup highways on TDM connections. One prominent aspect of our highways is that the dynamic setup method has no interference with the normal TDM data transfer and no dependency on the TDM NoC architecture. Thus, it can be utilized by TDM NoCs with either a distributed or a centralized TDM connection setup method, with either source routing or distributed routing by using distributed slot tables.

We use both synthetic traffic pattern and application benchmarks to evaluate our highway technique. With synthetic traffic pattern we find a delay reduction up to 80% and a throughput enhancement upto 417% in a statically scheduled TDM NoC, as well as up to 80% delay reduction and 17% throughput enhancement in a dynamically allocated TDM NoC. Generally speaking, the more unused slots, the more benefits; the larger the burst size, the more improvements. Also, using more HWCs for a link and more buffers per HWC can provide more performance enhancement. However, the cost-performance study suggests that using one or two HWCs per link and 2 buffers per HWC is most cost-effective in our extensive experiments. With application benchmarks, we confirm that highways can enhance the performance of a TDM NoC. However, the enhancement on NoC can reduce the run time of an application only if it is communication intensive.

References

[1] K. Goossens and A. Hansson, "The Æthereal network on chip after ten years: Goals, evolution, lessons, and future," in *DAC*, 2010

[2] R. Stefan, A. Molnos, and K. Goossens, "dAElite: a TDM NoC supporting QoS, multicast, and fast connection set-up," *IEEE Transactions on Computers*, vol. PP, no. 99, p. 1, 2012.

[3] R. Stefan, A. Molnos, A. Ambrose, and K. Goossens, "A TDM NoC supporting QoS, multicast, and fast connection set-up," in *DATE*, 2012.

[4] A. Hansson, M. Subburaman, and K. Goossens, "Aelite: A flit-synchronous network on chip with composable and predictable services," in *DATE*, 2009.

[5] Moreira, Orlando and Mol, Jacob Jan-David and Bekooij, Marco," Online resource management in a multiprocessor with a network-on-chip" in *ACM* symposium on Applied computing, 2007.

[6] T. Marescaux, H. and Corporaal, "Introducing the SuperGT Network-on-Chip; SuperGT QoS: more than just GT " in *DAC*, 2007.

[7] Mirza, Usman Mazhar and Gruian, Flavius and Kuchcinski, Krzysztof, "Design Space Exploration for Streaming Applications on Multiprocessors with Guaranteed Service NoC" in *NocArc*, 2013.

[8] Ma, Ning and Lu, Zhonghai and Zheng, Lirong, "System design of full HD MVC decoding on mesh-based multicore NoCs" in *Hournal of Microprocessors and Microsystems*, vol. 35, no. 2, pp.217–229, 2013.

[9] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: concepts, architectures, and implementations" in *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.

[10] Zhonghai Lu and Axel Jantsch, "TDM virtual-circuit configuration for network-on-chip" in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* vol. 16, no. 28, pp. 1021–1034, 2008

[11] Dally, William James and Towles, Brian Patrick, "Principles and practices of interconnection networks" in *Published by Elsevier* pp. 245–246, 2004.

[12] D. Becker and W. Dally, "Allocator implementations for network-on-chip routers," in *ACM Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. pp.52:1–52:12, 2009.

[13] S. Liu, A. Jantsch, and Z. Lu, "Analysis and evaluation of circuit switched NoC and packet switched NoC" in Euromicro Conference on Digital System Design (DSD), 2013.

[14] E. Pekkarinen, L. Lehtonen, E . Salminen, et al. "A set of traffic models for Network-on-Chip benchmarking " in *IEEE International Symposium on System on Chip (SoC)*, 2011

[15] http://www.tkt.cs.tut.fi/research/nocbench/download.html