A TDM NoC supporting QoS, multicast, and fast connection set-up

Radu Stefan
TU Eindhoven
R.Stefan@tue.nl

Anca Molnos
TU Delft
A.M.Molnos@tudelft.nl

Angelo Ambrose
UNSW
ajangelo@cse.unsw.edu.au

Kees Goossens
TU Eindhoven
K.G.W.Goossens@tue.nl

***Abstract*—Networks-on-Chip are seen as promising interconnect solutions, offering the advantages of scalability and high frequency operation which the traditional bus interconnects lack. Several NoC implementations have been presented in the literature, some of them having mature tool-flows and ecosystems. The main differentiating factor between the various implementations are the services and communication patters they offer to the end-user. In this paper we present dAElite, a TDM Network-on-Chip that offers a unique combinations of features, namely guaranteed bandwidth and latency per connection, built-in support for multicast, and a short connection set-up time. While our NoC was designed from the ground up, we leverage on existing tools for network dimensioning, analysis and instantiation. We have implemented and tested our proposal in hardware and we found it to compare favorably to the other NoCs in terms of hardware area. Compared with aelite, which is closest in terms of offered services our network offers connection set-up times faster by a factor of 10 network, traversal latencies decreased by 33%, and improved bandwidth.**

I. Introduction

As the complexity of Systems-on-Chip (SoC) increases, traditional bus-based interconnects become limited in terms of efficiency and performance. Networks-on-Chip [9], [4] (NoC) were proposed as a scalable replacement that can cope with the increasing number of on-chip IPs.

SoCs typically execute various, real-time or non real-time, applications which may have diverse requirements from the interconnect, e.g., high throughput for video, low latency to serve cache misses, etc. These applications run concurrently in different combinations denoted as "use-cases". Providing NoC service guarantees, e.g., minimum bandwidth, bounded latency, is crucial for the timing analysis and verification of real-time applications [15]. At the same time, special communication patterns like multicast or broadcast may be required, for example for implementing cache coherence or synchronization primitives, and in certain applications that involve distributed decision making algorithms. Besides the service guarantees and various communication patterns, the NoC implementation should have low cost, and ideally provide fast (re)configuration to adapt to dynamic use case switches. Existing NoC approaches either (i) offer service guarantees but do not support multicast [14], [17], or (ii) provide multicast but at the expense of a high cost [10], [23] or of compromising the guarantees of service [35].

In this paper we propose a Circuit Switching network which supports multicast and offers hard guarantees in terms of bandwidth and latency per connection. Our network uses a time-division-multiplexing (TDM), contention-free scheme and a distributed routing model similar to one of the Æthereal [10] flavors. However, we propose a new configuration infrastructure, that is one order of magnitude faster than Æthereal, and has efficient encoding of the configuration data, thus increasing the speed of setting up and tearing down connections. Our proposal compares favorably in terms of hardware cost and performance to the most cost-effective of the Æthereal models. We support only guaranteed-services (GS) because, as suggested in [11], GS offers a better performance-cost ratio and are in fact the more likely to be required by applications in the embedded domain. We refer to our network as distributed-aelite, dAElite, as for network dimensioning and hardware instantiation we use the standard Æthereal tools, with a modified back-end to generate the new architecture.

The rest of this paper is organized as follows. Section II describes similar network implementations and other related work. In Section III we describe the contention-free routing model which is used by our current proposal. We present the building blocks and operation details of our network in Section IV. Experimental results are presented in Section V while the last section presents conclusions.

II. Related work

Many NoC implementations, either connectionless or connection-oriented, have been proposed in the literature. These networks may offer both Best-Effort (BE) and Guaranteed Services (GS). Networks-on-chip as SPIN [1], xPipes [5], qNoC [8], SoCIN [34], artNoC [28], Quarc [24] and [27], implement a connectionless packet switching approach. QNoC implements quality-of-service through the means of traffic classes, but the guarantees offered are at best statistical. ArtNoC has support for multicast but only from one node at a time. Support for multicast is also provided in [24] and [27]. Another approach is BENoC [21], which uses a bus to complement the services of the NoC. While the NoC would provide high data throughput, the bus would provide low latency messaging, multicast and broadcast. Compared to BENoC the advantage of our approach is that we can provide high-throughput multicast and more multicast connections operating in parallel.

Connectionless packet switching NoCs typically do not offer latency and bandwidth guarantees, thus we do not discuss them further. In the following we comment on the connection-oriented, circuit-switching NoCs, as they are similar to dAElite. Among these we give special attention to [10], as it is the closest approach to ours.

978-3-9810801-8-6/DATE12/©2012 EDAA

TABLE I
COMPARISON WITH NETWORK IMPLEMENTATIONS USING SIMILAR CONCEPTS

Network	Æthereal [10]	aelite [14]	dAElite	Kavaldjiev [17]	Wolkotte [32]	Nostrum [23]	SoCBUS [20]
Link sharing	TDM	TDM	TDM	VCs	SDM	TDM,looped	none
Routing	source/distributed	source	distributed	source	distributed	unspecified[2]	distributed
Connection Setup	GS/BE,guaranteed	GS	dedicated	packet,BE[3]	separate BE	container[4]	packet,BE
End-to-End Flow Cont	headers	headers	separate wire,TDM	none	separate wire	none	none
Connection types	1-1,multicast[5]	1-1,channel trees	1-1,multicast	1-1	1-1	1-1,multicast	1-1

Æthereal [10] is a hybrid network offering both Best-Effort and Guaranteed Services. Æthereal supports three routing models, distributed routing with BE configuration, source routing with BE configuration and source routing with GS configuration. More recent studies [11] suggest that the BE versions of Æthereal are not very cost-effective. For guaranteed services, Æthereal makes use of a routing model called contention-free routing in which each connection may use a link in a given timeslot. Channel trees [13] enhance the performance of this basic scheme, by allowing sharing of timeslots between channels, i.e., connections. This sharing may render invalid the service guarantees per connection, thus are not discussed further.

aelite [14] inherits the GS-only model from Æthereal, and introduces the possibility of using asynchronous and mesochronous links. Although we have not currently investigated this possibility, we believe that the same techniques can be used in dAElite. From here onwards, we will refer to the GS-only version of Æthereal as aelite, without any implications to a particular asynchronous or mesochronous link implementation scheme.

[26] proposed the implementation of multicast in Æthereal using separate connections. dAElite uses instead a broadcast/multicast tree to achieve the same result. Our solution is more efficient since it avoids both using separate channels inside the NI and using the link bandwidth n times, one for each of n destinations. Compared to Æthereal, we also use a more efficient, low-cost connection set-up mechanism. The connection state is stored inside all network elements in a distributed manner and the network configuration mechanism is centralized.

Æthereal and dAElite use a TDM scheme to share the link bandwidth between connections, the model is described in more depth in the following section. One of its advantages is low buffer requirements at router level. Another network that uses a TDM scheme to provide guaranteed bandwidth is Nostrum [23]. Nostrum does not have a fixed TDM wheel size, but instead, the TDM period is linked to the length of looping connections. Multicast is supported by adding more receiver nodes to a closed loop. Nostrum also offers BE communication using deflection routing. One disadvantage of Nostrum is that routing paths, and consequently multicast node sets, must be decided at design time.

[2]The paper only mentions that routes are decided at run-time, possibly they are stored in a distributed fashion inside the routers

[3]Guaranteed connections have preallocated VCs and setup is assumed to always succeed

[4]No explicit connection setup is required, containers can be added and removed at will at runtime by any of the nodes on the route but lack of conflicts must be ensured

[5]The distributed version of Æthereal could in theory support multicast at network level, although a solution for configuring the nework for this scenario was not proposed; multicast was proposed using separate connections for each target

The network proposed in [17] uses virtual circuits (a.k.a. per-connection virtual channels) and round-robin arbitration to provide communication guarantees. Virtual circuits are in general expensive as they require buffers, multiplexers, demultiplexers and separate flow control. The number of virtual circuits per router suggested by the authors is limited 4 due to cost concerns which may restrict the number of simultaneously supported connections.

aSOC [19], [18] implements the same type of static TDM schedule found in Æthereal, but it does not implement the actual end-to-end connections, leaving this task to the IPs.

MANGO [6] is an asynchronous network implementation that uses, as [17], virtual circuits. The routers are not synchronized with one another, and QoS guarantees are given with prioritized virtual circuits. Like Æthereal, connection setup is provided by using a Best-Effort network.

Another possibility for link sharing is SDM, used by [32]. Like our implementation, it makes use of an external network for route configuration, but it does not explicitly specify how this network is implemented. Reported configuration times are higher than those of dAElite.

Some implementations like SoCBUS [20] do not share the link between connections. This approach has a very low cost but it may result in excessive blocking.

Table I summarizes the related approaches to several aspects of the NoC implementation. One key differentiator is the type of routing employed which also has implications on the location to store the connection state. Source routing encodes the packet path in the header of the packet while distributed routing relies on separate routing decisions at each hop. We consider source routing to be too expensive for multicast and broadcast especially if small packets are considered, thus dAElite utilises distributed routing.

III. BACKGROUND: CONTENTION-FREE ROUTING

In this section we present contention-free routing, which is the routing model that we used to provide guaranteed services. Under this model, the bandwidth of each link is split, in the time domain, into a predefined number of timeslots. Each connection receives exclusive use of some of these timeslots from the moment it is set up until torn down, in a typical circuit-switching scheme. A network-wide schedule guarantees that packets never collide and never have to wait for each other (Figure 1), hence the name of contention-free routing. This reduces buffer requirements as well as network traversal time.

Contention-free routing may be used in combination with either source or distributed routing to implement the global communication schedule. In source routing (Figure 2a) the path corresponding to each connection is stored inside the Network Interface (NI) and is sent inside the header of each packet. Slot tables inside the NI control the exact time when

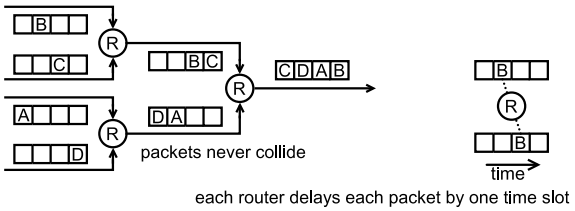

Fig. 1. Contention-free routing

packets are allowed to be inserted into the network. No provisions are made to send data to multiple destinations.

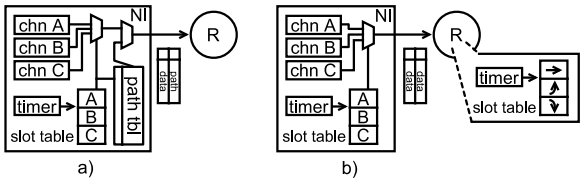

Fig. 2. Source routing (a) and distributed routing (b)

Distributed routing (Figure 2b) uses slot tables inside each router in addition to the ones inside the NIs. Packets are routed based on their time of insertion into the network and their time of arrival at each router. No header is thus necessary and the payload efficiency is higher. Broadcast and multicast can be easily achieved by setting up the router slot tables to forward the data packet to multiple destinations simultaneously. Existing distributed models [10] rely on the Best-Effort (BE) infrastructure for connection set-up which is both expensive and does not deliver guarantees regarding the set-up time [11]. (Centralised models [14] do provide guarantees on reconfiguration, but no multicast.) In dAElite we use the same contention-free model with distributed slot tables but we rely on a different configuration mechanism, which uses a dedicated broadcast configuration network.

IV. dAElite

In this section we present the hardware implementation of our proposed NoC. We present the configuration infrastructure, the router and NI architecture, the network connection setup procedure and the mechanism to achieve multicast.

We assume a SoC platform as the one exemplified in Figure 3. IPs are connected to lightweight local buses which only (de)multiplex transactions to and from different network connections. Network shells have the role of serializing these requests into network messages [16].

A typical usage scenario is that the required connections are set up before starting an application or an execution phase of an application [25]. The application can use the configured connections during that execution phase without further intervention to the network configuration. The connections are torn down once they are no longer needed. Setting up and tearing down connection can be done dynamically without affecting the normal operation of the system, i.e., an application can use certain connections while others are being set up and torn down.

The schedule which guarantees contention-free routing for an application is typically computed at design time, although computation at run-time is also possible [22], [30].

Configuration infrastructure

We implement the network configuration mechanism as a dedicated broadcast network with a tree topology, with links running in parallel to a subset of the normal data network links. One IP, by convention called host, has exclusive control over the configuration infrastructure through a configuration module. The subset of links forming the configuration tree is chosen in such a way as to minimize the distance from the host to any of the network nodes.

The configuration infrastructure is used to set up data connections by updating the contents of the slot tables inside routers and NIs, to configure and read back the state of the network interfaces and to configure the buses adjacent to the network.

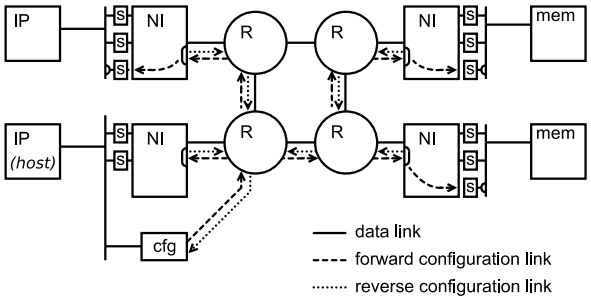

Fig. 3. Example of platform with dAElite

The configuration links consist of a forward and a reverse connection. The forward connection is of broadcast type, that is, each intermediate node forwards the data it receives on its input to all of its outputs. The responses converge on the reverse path in the same tree structure. There is no arbitration on the response path and as a result a policy of only one active request at a time is enforced.

Network Routers

The structure of network routers is presented in Figure 4. Because we are using a distributed routing mechanism each router contains a slot table to store the TDM schedule. Incoming packets are "blindly" routed based on this schedule. In the absence of contention, no link-level flow control is required. A configuration submodule interprets the messages received through the configuration port and updates the schedule. Routers are also nodes in the configuration broadcast tree and they forward their configuration data to a parameterizable number of neighbors.

In dAElite the latency per hop is fixed to two cycles: one cycle for link traversal and one for router crossbar traversal. Data is thus buffered twice inside the router. For reasons of symmetry data is also buffered twice at each hop in the configuration tree.

Network Interfaces

The structure of Network Interfaces is shown in Figure 5. The NI contains a slot table governing both packet departures and arrivals. This is because NIs have to know both when they

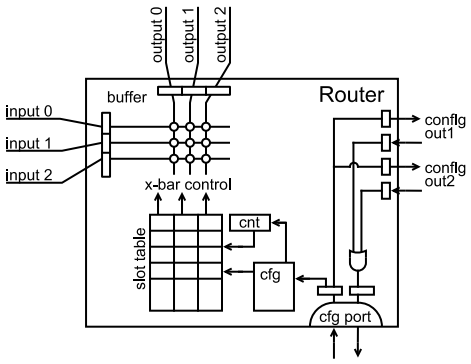

Fig. 4. dAElite Network Router

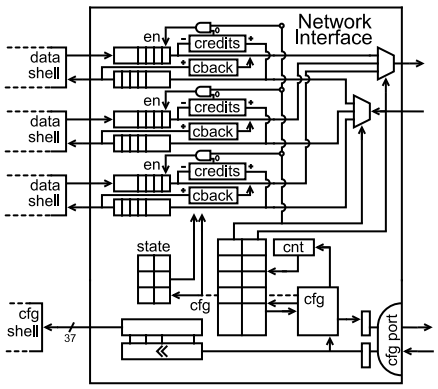

Fig. 5. dAElite Network Interface

are allowed to insert packets into the network, and into which channel queue they have to deposit the arriving packets.

dAElite implements a connection-based network and the NIs perform the function of end-to-end flow control for the connections. We use a credit-based flow control scheme which employs two credit counters for each channel. A counter at the source keeps track of the available space in the destination queue, and a counter at the destination stores the number of words that were already delivered until this value can be sent back to the source.

Connections are bidirectional and credits for one direction are sent on separate bit-lines alongside data in the opposite direction. The separate credit lines and data obey the same TDM scheme and there is actually no distinction between the two at the router level. The bit-width of the credit information is configurable. In our experiments, 3 wires dedicated to sending credit data are enough to send the value of a 6-bit credit counter during each slot cycle.

A configuration submodule, similar to the one inside the router interprets the configuration messages and configures the NI slot table, the credit counters and connection state flags. Reading back flags and flow control information from the NI is supported, as is the configuration of adjacent buses. For the latter, the configuration words are deserialized into wider words which are translated by an NI shell into the appropriate bus standard (DTL in our case) used by the configuration port of the bus.

Connection setup and tear-down

The configuration network supports setting up and tearing down communication channels, configuring and reading back credit information, and configuring the buses adjacent to the network. Due to lack of space we will only present here in detail the path set-up process.

Network configuration, including path setup and tear-down is performed using configuration packets, consisting of several words, transmitted one per cycle over the configuration links. The configuration links have small bit-width, that is equal to the size of the configuration words.

For ease of implementation we have selected a configuration word width that is sufficient to encode a network element ID, a pair of input and output port IDs or the value of a credit counter. The configuration word size of 7 bits, used in our experiments, is sufficient for networks with up to 64 network elements (routers and NIs), routers with an arity of 7, and end-to-end buffers of up to 63 words.

Consider the following example illustrated in Figure 6. A set-up operation is performed for a communication channel using the path NI10-R10-R11-NI11. The host IP in charge of network configuration writes 3 data words to the configuration module using normal write operations. These words are then serialized into 7-bit configuration words. 0-padding is allowed.

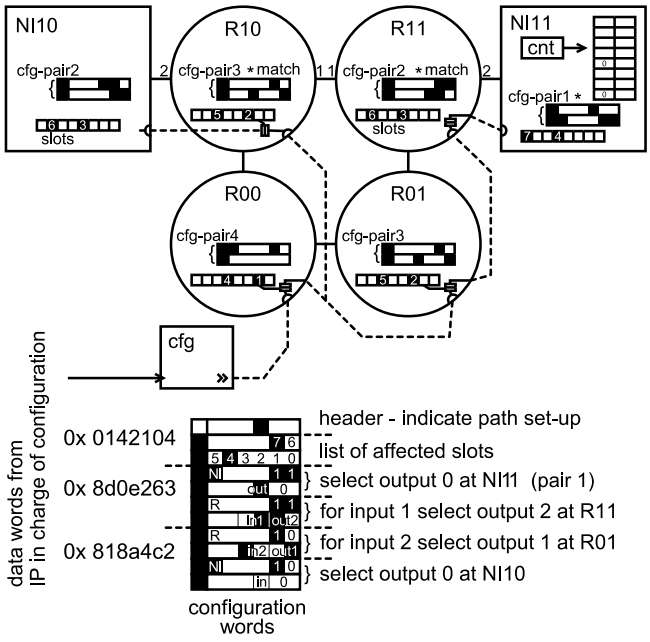

Fig. 6. Path set-up example

The first configuration word is a header that informs the network elements that a path setup sequence will follow. The next two configuration words contain a table of slots affected by this path set-up operation. We assume here a slot table size of 8. The two bits set to one in this example identify slots 7 and 4. These are followed by pairs of configuration words, the first word in each pair representing a network element ID and the second a pair of input and output ports or if the network element was a network interface a single input or output port.

Each of the network elements stores the table of affected slots. For each of the subsequent pair of configuration words, all network elements try to match the first word against its own ID. In case of a mismatch they rotate the table of affected slots by one position, in case of a match they configure the entries in their own slot table marked by the table of affected slots with the data in the second configuration word.

In our example, the first pair of configuration words in the configuration packet after the list of affected slots instructs NI-11 to use output 0 during slots 4 and 7. The second pair instructs router R-11 to forward data from input 1 to output 2 during slots 3 and 6 because the list of affected slots has already been rotated by one position. The third pair instructs router R-10 to forward data from input 2 to output 1, etc.

A cool-down period during which no new configuration packets are accepted, is enforced after each complete path set-up by the configuration module. This allows the routers and NIs to internally update their slot tables.

The list of traversed routers/NIs begins at the destination to ensure that downstream routers are initialized before the upstream NI and routers start sending packets. It is not mandatory that a packet contains a complete source-to-destination NI path, independent path segments can be initialized as well. This is used to set up broadcast or multicast trees, explained below. Tearing down a communication channel is performed in the same way as setting one up, except intermediate routers and the destination NI are instructed to not forward any value to the selected output during the affected slots.

Multicast

dAElite offers a mechanism to achieve multicast that is both simple and efficient. The TDM schedule in a dAElite router is implemented as a table that specifies for each output port which input port should the data be taken from during each cycle. Two (or more) output ports are allowed to use the same input port as a source (Figure 7).

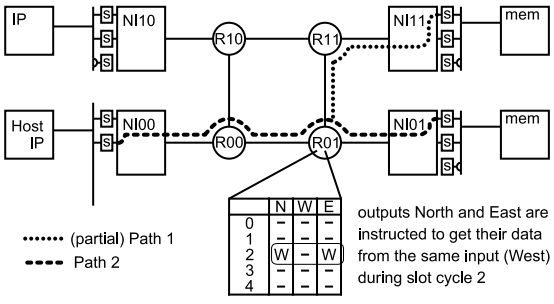

Fig. 7. Multicast in dAElite

The multiple paths to the different destinations form a tree, rooted at the source NI. This is more efficient and offers higher performance than having separate connections from the source NI to all destinations because in the latter case the bandwidth on output link of the source NI would need to be divided between all the connections.

The configuration mechanism allows setting up partial paths; i.e., paths that start at a router instead of a source NI (Figure 7).

All multicast destination shells will receive the same stream of messages and will translate them into the same write commands on the destination IP ports. There is no corresponding multi-destination read, which would require somehow merging read responses.

When using multicast it is necessary to ensure that the destinations can process data at the same rate as it is delivered, as the default flow-control mechanism cannot be used (the source NI only has one credit counter for each communication channel).

V. Experimental results

In this section we compare our proposed network to other NoCs presented in the literature. The proposal most similar to ours in terms of offered services and network organization is aelite, a GS-only version of the Æthereal network. For aelite we will compare the cost of the full interconnect, including network interface shells and adjacent buses. The system setup is the one represented in Figure 3.

For the other networks, we compare the router area reported in the literature with the area of one of our routers with the same parameters: number of ports, link width and, where applicable, number of SDM lanes or TDM slots. Unless otherwise noted the designs were synthesized in the same technology node. In Table II we show the reduction in area our proposal achieves compared to the other networks. The reduction is expressed as $(area_{otherNoC} - area_{dAElite})/area_{otherNoC}$.

TABLE II
dAElite area reduction compared to other implementations

aelite [14] 2x2 mesh, 32 TDM slots (65nm TSMC)	10%
aelite, -/- (FPGA, Virtex-6 slices)	16%
artnoc [28] router, 2-flit buffers, 4 VCs (130nm)	73%
Wolkotte [33] circuit switched router (130nm)	68%
Wolkotte [33] packet switched router (130nm)	91%
Mango [7] router, 8 VCs (120nm) [6]	89%
Quarc [24] 8-port router (130nm) [7]	15%
SPIN [2] 8-port router (130nm)	76%
Banerjee and Wolkotte [3], 5-port router, 4 SDM lanes (90nm)	85%
xpipes lite [31], 4-port router (130nm)	78%

Network Performance

We compare our proposed network in terms of performance with aelite which provides similar services in terms of bandwidth guarantees and same as our network and makes use of centralized configuration. We compare path set-up time based on FPGA execution and give analytical measures in terms of bandwidth and latency.

Table III presents the number of cycles required to set up one connection (request and response path). For dAElite, the set-up time is dependent on path length but not on the number of slots used by the connection. For aelite we provide only the average value reported in the literature and the range of values obtained experimentally, as the set-up time depends on multiple factors: distance from configuration node to the source node and to the destination node, number of slots

[6] the area value reported in the literature is for 120nm technology while our router used for comparison uses a 130nm technology

[7] the Quarc router does not implement a full 8x8 crossbar while our router used for comparison does

used by the connection. The ideal value reported for aelite is taken from [12] and represents the configuration delay without taking into account processor execution time of the configuration code, but only the actual read and writes. The ideal value for our proposal is computed analytically from the number of configuration words that are being written in each case to which the cool-down latency was added.

TABLE III
CONNECTION SETUP TIME

	aelite		dAElite	
	ideal	measured	ideal	measured
6 hops	246	822-1555	60	81
8 hops			68	94
10 hops			76	119
12 hops			84	133

Our FPGA experiments indicate that dAElite configuration is roughly one order of magnitude faster than aelite.

In terms of bandwidth and latency we take into account the fact that both networks are able to operate similar frequency. The FPGA area results in Table II were based on designs both constrained to a frequency of operation of 200MHz while the ASIC synthesis which was unconstrained resulted in frequency of 885 MHz for aelite and 925 MHz for dAElite.

In dAElite, the router (and link) traversal delay is 2 cycles. This is lower than the 3 cycles used by aelite. We are able to achieve this without a negative impact on the clock frequency, because dAElite does not need to look at packet contents before making a routing decision. Routing is performed solely based on the packet arrival time and the contents of router's own slot table. This results in a reduction in the network traversal latency of 33%.

The dAElite TDM slot is 2 words, and could be further decreased to a single word if necessary. In aelite it is not possible to arbitrarily decrease the slot table size because packets contain a header and the header overhead would become higher in shorter packets. A small TDM slot size is useful to improve the scheduling latency (packets need to wait for their turn before they can be inserted into the network).

dAElite has no header overhead. which in aelite is between 11% and 33%: one header is required at least every 3 slots (possibly every slot when slots belong to different connections) and the header represents one third of the slot size.

dAElite allows routing one connection over multiple paths at no additional cost. In [29] it was shown that multipath routing can provide bandwidth gains of 24% on average. Multipath routing is also possible in aelite [29], but with higher area.

Furthermore, aelite reserves at least one slot on each of the NI-router and router-NI links for configuration traffic. For a slot wheel size of 16 this is a 6.25% loss of data bandwidth. This is not the case for dAElite.

VI. CONCLUSIONS

In this paper we have proposed, implemented and evaluated a hardware prototype of a TDM NoC using contention-free, distributed routing, that has the following distinctive features: (i) support for QoS; (ii) support for multicast; (iii) lower area cost than previously proposed implementations and no header overhead; and (iv) configuration and path set-up times significantly shorter than the closest approach.

REFERENCES

[1] A. Adriahantenaina *et al.* SPIN: a scalable, packet switched, on-chip micro-network. In *DATE*, 2003.
[2] A. Andriahantenaina and A. Greiner. Micro-Network for SoC: implementation of a 32-Port SPIN network. In *DATE*, 2003.
[3] A. Banerjee *et al.* An energy and performance exploration of Network-on-Chip architectures. *TVLSI*, 2009.
[4] L. Benini *et al.* Networks on chips: a new SoC paradigm. *Comp.*, 2002.
[5] D. Bertozzi and L. Benini. Xpipes: a network-on-chip architecture for gigascale systems-on-chip. *IEEE Circuits and Systems Magazine*, 2004.
[6] T. Bjerregaard. The MANGO clockless network-on-chip: Concepts and implementation. *PhD Thesis*, 2005.
[7] T. Bjerregaard *et al.* A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip. In *DATE*, 2005.
[8] E. Bolotin *et al.* QNoC: QoS architecture and design process for network on chip. *Journal of Systems Architecture*, 2004.
[9] William J. Dally and Brian Towles. Route packets, not wires: On-chip inteconnection networks. In *DAC*, 2001.
[10] K. Goossens *et al.* Æthereal network on chip: Concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 2005.
[11] K. Goossens *et al.* The aethereal network on chip after ten years: Goals, evolution, lessons, and future. In *DAC*, 2010.
[12] A. Hansson and K. Goossens. Trade-offs in the configuration of a network on chip for multiple use-cases. In *NOCS*, 2007.
[13] A. Hansson *et al.* Channel trees: reducing latency by sharing time slots in time-multiplexed networks on chip. In *CODES+ISSS*, 2007.
[14] A. Hansson *et al.* aelite: A flit-synchronous network on chip with composable and predictable services. In *DATE*, 2009.
[15] A. Hansson *et al.* CoMPSoC: A template for composable and predictable multi-processor system on chips. *TODAES*, 2009.
[16] H. Hansson *et al.* An on-chip interconnect and protocol stack for multiple communication paradigms and programming models. In *CODES-ISSS*, 2009.
[17] N. Kavaldjiev *et al.* A virtual channel Network-on-Chip for GT and BE traffic. In *ISVLSI*, 2006.
[18] A. Laffely *et al.* Adaptive system on a chip (ASOC): a backbone for power-aware signal processing cores. In *ICIP*, 2003.
[19] Jian Liang *et al.* aSOC: A scalable, single-chip communications architecture. In *PACT*, 2000.
[20] D. Liu *et al.* SoCBUS: the solution of high communication bandwidth on chip and short TTM. In *RTECC*, 2002.
[21] R. Manevich *et al.* Benoc: A bus-enhanced network on-chip for a power efficient CMP. *Comp. Arch. Letters*, 2008.
[22] T. Marescaux *et al.* Dynamic time-slot allocation for QoS enabled networks on chip. In *ESTIMedia*, 2005.
[23] M. Millberg *et al.* Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip. In *DATE*, 2004.
[24] M. Moadeli *et al.* Quarc: A High-Efficiency network on-Chip architecture. In *AINA*, 2009.
[25] S. Murali *et al.* Mapping and configuration methods for multi-use-case networks on chips. In *ASPDAC*, 2006.
[26] A. Radulescu *et al.* An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network programming. *Trans. on CAD of Integrated Circuits and Systems*, 2005.
[27] F.A. Samman *et al.* Adaptive and deadlock-free tree-based multicast routing for networks-on-chip. *TVLSI*, 2010.
[28] C. Schuck *et al.* artNoC - a novel Multi-Functional router architecture for organic computing. In *FPL*, 2007.
[29] R. Stefan and K. Goossens. A TDM slot allocation flow based on multipath routing in NoCs. *MICPRO*, 2011.
[30] R. Stefan *et al.* Online allocation for contention-free-routing NoCs. In *INA-OCMC*, 2012.
[31] S. Stergiou *et al.* Xpipes lite: a synthesis oriented design library for networks on chips. In *DATE*, 2005.
[32] P.T. Wolkotte *et al.* An Energy-Efficient reconfigurable Circuit-Switched Network-on-Chip. In *IPDPS*, 2005.
[33] P.T. Wolkotte *et al.* An energy-efficient reconfigurable circuit-switched network-on-chip. In *IPDPS*, 2005.
[34] C.A. Zeferino and A.A. Susin. SoCIN: a parametric and scalable network-on-chip. In *SBCCI*, 2003.
[35] L. Zhonghai *et al.* Connection-oriented multicasting in wormhole-switched networks on chip. In *ISVLSI*, 2006.