Address Interleaving for Low-Cost NoCs

Miltos D. Grammatikakis*, Kyprianos Papadimitriou*, Polydoros Petrakis*, Marcello Coppola†, and Michael Soulie†

*Technological Educational Institute of Crete - GREECE
†STMicroelectronics - FRANCE

***Abstract*—New generations of NoC-based platforms incorporate address interleaving, which enables balancing transactions between the memory nodes. The memory space is distributed in different nodes of the NoC, and accessed alternately by each on-chip initiator. A memory node is accessed depending on the transaction request address through a memory map. Interleaving can allow for efficient use of NoC bandwidth and congestion reduction, and we study whether its gains scale over system size. In this work we concentrate on an instance of a customizable point-to-point interconnect from STMicroelectronics called STNoC. We first evaluate a setup with 4 CPU initiators and 4 memories, and show that interleaving relieves the NoC from congestion and permits higher packet injection rates. We also show that this depends on the number of packets sent per transaction by an initiator prior to changing destination memory node; this is called interleaving step. We then enriched the setup with several DMA engines, which is in accordance with industry roadmap. We experimented with MPSoCs having up to 32-nodes and for various link-widths of the STNoC. When link-width was 32 Bytes, the aggregate throughput gain from address interleaving was 20.8%, but when we set it 8 Bytes the throughput gain reached 69.64%. This implies silicon savings in SoCs, as it is not always necessary to configure NoCs with wide link-widths.**

***Index Terms*—Network-on-Chip (NoC), network interface (NI), address interleaving, address decoding, throughput, link-width, saturation**

I. Motivation

Embedded computing is changing constantly, and on-chip processing using multiple cores is becoming the dominant solution. Some argue that single-purpose nature of embedded systems make multi-core processing unattractive, and it is true that even today there are several applications where single-core is the preferred solution. But the complexity and variance of processing tasks is increasing, and there is an increasing interest in running multiple applications with different criticality levels onto the same chip. Given the asymmetric nature of processing tasks within an embedded system, multi-core execution permits lower clock rates which results in reducing power consumption, one of the most important factors in the embedded domain.

On-chip networks implement a flexible packet-based communication, aiming to replace shared buses to some extent in developing efficient and cost effective multi-core SoCs [1]. They come with inherent architectural advantages in power consumption, performance, and area over traditional crossbar and bus-based interconnect technologies, which allows for the designers to cope with the growing complexity of heterogeneous architectures while reducing the research, development, and unit costs [2]. In the present work we enhance on-chip networks with an address interleaving scheme, and explore whether it allows for improving the performance of NoC-based MPSoCs, and at the same time reduce the silicon cost and power consumption. The two latter factors are critical in the embedded domain, considering that the amount of tiles interconnected in an MPSoC can be large, e.g. in [3] STNoC interconnects 148 tiles.

Within this scope, it is in the plans of industry to incorporate several low power DDR controllers - called channels - that will be accessed in burst by multiple initiators, such as CPUs, DMA engines, or other peripherals [4]. Having different channels allows balancing the network load amongst them, which can also result in balancing the traffic in the NoC itself. The transactions are split and interleaved within the interconnect and the NIs of the NoC; the decision to which channel a transaction belongs, depends on the address of the transaction. This is realized with an address interleaving scheme implemented in the initiator's network interface (INI). An important parameter set during configuration time is the interleaving step, i.e. the number of network packets sent per transaction by an initiator prior to changing destination memory node. This corresponds to the number of bytes to be written before changing memory node. In the present work we perform a first in-depth analysis of the address interleaving scheme. We experimented with different setups, with and without interleaving, for various number of nodes and link-widths. In the first setup CPUs initiate transactions with variable injection rates, while in the other two setups the DMA engines are the initiators. We varied the interleaving step along with the injection rate so as to find the best match that serves the requirements of an application; the injection rate can be provided by a service-level agreement specifying the requirements in terms of the data arrival rate. An important finding of our analysis is that the gains are higher for a narrow link-width, which accounts for a low-cost, low-power solution.

This paper is structured as follows. Section II has information on the Spidergon STNoC. In Section III we explain the concept of address interleaving, and in Section IV we overview the related work. Section V discusses the setups and the scenario we deployed to evaluate the interleaving scheme. Section VI has the experimental results, and Section VII provides information on the address interleaving implementation along with its hardware cost. Section VIII concludes the paper.

corresponding e-mail address: `mdgramma@cs.teicrete.gr`

II. Spidergon on-chip Communication

It is a ring-based topology comprising three main building ingredients; network interface (NI) that functions as access point to the interconnect onto which initiators and targets are directly connected; simple non-programmable router; and physical link. STNoC relies on network operation, thus it provides programmable services such as changing the routing information and Quality-of-Service (QoS). It can be customized according to given specifications to efficiently interconnect MPSoC components, i.e. CPUs, DMA engines, memories, and peripherals. From the network point of view, each component is a node (called also tile), and information - instructions and data - is transmitted across the network in packets. Each packet carries a header information that allows the receiver to reconstruct the original message. Typically, packets sent from a certain tile follow always the same path. This is determined with a static routing policy, which functions based on the information contained in the packet header.

Prior to transmitting a packet, this is broken down in smaller units called flits that in turn are released into the network. A flit carries the maximum amount of data that can traverse the physical link per transaction. Thus, for a given packet size, the amount of flits comprising each packet depends on the link-width. During customization, along with the topology, the NoC designer tunes-up these parameters, i.e. link-width and packet size. For example, for an STNoC with 16 $Bytes$ link-width, if the packet has 136 $Bytes$, e.g. 128 for payload and 8 for header, this is split up in $136 \div 16 \approx 9$ $flits$.

Each router is equipped with buffer(s) that store temporarily the flits, prior to forwarding them to the next router. A buffer implements essentially the virtual channel (VC) on which the packets are transferred. The routers can be configured with up to 2 buffers each, thus 2 VCs, and each one can be set with a different priority, i.e. high and low. This comes at the expense of additional hardware cost, but it allows to simultaneously store packets from different flows onto the same router, which accounts for congestion reduction. Different traffic policies can be activated depending on the amount of available VCs; this relates to the QoS of NoC. A traffic policy is that of assigning priorities for the different flows sharing the same VC. This can be done with the QoS Fair Bandwidth Allocation (FBA), a low cost arbitration algorithm that distributes the NI target bandwidth among different initiators during peak request time. This algorithm is enabled at the garnet router and allows for arbitrating packets from different flows that compete for the same output port of the router. It is worth noting that in older STNoC architectures - until 2006 - packets of the same flow could switch amongst the two VCs, from VC0 to VC1 and vice-versa, while transferred across the network. This mechanism was abandoned as it turned out that switching amongst different VCs didn't offer considerable increasing returns.

Typically, STNoC runs at $1GHz$, thus once a packet transmission begins each flit is injected from the NI every $1ns$. In particular, after the head flit of a packet is released into the network, the remaining flits are transferred in a pipelined manner, thus 1 $more\ clock\ cycle\ per\ flit$ is needed for delivering the entire packet to the target NI. Packet routing is static, source-based, following the across-first strategy.

Recapitulating, the different STNoC configurations offer a wide range of choices, but they are related to cost. On-chip networks with 8 $Bytes$ link-widths cost less than 32 $Bytes$, and 1 VC costs less than 2 VCs, thus selecting wisely the parameters prevents from increasing unnecessarily the hardware cost. This can become huge, considering that existing SoC products using STNoC such as the STiH312 Cannes Ultra platform interconnect up to 148 tiles [3].

III. Address Interleaving in STNoC

Interleaved memory was initially proposed to compensate for the slow speed of dynamic random-access memory (DRAM), by distributing memory addresses evenly across memory banks. In this way, contiguous memory requests are headed towards each memory bank in turn, i.e. they are accessed sequentially, one at a time. This results in higher memory throughput due to the reduced waiting time for the memory banks to become available for the given operation, write or read. Consequently, interleaving helps in hiding the memory refresh time of DRAMs.

The STMicroelectronics STNoC is an Interconnect Processing Unit (IPU), whereas an integrated interactive EDA design flow composed of the iNoC and Meta-NoC environments, low- and intermediate-level device driver generators, and validation subsystems, can be used to design a custom cost-effective NoC solution on a single die that may include optional HW/SW components at the network interface (NI) and possibly NoC layer. These components jointly implement advanced system/application services via a set of low-level primitives. Optional STNoC services include end-to-end QoS via an FBA protocol implemented at NoC layer and target NI, firewall mechanisms implemented at the initiator NI, or fault tolerant routing by implementing on-the-fly post-silicon re-programmability of the routing function at the NI. STNoC technology has enriched its QoS features with the implementation of address interleaving.

STNoC uses source-based routing, i.e. the routing path depends only on the source address of the request transaction. Address interleaving is implemented in the shell (IP-specific subsystem) of the initiator network interfaces, on top of the NoC layer, which provides the routing service including packetization and size/frequency conversion. Address interleaving is able to modify the current STNoC memory map settings. If instantiated during STNoC configuration time, it can be switched on/off and reprogrammed at runtime through a set of registers. When address interleaving is disabled, the classical address decoding applies.

In order to guarantee flexibility and data integrity, all INIs must be able to route traffic consistently towards the same address space at runtime. Reprogramming of address interleaving registers across all related INIs is supported via a coherent reprogramming system driver which allows an external "secure code" to stop incoming traffic, reprogram all related INI, wait for reprogramming to end, and then restart NoC traffic.

Figure 1 illustrates the operation of STNoC address interleaving. Different pages are mapped to different memory

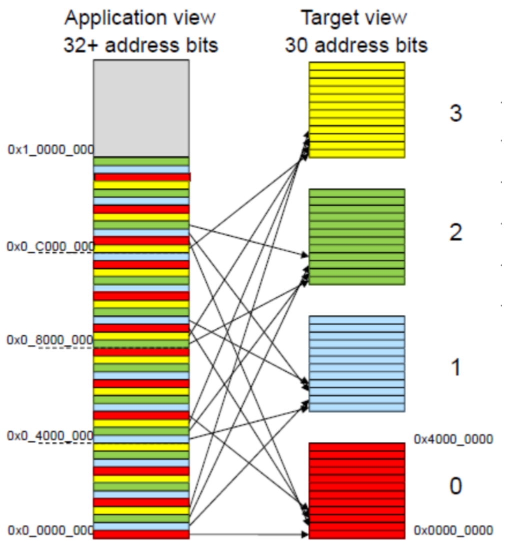

Fig. 1. Address interleaving in STNoC. Requests to addresses from an initiator change in turn, headed towards different memory controllers - called channels - each connected onto a different node. Each different color on the right side of the Figure corresponds to a different target node. On-chip networks can support this solution natively due to that packets from the same initiator can be routed via different paths.

controllers called channels. While a classical routing function associates an address to a routing through a memory address range, STNoC address interleaving is based on defining at configuration time which memory region will be part of an Interleaved Memory Region Set (IMRSet). A designer can create IMRSets by defining the list of targets participating into a set, the number of channels, the interleaving step for this IMRSet, and the starting and end Interleaved Memory Region (IMR) within a channel. A maximum of 16 IMRSets per STNoC instance can be configured, and each IMRSet can have up to 16 channels. If the incoming transaction address belongs to an IMRSet, the associated routing path is computed based on the current address, configured interleaving step, and number of channels for the given IMR. As a result of this computation, an appropriate routing path is selected, leading to one of the channels constituting the IMRSet. Ordering issues related to request transactions heading to the same destination are assumed to be resolved at IP-level. The case illustrated in Figure 1 corresponds to 1 IMR with 4 channels of $1GB$ each. The way address interleaving is implemented is by spreading the memory space into different memory nodes, each one having different addresses; each target node holds a different memory controller. The memory controllers, i.e. the channels, should be located on contiguous nodes. We should notice here that the number of channels and their memory map location are fixed parameters intrinsic to the system. Another solution available - but not studied in the present work - is that requests to addresses can change in turn towards different banks belonging to the same channel.

IV. Related Work

In order to improve performance in MPSoCs, it is common practice to employ interleaved access not only to different physical memories, but also to banks, ranks or DIMMs of a specific memory controller. This is especially important for memory bandwidth-intensive, vector computations, whereas a high degree of spatial and temporal locality provides a significant advantage for overlapping parallel memory access with computation [5]. Similar to interleaving, related skewed data allocation schemes support parallel (strided) array accesses to array rows, columns or diagonals, thus optimizing linear algebraic computations in multiprocessors with circuit-switched interconnection networks [6]. In addition, XOR address mapping principles randomize bank ordering access in order to improve cache performance [7], [8].

Since STNoC address interleaving is implemented at the initiator's network interface (INI), our study is very similar to [9]. In that study the authors model distributed interleaving at the INI to balance traffic in both the NoC and the memory channels of a WideIO DRAM memory. Experimental comparisons with $i)$ centralized solutions whereas a single multi-ported controller can access all channels of the WideIO memory, and $ii)$ independent distributed controllers assigned to each channel (without interleaving), are based on communication-only task subgraphs of a modem IP-core. Their results indicated that the proposed distributed solution at the initiator NI improves overall throughput, while minimally impacting the performance of latency sensitive communication flows. Unlike that study, we focus on optimizing interleaving performance by considering important NoC parameters such as link-width, and we do not examine the performance of different interleaving schemes.

Since 1970s, interleaved schemes have been integrated in different generations of commercial systems, such as IBM 360/85, Ultrasparc III, and Cray-Y MP. Nowadays, several commercial multicore SoCs support bank interleaving. For example, Intel regularly issues best practices for optimized DDR memory performance in systems integrating Xeon processors and servers that support interleaving. These practices require an even number of banks per channel, an even number of ranks per bank, an odd/even number of channels depending on memory size. Moreover, the number of ranks is shown to be related to the number of DIMMs and the operating frequency of the memory subsystem [10]. In addition, Arteris [11] and Sonics [12] integrate memory interleaving in their interconnect solutions, but they do not provide detailed experimental results. Sonics has released limited results only for their SonicsGN-3.0 NoC technology using two channels and off-chip DDR3 memories.

V. Experimental Setup

This Section overviews the setups and the environment we used to evaluate address interleaving. Our aim was to explore the parameters allowing to achieve high performance without consuming unnecessarily silicon resources and power. Tailoring wisely the system will result in a well-balanced setup, e.g. a 32 $Bytes$ link-width for the NoC might not be necessary if an 8 $Bytes$ link-width is sufficient. We varied the link-width, and kept fixed the number of VCs across our experiments, equal to 2. The FBA mechanism in the routers was inactive; a study of the FBA impact is available in [13] and further discussion on this is beyond the scope of the present work. The router's buffer size was set equal to the number of flits comprising a packet. On the other hand, we set large queues at the NIs in order to avoid potential overflows when new data arrive; although this is unrealistic, it prohibited packet dropping, and thus allowed us to find the time elapsed for serving a packet in the worst case and also assess saturation cases, by measuring the time a packet remains in the NI queue prior to releasing it into the STNoC. In a real system the NI queue is relatively small, and packet dropping could be addressed by either controlling the arrival rate of data entering the queue depending on its status, by employing backpressure flow control, or, with retransmission of dropped packets.

A. Evaluation Framework

We performed experiments in the framework published in [14] that allows measuring the end-to-end delivery times, and the power consumption of routers and links. It extends the gem5 simulation framework by connecting together initiator tiles such as CPUs or DMA engines, and target tiles such as memory controllers, using a cycle-accurate model of the commercial STMicroelectronics STNoC backbone interconnect. We measure two types of delay; the time a packet is waiting in the NI queue before it is split up into flits and released to the network (called NI queue delay), and then, after the head flit leaves the NI queue, the time elapsed for traversing the network until it reaches the destination. The latter is referred to as network traversal delay, and it is the time required for the flit to pass through all the routers in its path and reach the target NI. This way me can measure the time for transferring each flit and also the entire packet; we recall that once the head flit enters the on-chip network, successive flits of the packet are transmitted in a pipelined manner, cycle-per-cycle. Besides measuring the transfer delay for write and read requests, we also calculate the throughput. In specific, we report the average values of the transfer delay on the NoC at flit-level, and of the power consumption at the routers, and the total data transfer rate through the NoC, i.e. the aggregate throughput.

B. Application Mapping and Traffic Generation

In all setups we used a synthetic benchmark that generates traffic from all initiator tiles to perform write and/or read requests to the target memory nodes. A memory node is accessed depending on the transaction address, through a memory map. We perform address interleaving through our software code and proper memory allocation. In particular, the address mapping is carried out so as each request is headed towards a memory node depending on the interleaving step; the interleaving step refers to the pace with which an initiator changes destination. For instance, if $interleaving\ step = 1$, every new request from an initiator is forwarded to an address located in a different memory node from the previous request; in this way we assess NoC behaviour when each new packet sent from an initiator is following a different routing path from the previous packet in the sequence. If $interleaving\ step = 2$, two consecutive requests from an initiator are forwarded to the same memory node, the next two requests to a different memory node, and so on.

The address interleaving scheme is implemented in the initiator NIs, and thus it is their responsibility of splitting the transactions into packets and sending them to the proper channel according to the address. Further details on the traffic generation benchmark are unnecessary as we only used it in order to expose the NoC behaviour under marginal conditions in the setup with CPUs, and to study its behaviour in the setup with multiple DMA engines. In all our experiments the data size of packets is 128 $Bytes$. It is application-dependent and we selected this size based on the specifications of a healthcare application [15], which we will deploy in the future.

C. CPUs and DMAs as Initiators

We experimented with three different setups. The first setup has 8-nodes in which four CPUs and four memories are interconnected via a Spidergon STNoC. CPUs initiate memory requests, and we studied NoC behaviour for different packet injection rates. In particular, the purpose of these experiments was two-fold: first to explore whether interleaving scheme can remedy the NoC from saturation [16], and second to explore the effect from changing the interleaving step value. We should notice that interleaving between transactions takes place only when the current transaction targets a different memory address from the previous one; if it targets the same address, the destination channel is not changed. In this setup we didn't model the delay added by the memory access at the target NI; it is unnecessary in these experiments as we are primarily interested in studying NoC maximum serving capability only in order to find the conditions under which NoC saturates, and whether address interleaving can remedy saturation. In a recent study for different packet injection rates we exposed such extreme cases in a similar setup with four CPUs initiating requests to four memories [16]. We demonstrated that when all CPUs send random requests to all memories, and when link-width is 8 $Bytes$, saturation occurs above the rate of 0.035 $packets\ per\ clock\ cycle\ per\ CPU$. In that work we varied the injection rate only, without studying the effect of address interleaving, and we also didn't model the memory access delay.

The other two setups incorporate DMA engines to perform write and read memory requests, and CPUs do not produce traffic. Each setup interconnects a different number of nodes via the Spidergon STNoC, i.e. 16-nodes and 32-nodes setup. The first setup has 7 DMA engines, and the second setup has 23 DMA engines. The incorporation of multiple DMA engines accessing multiple memories has been previously studied and is close to the industry roadmap [4]. In both these setups we have modelled the DDR controller, thus our measurements include the memory access delay. A DMA engine generates a new request only once the previous one completes; completion of a request is signified at the initiator side with an ACK packet sent by the target. Therefore, the notion of injection rate does not apply here. Figure 2 illustrates the 16-nodes setup in which the memory space is distributed evenly across 8 channels,

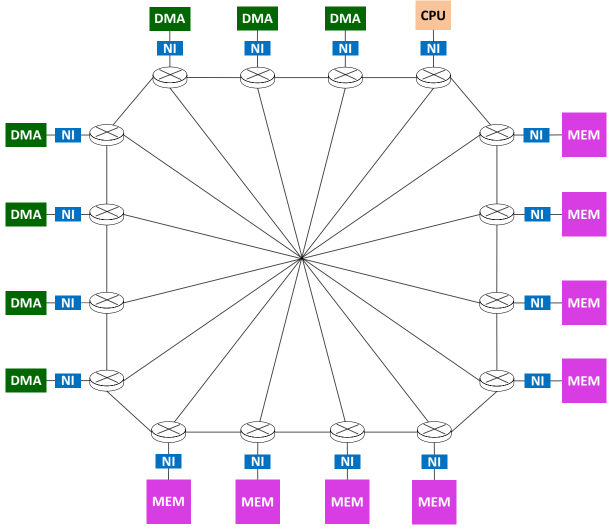

Fig. 2. 16-nodes Spidergon STNoC setup with 7 DMA engines. We depict only the case with 8 channels, but we conducted experiments with 1 channel (without address interleaving), 2, 4 and 8 channels. The channels, i.e. DDR controllers, are placed in contiguous nodes. Memory space is distributed evenly amongst the nodes.

each one located onto a different node. We experimented with various number of channels, ranging from 1 to 8, in both the 16-nodes and the 32-nodes setup. STNoC supports up to 16 channels, but this constitutes an extreme case and we didn't study it in the present work.

D. The DDR Memory Controller

In the two latter setups we instantiated one or multiple copies (number of copies depends on the number of channels in the experiment) of the default DDR2-800 memory controller module provided by the ruby simulation model in [17] of gem5 ver. 9378, Jan 2013. Its theoretical peak transfer rate is $6,400\ MB/sec$. It models a single channel, connected to 2 DIMMs, 2 ranks of DRAMs each, with 8 banks per rank. It assumes closed bank policy, i.e. each bank is closed automatically after a single read or write. It operates with minimum burst length, i.e. 4-bit-time transfer on the data wires, and assumes "posted CAS", i.e. the "READ" or "WRITE" is sent immediately after the "ACTIVATE".

VI. Experiments and Results

We first deployed the 8-nodes setup in which all CPUs send write-only requests to all memory nodes, and studied Spidergon STNoC behaviour, with and without address interleaving. Figure 3 shows the average NoC delay per flit for different number of channels, packet injection rates, and interleaving steps. We observe that the NoC delay is smaller when $interleaving\ step = 1$, i.e. each initiator changes destination channel after each transaction. This stands as long as the current memory address that is accessed is different from the previous one; if the current transaction concerns the same memory address with the previous transaction, the initiator does not change destination channel. It also appears that the benefit from increasing the number of channels is significant; the more the channels, the larger the maximum allowed packet injection rate, i.e. we obtain that for 4 channels, when injection rate is up to 0.07 $packets\ per\ clock\ cycle\ per\ CPU$, the NoC does not saturate. This tendency holds for all values of interleaving steps, but we observe that higher values result in diminishing returns; e.g. if $interleaving\ step = 32$, for 4 channels, saturation begins earlier, when packet injection rate is between $0.04-0.05$. Table I has the maximum packet injection rates without causing NoC saturation for different number of channels and $interleaving\ step = 1$. The improvement in the best case can be bigger than $2x$ as compared to the case without address interleaving.

TABLE I
Improvement in the maximum packet injection rate - per clock cycle per CPU - that Spidergon STNoC serves without being saturated, when enhanced with address interleaving. $interleaving\ step = 1$ and $link\ width = 16\ Bytes$.

# channels	max packet injection rate	improvement
w/o interleaving (reference)	0.03	-
2	0.05	1.66x
4	0.07	2.33x

We then proceeded with experiments in which the DMA engines perform write and read requests. We kept the interleaving step equal to 1, which turned out to be the most effective value in the previous setup, and varied the link-width and the number of channels. We repeated the experiments for different values of the DMA transfer length, between 16 and $128\ Bytes$, in powers of two. Figure 4 shows the aggregate throughput results, i.e. total transfer data rate across the device, for the 32-nodes setup in which 23 DMA engines initiate requests. We varied the number of channels between 1, i.e. case without address interleaving in which all requests are headed to one memory controller only, and 8, i.e. maximum number of channels. When channels are 2, 4 or 8, in a sequence of transactions from any DMA engine, each new transaction is sent to a different channel from the previous one. All cases in Figure 4 illustrate that the benefits from enhancing the NoC with address interleaving are higher for the narrowest link; for link-widths equal to $16\ or\ 32\ Bytes$ the gains are not that high. Table II shows this clearly by comparing the improvement between the different cases.

Table II also shows that improvement increases with the number of channels; we pointed this out previously as well in the study for the 8-nodes setup. The maximum improvement metric at the bottom of Table II refers to the comparison between the setup without address interleaving and the setup with 8 channels. On the other hand, improvement seems to saturate, which is more obvious when link-width is $32\ Bytes$. If this is not a "real" saturation coming from the performance limitations of the address interleaving scheme itself, it is due to a bottleneck, coming either by the rate at which DMA requests arrive at the initiator NI prior to releasing them into the NoC (the notion of injection rate is lacking here), or, by the way the DMA operates, or, due to the access delay of the specific DDR controllers. To identify the exact reason behind this occurrence, further experimentation is needed.

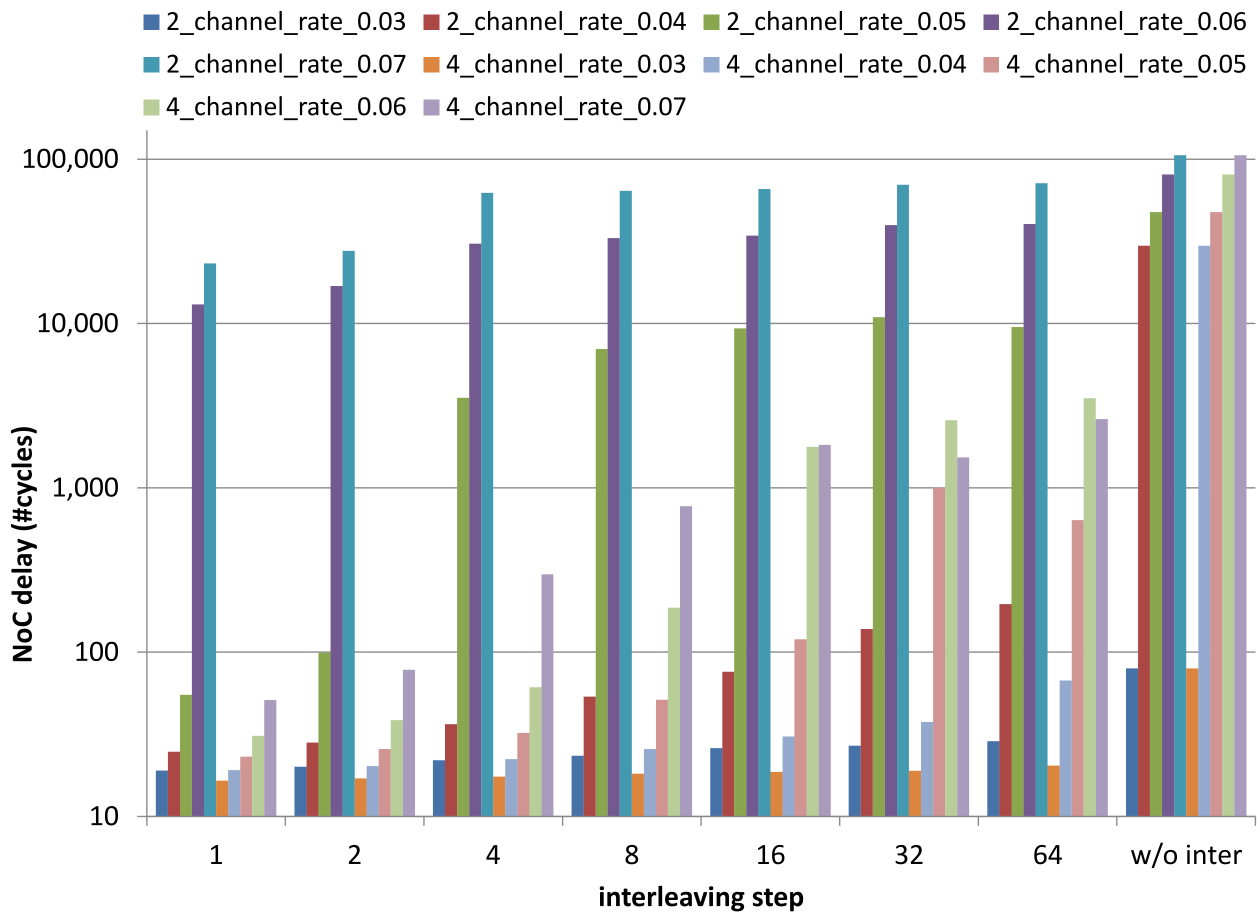

Fig. 3. Average transfer delay (in logarithmic scale) in the Spidergon STNoC for different interleaving steps, number of channels (2 or 4) and injection rates (0.03-0.07). Runs concern the 8-nodes setup, with 4 CPUs and 4 memory nodes, in which the CPUs are the initiators, and link-width is 16 Bytes.

One of the most valuable results in this experiment is that by enhancing a narrow STNoC with address interleaving, the achieved aggregate throughput is close to the aggregate throughput of STNoC when link-width is 32 $Bytes$. Figure 4 illustrates that this holds for all DMA transfer lengths.

We conclude that it is possible to build large-scale systems with 32-nodes around a NoC architecture with narrow link-width. This accounts for considerable savings in wires and routers, and for less power consumption. Table III has the power results for different link-widths and number of channels. It shows only the case in which the DMA transfer length is 128 $Bytes$, but the results and tendency are the same for the rest transfer lengths as well, i.e. $16, 32$, and 64 $Bytes$. We observe that router's total power is much higher when STNoC is configured with a wide link. This is mainly due to the router's clock power; as a generic rule, when link-width doubles, clock power also doubles. Thus moving from a link-width of 8 $Bytes$ to a link-width of 32 $Bytes$, this results in a $4x$ increase of the power consumption of router's clock. In [13] we are analyzing this by breaking down the power overhead. Moreover, Table III shows that for each different link-width,

TABLE II
IMPROVEMENT IN THE AGGREGATE THROUGHPUT WHEN EMPLOYING ADDRESS INTERLEAVING IN THE 32-NODES SETUP WITH THE 23 DMA ENGINES, BY VARYING THE LINK-WIDTH (LW) AND THE NUMBER OF CHANNELS. $interleaving\ step = 1$ AND $DMA\ transfer\ length = 128\ Bytes$.

	aggregate throughput (MB/sec)		
# channels	lw=8 B	lw=16 B	lw=32 B
w/o interleaving (reference)	10,710.95	15,119.96	15,960.79
2	16,181.08	18,306.28	18,882.89
4	17,676.93	18,850.63	19,264.97
8	18,170.43	19,056.23	19,280.95
max improvement	69.64%	26.04%	20.8%

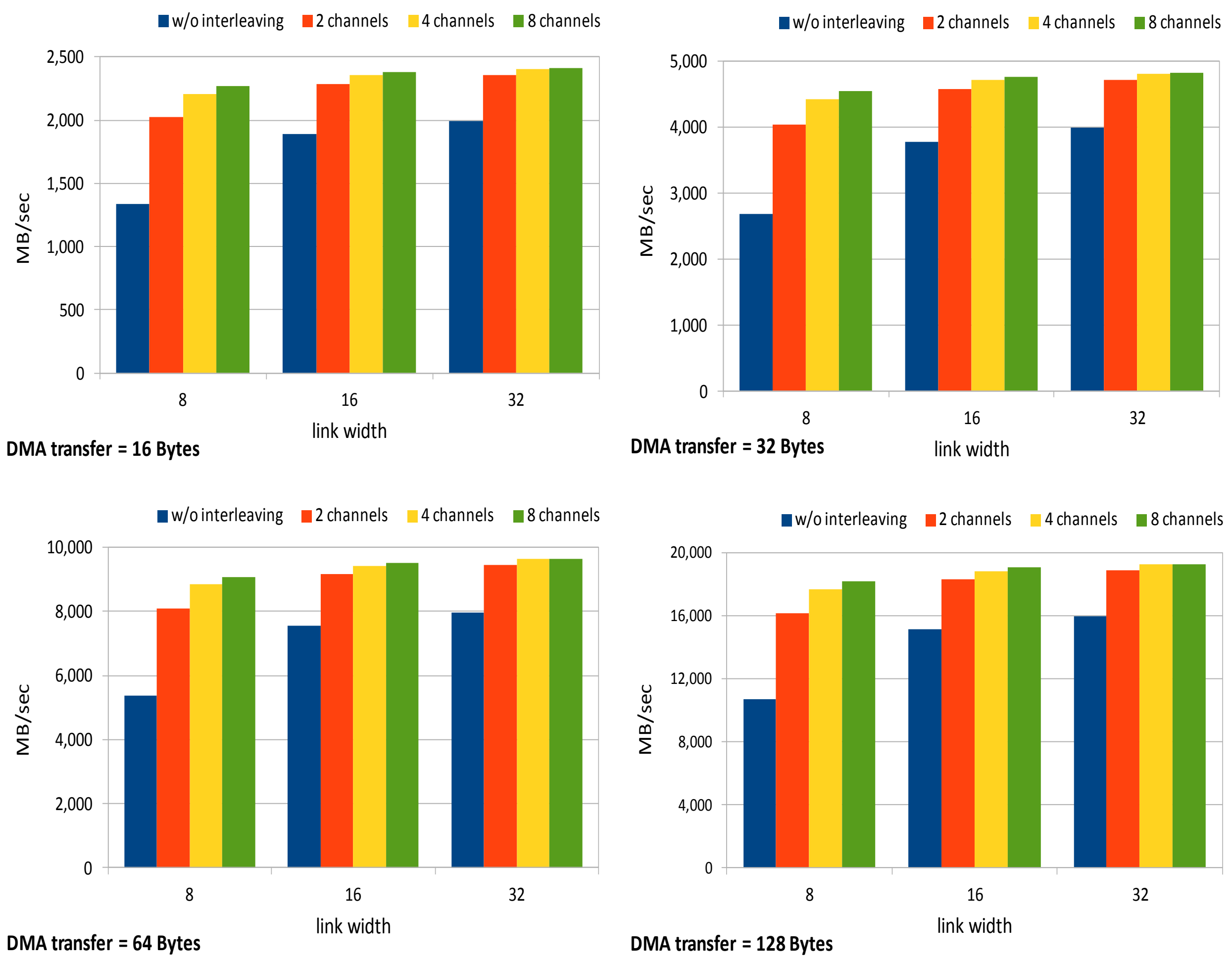

Fig. 4. Aggregate throughput (MB/sec) in the 32-nodes Spidergon STNoC setup with 23 DMA engines, with and without address interleaving. We conducted experiments with up to 8 channels, for various link-widths and DMA transfer lengths.

the power consumption increases slightly when the number of channels increases. This is indicated by the maximum power overhead added metric at the bottom of the Table, which refers to the comparison between the setup without address interleaving and the setup with 8 channels. It is worth noting that the biggest increase in the power consumption appears when link-width is $8\ Bytes$.

TABLE III
EFFECT FROM THE VARIATION OF LINK-WIDTH (LW) AND THE NUMBER OF CHANNELS ON THE TOTAL POWER CONSUMPTION OF ROUTERS. VALUES ARE AVERAGE AND CONCERN THE 32-NODES SETUP WITH THE 23 DMA ENGINES. $interleaving\ step = 1$ AND $DMA\ transfer\ length = 128 Bytes$.

# channels	router power (watt)		
	lw=8 B	lw=16 B	lw=32 B
w/o interleaving (reference)	1.194	1.721	2.759
2	1.274	1.769	2.823
4	1.307	1.786	2.845
8	1.321	1.798	2.856
max power overhead added	10.63%	4.47%	3.51%

We conducted the same experiments for the 16-nodes setup with 7 DMA engines, but the benefits from enhancing the initiators' NIs with address interleaving were small. We varied again the link-width, the number of channels, and the DMA transfer length, but the improvement in the best case was less than 2%. This holds for the specific setup, i.e. DMA engines and selected DDR controllers, and cannot be attributed solely to the address interleaving performance; further experiments are needed to reveal the exact reason behind this result.

VII. IMPLEMENTATION COST OF ADDRESS INTERLEAVING

Implementation of STNoC address interleaving is based on a non-disruptive modification of the STNoC routing function, and is supported via the iNoC EDA configuration tool. We implemented it by extending the address decoding mechanism of the initiator's network interface (INI). In particular, we enhanced the address comparators with an additional output signal indicating whether the incoming address belongs to an interleaved memory region (IMR). During operation, if the incoming address doesn't belong to an IMR, the routing and QoS setting corresponding to this address range are transferred to the network layer header encoder, as normal. If the address belongs to an active IMR, interleaving function is triggered:

- reference routing for the destination channel is retrieved from preconfigured - optionally reconfigurable - routing tables;
- interleaving constants, i.e. channel offset, interleaving step and number of channels, and QoS settings for the current IMR are retrieved from preconfigured tables and passed to the network interface encoder; these tables have been set using the iNoC tool; and
- interleaving algorithm is executed to identify the new destination channel.

In terms of RTL cost, implementation of address interleaving typical configuration adds an overhead of 5% more gates in the INI address decoding hardware when synthesized in $28nm$ FSDOI ST technology. It reuses the already in place routing function, and for concurrent solutions, i.e. multiple parallel accesses with different IMRSets to the memory controllers, it also requires de-interleaving support at the target's network interface (TNI). STNoC address interleaving relies on routing packets to alternative destinations using different routing paths, and the address signals are never modified in the process. Optionally, address interleaving can be reprogrammed at runtime to enable or disable a preconfigured setting. This is achieved by writing a set of new registers located at the INI, and an external STNoC driver which accesses them via the STNoC NI programming port. To guarantee memory coherency and data integrity, all INIs implied in an IMRSet must be reprogrammed in a coherent way via System Coherent Reprogramming (SCR). This allows for freezing data traffic, reprogramming several INIs concurrently, and then restarting data traffic while preserving data integrity and memory coherency.

VIII. Discussion

We have analyzed the benefits from enhancing the Spidergon STNoC with the address interleaving scheme. We showed that the smaller the interleaving step, and the more the number of channels, the higher the improvement. We also showed that the gains from enhancing the NI with address interleaving are higher when link-width is $8\ Bytes$, and the aggregate throughput in this case is close to the throughput when link-width is $32\ Bytes$. Results concern experiments with a DDR2-800 memory controller and we plan to extend them using a faster memory controller, e.g. recent models of low power DDR controllers such as LPDDR3 or LPDDR4. To gather the results we analyzed three different setups, and we saw that address interleaving might not be promising in all setups with the DMA engines. In the 32-nodes setup the results were outstanding, thus configuring the NoC with an $8\ Bytes$ link-width allows for significant savings in wires, router's size, and power consumption. Moreover, the setup with the CPUs acting as initiators showed that interleaving relieves considerably the NoC from saturation.

Our gem5 STNoC simulation environment and related methodology can be used for high-level configuration of address interleaving parameters. For instance, in use cases with specific, well-defined communication patterns, our framework can be used to explore and auto-generate different address interleaving features, such as the number of interleaved memory regions (IMR), the default IMR start/end addresses in the memory map, the offset, number of channels and default step value for each IMR. In this respect, our framework can extend the STNoC configuration toolsets, i.e. iNoC and Meta-NoC. More specifically, it can interface with existing IP mapping and partitioning tools able to configure the connectivity memory map of IPs to STNoC network nodes and define the corresponding routing paths and NI look-up tables for embedded applications with well-defined access patterns.

Acknowledgment

This research was partially funded by the European Union's Seventh Framework Programme FP7/2007-2013 under grant agreement No. 610640, DREAMS project.

References

[1] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra, "Spidergon: A Novel On-chip Communication Network," in *4th International Symposium on System-on-Chip (ISSOC)*, November 2004, pp. 15–26.

[2] C. Janac and M. Tang, "White Paper from Arteris: NoC Interconnect Fabric IP Improves Power, Performance and Area," Tech. Rep., Jan 2016.

[3] *Cannes Ultra - ARM-based, UHD Multimedia Connected Client-Box Platform*, STMicroelectronics, Dec 2013.

[4] P. S. Paolucci, F. L. Cicero, A. Lonardo, M. Perra, D. Rossetti, C. Sidore, P. Vicini, M. Coppola, L. Raffo, G. Mereu, F. Palumbo, L. Fanucci, S. Saponara, and F. Vitullo, "Introduction to the Tiled HW Architecture of SHAPES," in *Design, Automation and Test in Europe (DATE)*, April 2007.

[5] G. S. Sohi, "High-bandwidth Interleaved Memories for Vector Processors - A Simulation Study," *IEEE Transactions on Computers*, pp. 34–44, 1993.

[6] D. H. Lawrie, "Access and Alignment of Data in an Array Processor," *IEEE Transactions on Computers*, vol. 6, no. 3, pp. 1145–1155, 1975.

[7] W. Frailong, J. M. Jalby, and J. Lenfant, "XOR-schemes: a Flexible Data Organization in Parallel Memories," in *International Conference on Parallel Processing*, 1985, pp. 276–283.

[8] W.-F. Lin, S. Reinhardt, and D. Burger, "Reducing DRAM Latencies with Integrated Memory Hierarchy Design," in *International Symposium on High-Performance Computer Architecture (HPCA)*, January 2001, pp. 301–312.

[9] C. Seiculescu, L. Benini, and G. D. Micheli, "A Distributed Interleaving Scheme for Efficient Access to WideIO DRAM Memory," in *International Conference on Hardware/Software Codesign and System Synthesis CODES+ISSS*, October 2012, pp. 103–112.

[10] Lenovo, "Optimizing Memory Performance of Lenovo Servers based on Intel Xeon E7v3 Processor," Tech. Rep., 2016.

[11] http://www.arteris.com.

[12] https://www.semiwiki.com/forum/content/4742-sonics-new-noc.html.

[13] P. Petrakis, M. Abuteir, M. D. Grammatikakis, K. Papadimitriou, R. Obermaisser, Z. Owda, A. Papagrigoriou, M. Soulie, and M. Coppola, "On-chip Networks for Mixed-Criticality Systems," in *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, July 2016.

[14] M. D. Grammatikakis, K. Papadimitriou, P. Petrakis, A. Papagrigoriou, G. Kornaros, I. Christoforakis, O. Tomoutzoglou, G. Tsamis, and M. Coppola, "Security in MPSoCs: A NoC Firewall and an Evaluation Framework," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, pp. 1344–1357, August 2015.

[15] Z. Owda, M. Abuteir, R. Obermaisser, and H. Dakheel, "Predictable and Reliable Time Triggered Platform for Ambient Assisted Living," in *IEEE International Symposium on Medical Information and Communication Technology (ISMICT)*, April 2014, pp. 1–5.

[16] K. Papadimitriou, P. Petrakis, M. D. Grammatikakis, and M. Coppola, "Security Enhancements for Building Saturation-free, Low-power NoC-based MPSoCs," in *IEEE International Workshop on Security and Privacy in Cybermatics (SPiCy), co-located with the IEEE International Conference on Communications and Network Security (CNS)*, September 2015, pp. 594–600.

[17] http://www.m5sim.org/Ruby.