MicroLET: A new SDNoC-based communication protocol for chipLET-based systems

Soultana Ellinidou, Gaurav Sharma, Olivier Markowitch, Sotirios Kontogiannis, Jean-Michel Dricot, Guy Gogniat

MicroLET: A new SDNoC-based communication protocol for chipLET-based systems

Soultana Ellinidou*, Gaurav Sharma*, Olivier Markowitch*, Sotirios Kontogiannis †, Jean-Michel Dricot* and Guy Gogniat‡
*Cybersecurity Research Center, Université Libre de Bruxelles, Brussels, Belgium
†Laboratory team of Distributed Microcomputer systems, Department of Mathematics, University of Ioannina, Ioannina, Greece
†Lab-STICC, Université de Bretagne Sud, Lorient, France
Email:{soultana.ellinidou, gsharma, olivier.markowitch, jdricot}@ulb.ac.be, skontog@cc.uoi.gr, guy.gogniat@univ-ubs.fr

***Abstract*—Currently the industry moves to smaller process nodes even if the cost for yielding large dies continues to increase, moving to the 5nm and even 3nm nodes. Hence a chiplet-based design has been initiated and quickly gain attention from industry, academia and government agencies. This cutting edge approach became advantageous to break down a large die into smaller chiplets in order to improve yield and binning. In order to exploit this new approach the interconnect fabric connecting the nodes of the entire system should be of high importance to enable the properly distribution of the data. Each individual chiplet may contain its own local Network on Chip (NoC), which operates for intra-chiplet traffic. However the communication over chiplet-based systems is complicated enough, due to various routing algorithms and NoC topologies and an alternative solution is needed. In this paper we introduce an SDNoC(Software Define Network on Chip)-based communication protocol for chiplet-based systems, called MicroLET, which consists of a flexible and modular SDNoC architecture and 3 main phases: Handshake, Network Monitoring, Routing. An implementation of the SDNoC architecture and an evaluation of the proposed routing algorithm compared to the XY and the Odd-Even algorithms within different traffic scenarios is presented. Through the evaluation of the MicroLET protocol, it is proven that it could be a good candidate for the future chiplet-based systems.**

***Index Terms*—SDNoC, NoC, Chiplet Systems, Communication Protocol**

I. Introduction

Since fifty years, the number of transistors that was able to fit into a single piece of silicon increased on a predictable way known as Moore's law [16]. This had as a result the digital evolution of minicomputers to PCs, afterwards to smartphones and to cloud, by placing more and more transistors into each generation of their microchip and simultaneously making them more powerful and able to support the dynamic nature of today's applications (for example in automotives and avionics). However under the umbrella of Internet of Things (IoT) and Internet of Everything (IoE) a big variety of applications pop up, in order to satisfy the people's needs in transportation, health-care, manufacturing, and energy management with diverse requirements, which traditional SoCs are not always capable to support due to the cost of semiconductor processing and fabrication and the complexity in terms of the amount of circuit elements for a large die. At the same time the smallest features of transistors reached 7nm [25] and IMEC manufactured first 3nm transistor [6]. Furthermore a huge increase in Integrated Circuits (IC) cost is observed.

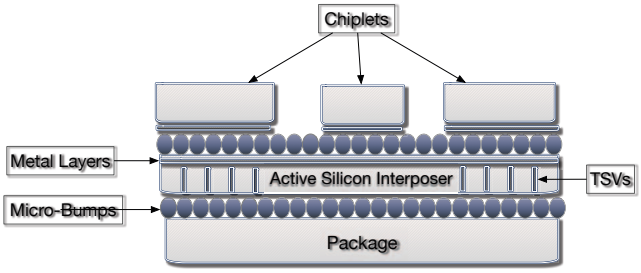

Figure 1: chiplet based system

Hence the chipmakers start to look for alternate ways. The current top-notch approach, which the industry is investigating is the "chiplets" on a substrate to reduce the cost of complex semiconductor solutions, since the fabrication of large monolithic dies will become more costly.

Chiplets came into the surface to break a conventional monolithic SoC into smaller pieces. More precisely Chiplets refer to the independent constituents which make up a large chip which consists of multiple smaller dies. The need to go with multiple chips comes from reticle limit which dictated the maximum size of chip possible to be fabricated. Designs that exceeded the reticle limit had to be split up into smaller dies. The idea is that individual CPUs, memory, and other processing elements will be able to be mounted onto a relatively large slice of silicon, called an active interposer, which is thick with interconnects and routing circuits. Recently the chiplet approach has gain attention from academia [12, 13], industry [23, 24] but also from government agency [21].

A chiplet-based system is depicted in Figure 1. It consists of chiplets which are placed on the interposer, routing inter-chiplet connections through metal layers in the interposer, and placing Through Silicon Vias (TSV) in the interposer to connect chiplet microbumps to package-level interconnect. TSV in the bottom die provides external I/O access and power delivery to the top die.

As far as the industry is concerned, AMD used chiplet approach with a server processor [14], made with four chip-

978-1-7281-2862-7/19/$31.00 ©2019 IEEE
DOI 10.1109/DSD.2019.00019

lets. Epyc is a brand of x86-64 microprocessors designed and marketed by AMD based on the company's Zen micro-architecture specifically targeted for server and embedded system markets. Intel emerged with a processor for mobile PCs that combines an Intel CPU with a custom-designed graphics module from AMD (AMD Radeon Graphics technology) by using the chiplet aproach [9]. Marvell announced their first products based on its MOdular CHIp (MoChi) architecture, which is a chiplet-based approach [10].

However the design and validation of chiplets have already been well explored, the interconnect fabric connecting the nodes of the entire system must be equally explored in order to enable the properly distribution of the data within the system. Each individual chiplet may contain its own local Network on Chip (NoC), which operates for intra-chiplet traffic and different hierarchical layers of communication should be introduced.

While current multi-chiplet architectures utilize passive integration technologies such as silicon interposers, in this research we took into account the chiplet-based SoCs, which are based on active silicon interposer (Figure 1). Despite the high interest into the passive substrates [17], there is available research in academia [13], industry and government [4] focused on active interposers. The active interposer implements its own NoC in order to interconnect the chiplets. While connecting several NoCs together, they can introduce new resource cycles that cause cyclic dependencies across the chiplets.

The classical NoC interconnect was introduced on 2002 by Benini et al [2]. More precisely the authors present a unification of on-chip communication solutions, which consists of an on-chip packet or circuit switched micro-network of interconnects, called NoC. Processing Elements (PE) access the network by means of proper interfaces, and have their packets forwarded to destination through a multi-hop routing path. The scalable and modular nature of NoCs and their support for efficient on-chip communication potentially leads to NoC-based multi-processor systems characterized by high structural complexity and functional diversity.

However the complexity of the current NoCs motivates the researches to explore some alternatives of it [8, 3]. One NoC alternative that gains attention the last years is the SDNoC (Software Define Network on Chip). SDN emerged to support the dynamic nature of future network functions and intelligent applications while lowering operating costs through simplified hardware, software and management. The approach proposed by the SDN paradigm is that the data travels across multiple network entities (switches or routers) and efficient and effective data transfer is supported by a centralized controller. SoC architectures may adopt the SDN paradigm due to its advantages: reduced hardware complexity, high re-usability, and flexible management of communication policies. However, the challenge to apply the SDN may be the overhead for defining the paths in software against hardware-based approaches. Also the controller can implement different communication rules to define the paths, as Quality-of-Service (QoS), fault-tolerance, and security.

The work of this paper is mainly oriented towards the design and evaluation of a novel communication SDNoC Protocol standard called MicroLET but also towards the SDNoC integration within chiplet-based systems. The main contribution covers the networking aspects of Network on Chip for intra-chiplet communication which is inspired by large scale networks and the proposal of a new routing approach. The Microlet Protocol consist of 3 main phase: 1) Handshake Phase 2) Network Monitoring Phase 3) Routing Phase, which are detailed explained afterwards.

II. Related work

To the best of our knowledge, there is no existing literature for SDNoC-based Chiplet Systems, however there is limited literature available in SDNoC-based MPSoC (Multi Processor System on Chip) which includes SDN as a packet routing approach. SDNoC is a NoC communication paradigm rather than a specific design and implementation, presented first time in 2014 by Cong et al. [8], where the authors propose the SDNoC architecture where the control plane is deployed as a distributed unity at each router, however this is contrary to SDN philosophy because both planes are placed inside the router. Afterwards, the authors in [18] applied SDN principles in order to propose a SDNoC architecture. This architecture is focused on abstraction layers and interfaces that permit its deployment in a modular fashion and it has the potential to overcome the NoC management problems in the Many-Core era. Another interesting contribution of the same authors is presented in [19], where they evaluate the SDNoC architecture among the Processing Elements (PEs) in a Many-Core system with System C simulator, focusing on the configuration time, delay, and throughput of their architecture. Scionti et al. [20] use the SDN architecture in order to explore dynamic changes in the network topology, each PE has specific instructions to control the network topology by software, including switch off the links which are not used. In 2017, Berestizshevsky et al. [3], presented a novel NoC architecture, called SDNoC, based on a hybrid hardware-software approach. Their approach implements a software-based centralized Network Manager (NM), executed on a dedicated core. The NM allocates the route and switches forward the packets without storing them. Also, the switches do not maintain any routing table. Ellinidou et al. [11] proposed an SDNoC framework for Cloud-of-chip which consists of interconnected ICs and IC cores with different communication speeds and hierarchy levels, by using SDN they provide cloud-like flexibility within the system and they proposed a security protocol for the registration and authentication of every entity in the network. Later the same authors [22] proposed a new protocol in order to secure the communication and efficiently manage the routing within the Cloud-of-Chip, called SSPSoC. The SSPSoC protocol includes a private key derivation phase, a group key agreement (GKA) phase, and a data exchange phase in order to ensure that basic security primitives are preserved and provide secure communication and manage the IC to IC communication.

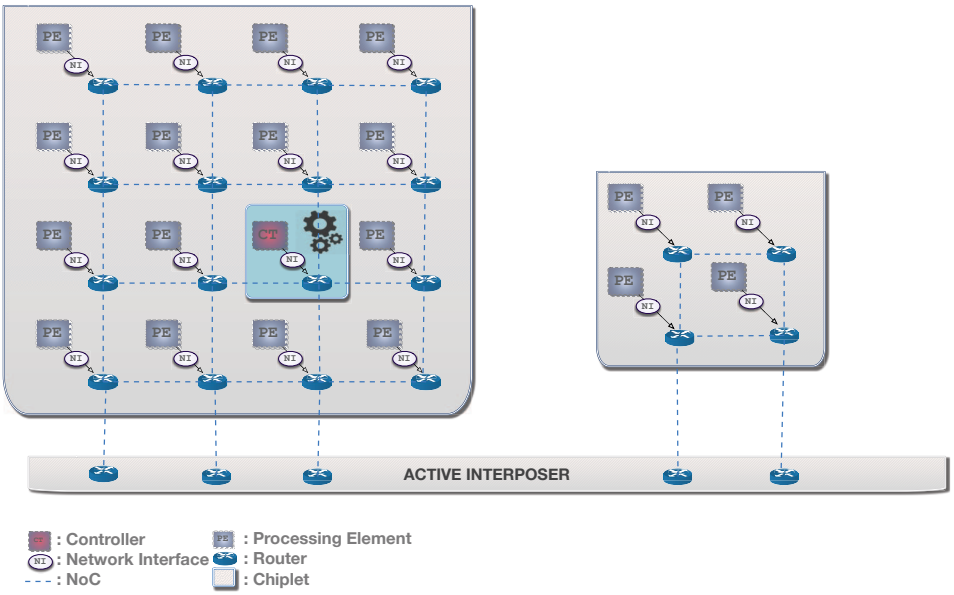

Figure 2: SDNoC architecture within a chiplet

III. SDNoC integration within chiplet-based systems

The typical intra-chiplet data communication is managed by Network-on-Chip (NoC), which supports regular interconnected topologies. However, in order to manage routing inside a chiplet with multiple cores, the size of routing tables will be large enough not to be accommodated on ordinary NoC routers. The memory overhead for routing tables will grow by $n^2 * k$ units where n^2 is the number of PEs on each chiplet and k is the number of chiplet on each Package. Therefore, to achieve secure inter-chiplet and intra-chiplet communication on a package, we do believe that some techniques based on SDN paradigm should be designed. The SDN concept came into the micro-scale networks recently, as it is presented in the previous section, and it is still limited under research. The traditional routing mechanism employs NoC hardware routers to manage the routes among chiplets. However, recent SDN based strategy implements controller with global view, which controls the routing in an adaptive manner. The proposed SDNoC network for chiplet-based system is depicted in Figure 2. Since in this work the main focus is to cover the intra-chiplet communication and leave the inter-chiplet for future work, the controller will be placed inside a chiplet and attached to one router, the rest of the routers within the network will communicate in order to ask for a possible route for the upcoming packets, with this way the SDN approach is enabled. Regarding the inter-chiplet communication an extra NoC is placed on the interposer and it is able to efficiently interconnect them.

OpenFlow (OF) is a common communication protocol used in SDN [15]. OF establishes a unicast communication channel between each individual router and the controller. It allows the controller to discover routers, create rules for the switching hardware and also collects statistics. Since OF is layer 4 protocol, designed for large scale networks, it will not be adaptable in micro-scale networks due to the vast number of network messages and rules that it contains. Hence a new communication protocol should be designed in order to fulfill the needs of micro-scale networks.

IV. MicroLET Architecture

The main entities of an SDNoC are: Network Interfaces (NI), Physical Links (PL), Routers (R), Processing Elements (PE) and the Controller (CT). The routers are linked to every Processing Element which could be a memory, a core or a processor and interconnect them through physical links. The NI is the intermediate entity between PE and router. In order to be more specific the packets are traveling between different nodes of the network which are routers and the packet routing is managed by a centralized controller, which is running as a process in a given PE. (Figure 2).

The Network Entities are explained below:

- **Routers**: Figure 3 illustrates the architecture of a SDNoC 5-port router employing virtual channel flow control and SDN based switching. The five ports correspond to the four cardinal directions and the local direction which connects the router with the PE through NI. The router consists of four components: the Flow Tables, the Arbiter, the buffers and the crossbar. It employs a pipelined design with speculative path selection to improve performance. The SDNoC router consists of a two-stage, pipelined architecture. The first stage is responsible for routing, where the router checks the flow tables and if there is

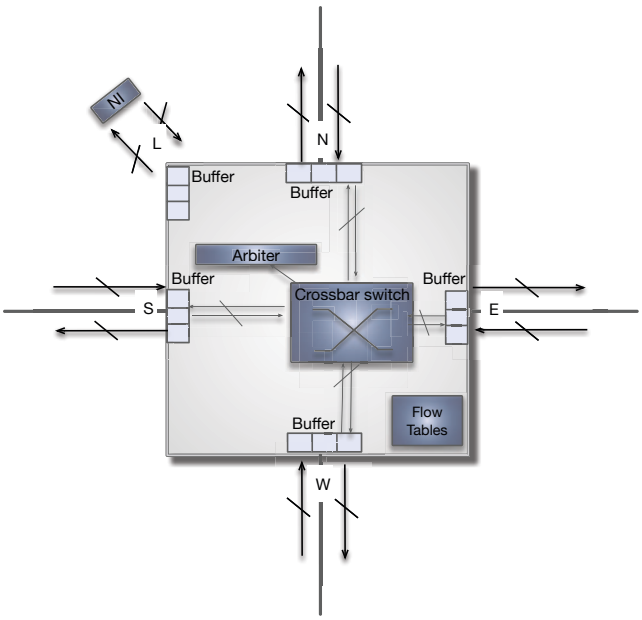

Figure 3: Router architecture

any flow rule it will send a request to controller to ask a new route. The second stage is responsible for crossbar traversal. In this work, the functionality of the router is described with respect to a 2D mesh interconnect.

- **Controller:** The SDN controller consists of a series of functions for sending packets for PE and router configuration, stop and start computation in a per flow basis. The SDN Controller can have partial network view. Specifically it provides the following services: sends configuration to a specific set of nodes in the network; collects state and statistics' data from a specific set of nodes in the network and generate a global or partial view (state) of the network. This software-based control enables to reduce the hardware complexity, moving to the software the decision to establish the network paths.
- **Physical Links**: The communication between controller and routers is managed by dedicated links, which interconnects them. Through the physical links the controller transfers control messages related to routing decisions to the routers and monitors the data network state.
- **The network Interface** is composed by two FIFO memories, one logic block to interface with the network, called "Router Adapter", and a logic block to interface with the processing unit (or core), called "Core Adapter". The Router Adapter is a logic block that interacts with the network dealing with the signals of physical channel and integrates the data that come from the network, to be delivered to the core. The Core Adapter also is a logic block that is connected with the core, and prepares the data that come from the core, to be written in the network, concatenating the fields Control bit, Origin Address and Destination Address to the Data field.

V. MicroLET Protocol

A. Packet format

Processing cores exchange data amongst themselves by sending packets across the NoC consequently through routers. Furthermore a router sends packets to controller but also to the other routers by using the data link layer. The packets are divided into a sequence of fixed-length flits, which is composed of a header flit, body flits, and a tail flit. The packet format of the SDNoC is illustrated in (Figure 4), more specifically it includes 8 fields:

- **TYPE:** indicates the type of the messages and the different type fields are shown in Table I.
- **SRC:** consists of the source ID.
- **DST:** consists of the destination ID.
- **NEXT_HOP:** consists of the next hop ID.
- **PRIO:** contains the priority of the packet, which can be high or low in order to be pipelined accordingly.
- **PAYLOAD:** contains the real data
- **TS:** is the timestamp and represents the send time
- **CRC:** represents the Cyclic Redundancy Check, which is the error-detecting code field.

B. Routing Operations

1) Network Messages: The network messages are flowing between the network entities through physical links. The different types of messages, which are integrated in order to fit in the packet format, are illustrated in Table I. The communication protocol includes 8 types of messages with different content. The `HELLO` message is designed for the handshake phase and the `ROUTE_REQUEST`, `ROUTE_REPLY`, `FLOW_UPDATE`, `NET_REQUEST`, `NET_REPLY` are designed for the network monitor and routing phase. Furthermore it is important to be mentioned that every `ROUTE_REPLY`, `FLOW_UPDATE`, `NET_REPLY` message should be acknowledged by an `ACK` message, otherwise it should be retransmitted.

2) Communication Protocol Phases: The MicroLET communication Protocol consists of 3 main phases:

1. **Handshake Phase:** During the Handshake Phase a HELLO messages are exchanged between the participants. Furthermore the controller is able to be aware about how many routers are in the network and also to be aware about there ID's.
2. **Network Monitoring Phase:** In order to move to Network Monitor phase, the Handshake phase should take place beforehand. The controller requests to be informed about network state by periodically sending a `NET_REQ` message to the routers. The receiver router should reply with a `NET_REPLY` message, which includes the current flits passing by every port. Each router has a counter in the buffer of every port and it is increasing according to the flits that are coming from this port in a given period. Therefore each routers monitors the flits that are inserted through North, East, South, West, Local ports during an interval time and forms the `NET_REPLY` message. As

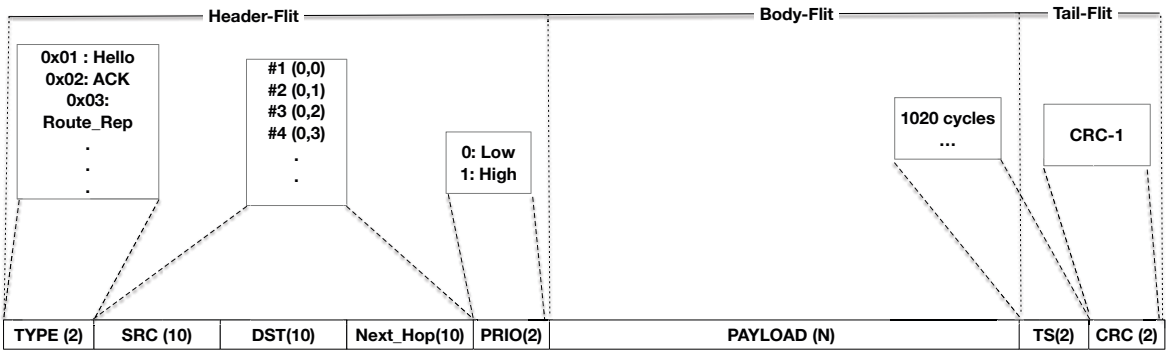

Figure 4: Packet format

soon as the controller receives a NET_REPLY, it should send back to routers an acknowledge and it updates its parameters that would be needed for the next phase. With this process the controller is managing to have a network state view, which is key element for the selection function of the routing phase.

3. **Routing Phase:** When the controller receives a ROUTE_REQ message from a source router, it extracts the PACKET_ID and the SRC and DST addresses from the upcoming flits which will be the input of the routing algorithm function. Afterwards, based on source and destination the routing algorithm outputs a set of admissible routes. Therefore, the routing algorithm has two main functionalities: the computation of the admissible routes and the selection of a route among the admissible routes. In order to compute the admissible routes sets, the proposed routing algorithm relies on a turn model routing algorithm. These algorithms have the advantages to be lightweight and deadlock-free. Among the existing turn model routing algorithms, Odd-Even (OE) [7] is used since it tends to provide better performance and higher adaptiveness than the others. Finally, once the controller has computed a set of admissible routes using the OE routing algorithm, it applies the selection function on the set in order to get the best possible route and forms the ROUTE_REPLY message.

3) Odd-Even Routing: The OE routing separates the columns of the mesh architecture as odd or even. The first column is even, the second column is odd. The admissible routes have to obey the two following rules:

Rule 1: In an even column, a turn from the east to the north or the south is forbidden.

Rule 2: In an odd column, a turn to the west is forbidden.

These two rules ensure the deadlock-freedom of the Odd-Even routing algorithm. In Figure 5, the blue lines indicate the valid turns and the red lines indicate the non-valid turns.

4) Selection Function: The selection function has a set of routes and the network state as inputs and outputs the optimal route from the set. In order to determine which route is the optimal one, the first step is to define metrics that assess the routes. The proposed selection algorithm that is implemented within the controller takes into account the link load and the router load. The load of a link (l_i) corresponds to the number of flits per second that flow through the link. The router load (r_{ij}) is the number of flits per second arriving towards the router. When the selection process takes place, the controller is responsible to run an algorithm in order to select the best route from an admissible set of routes. For this reason the two aforementioned formulas were designed in order to avoid the highly-loaded links and routers within the route.

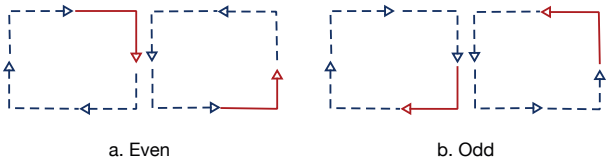

Figure 5: Odd-Even Routing

Highly-loaded links are affecting the bandwidth and their corresponding input buffers are likely to be full. Therefore, sending packets towards highly-loaded links will imply a considerable period of latency for the incoming packets since they will have to wait for the release of the links and the corresponding input buffers before accessing them. On the other hand, the more a router is loaded, the more time it will take to process incoming packets since it has first to process the already present packets. By avoiding the highly-loaded links and routers, the selection function (SL_{sum}) aims to balance the traffic as much as possible across the data network and therefore, avoids the formation of congested network areas. In order to determine the best route among an admissible routes set, these metrics have to be used to evaluate the routes. In that case, the controller computes a score (S) for each route among the set using a combination of the aforementioned metrics. With the proposed selection function, the route scores are computed by summing the load on the links and the routers along the routes. This score computation is shown in (1).

Table I: Designed Network messages

Type	Type Value	Description	Contents
HELLO	0x01	Sent by router to controller and vice vera or by router to router for the handshake process	HELLO
ACK	0x02	Sent by router to controller or by transmitter router to receiver router in order to ack the request.	ACK
ROUTE_REQ	0x03	Sent by router to controller which asks a route for a packet	Packet ID
ROUTE_REPLY	0x04	Sent by controller as an answer to Route˙Req	Packet ID, Route
FLOW_UPDATE	0x05	Sent by controller to routers in order to update the outport of a packet.	Packet ID, Route
NET_REQ	0x06	Sent by controller to routers which asks informations for network.	NET˙REQ
NET_REPLY	0x07	Sent by routers as an answer to Network˙State˙Req.	N= #, S=#, E=#, W=#, L=#
DATA	0x8	Contains the data	PAYLOAD

$$SL_{\text{sum}} = \sum_{i=0}^{L_f} l_i + \sum_{i=0}^{S_f} \sum_{j=0}^{S_f} r_{ij}. \quad (1)$$

Where L_f is the number of the sets of the link load values along the route and S_f the number of the sets of the router load values along the route. The controller is aware of the load of the links from the network monitoring process, and the load of a router is inferred from the load on the links arriving towards the routers as shown in (2).

$$r_{ij} = \sum_{i=1} \frac{l_i}{L}. \quad (2)$$

Where r_{ij} is computed as the average load on the links arriving towards the router so that the router load and the link load stay in the same order of magnitude and L is the number of the router links. Thereby, the route score is equally affected by the load on the links and on the routers.

At the end, the controller computes the S for each route within the set according to the SL_{sum} and chooses the route with the lowest score. In the case of multiple routes having the same S, a random choice is made.

VI. Evaluation

In order to evaluate the performance of the MicroLET protocol, simulations were performed with the Garnet2.0 [1], which is an NoC model implementation within the gem5 simulator [5]. The traffic generated by the processing cores according to the traffic injection rate (tir) is used, which is the average number of packets injected by the cores into the network per clock cycle ($0 < tir \leq 1$). Each core generates packets following a Bernoulli distribution with mean tir. In other words, each processing core will indeed generate a packet each $1/tir$ clock cycles on average, but the actual time at which the packets are transmitted is random.

Concerning the network monitoring phase, the NET_REQUEST and NET_REPLY messages were modeled as 1-flit packets. Nonetheless, they do not contain the content discussed before, because Garnet2.0 does not support the modulation of real payload within the exchanged packets. Moreover, to measure the link load, each router has a counter for each of its input channel. Each time a flit reaches an input channel, the corresponding counter is incremented. When the controller receives a NET_REPLY message from a router, it reads the value of the counters, divides it by to get the link load and stores it within the corresponding $N \times N$ matrix.

By assuming that the first phase is already available, the second and the third phase of the MicroLET Protocol, within a 5X5 SDNoC topology, are evaluated. The proposed routing algorithm is compared to the XY and the classic OE by using different traffic scenarios, the results are presented on Figure 6, Figure 7 and Figure 8.

Under uniform traffic, the classic OE and the OE with a selection function have lower performance than XY routing. This is due to unreliability of the Network Monitoring Phase. On the other hand, under the transpose and bit-reverse traffic, the proposed routing algorithm outperforms the XY and OE routing algorithms. Indeed, under such traffic scenarios, the controller relies on an accurate view of the network state abd it is able to balance the traffic across the network by avoiding the form of congested network areas. Conversely, under these scenarios, XY pushes the traffic towards the same links and switches. Therefore, the corresponding network areas become congested, which leads to a network performance decrease.

The proposed routing algorithm relies on the OE algorithm, which is partially adaptive and therefore, restricts the number of admissible routes. Secondly, the controller responds to the arriving ROUTE_REQUEST messages by allocating routes without being aware of future routes. Thereby, when the controller searches to allocate a route for a source-destination pair, it is possible that all the admissible routes being occupied, while there are still possible routes (but not admissible) flowing through unoccupied resources. This scenario can happen during some simulations according to the

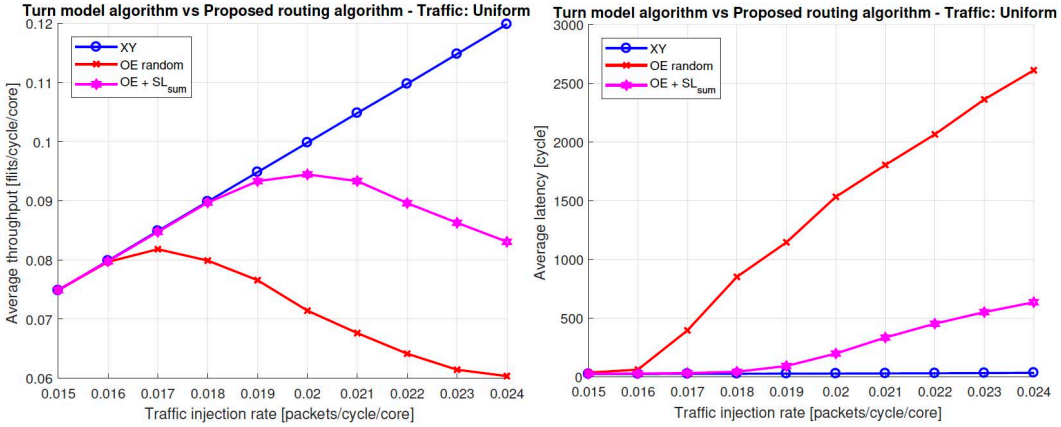

Figure 6: Performance of the proposed routing algorithm ($OE+SL_{\text{sum}}$) as compared to the turn model based routing algorithms with uniform traffic.

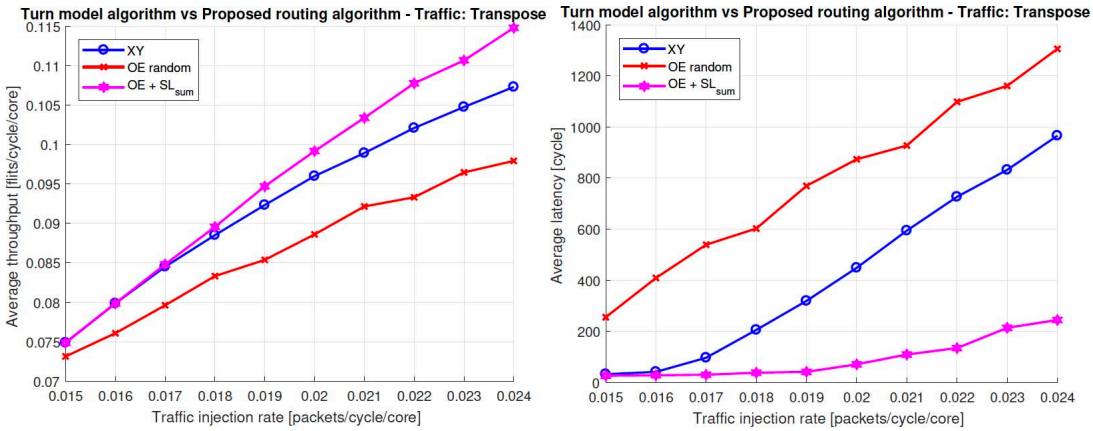

Figure 7: Performance of the proposed routing algorithm ($OE+SL_{\text{sum}}$) as compared to the turn model based routing algorithms with transpose traffic.

traffic. If the controller will be aware beforehand about the future `ROUTE_REQUEST`, it could adapt the allocation of the routes accordingly in order to avoid this scenario. Moreover, the lack of knowledge concerning the future requests is less critical if the controller uses a fully adaptive routing. Despite the higher standard deviations, the proposed routing approach still outperforms the XY routing algorithm under transpose and bit-reverse traffic and hence we believe that it could be a possible solution for chiplet-based systems.

VII. Conclusion and Future Work

In this paper, we propose a new SDNoC communication protocol for inter chiplet-based systems called MicroLET. The protocol is designed in order to provide a new routing approach based on SDN technology and a new message stack specifically designed for micro-scale networks. Furthermore through the evaluation of the MicroLET protocol, we proved that it could be a good candidate for the future chiplet-based systems. With the help of the flexibility which rises through the SDN-based approach, it is possible to accommodate any kind of routing techniques within the software based controller placed in the chiplet. Also, within the chiplet-based controller some machine learning algorithms can be implemented and tested during the run time in order to efficiently choose the appropriate routing algorithm respected to application requirements.

The MicroLET Protocol is designed and evaluated in order to cover the intra-chiplet communication. Since this paper refers to different hierarchy levels of communications, a different communication protocol should be designed in order to cover the communication between the NoC that is placed in the interposer and the NoC which is placed within the chiplets. A very nice routing composable, topology agnostic, deadlock-free routing methodology is presented on [26] that covers the whole chiplet-based system. We do believe that the integration of SDNoC approach in combination with the work proposed in [26] could be a strong contribution for future chiplet-based systems.

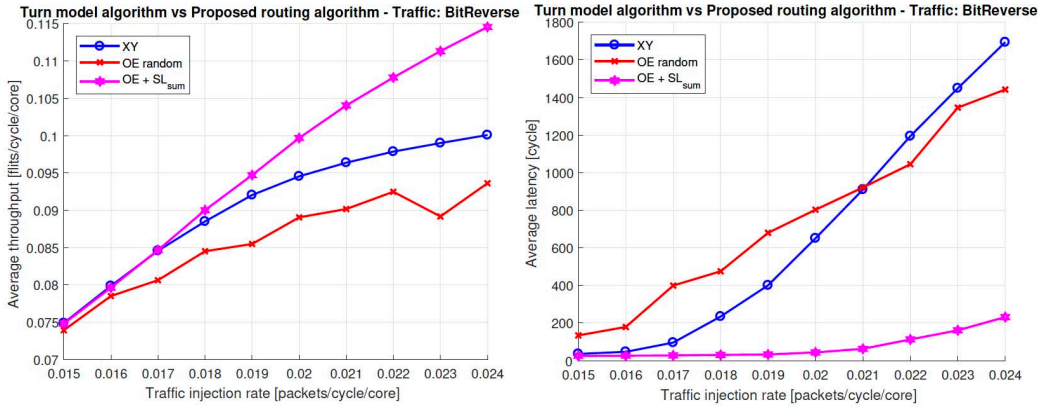

Figure 8: Performance of the proposed routing algorithm ($OE+SL_{\text{sum}}$) as compared to the turn model based routing algorithms with BitReverse traffic.

REFERENCES

[1] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *2009 IEEE international symposium on performance analysis of systems and software*, pages 33–42. IEEE, 2009.

[2] Luca Benini and Giovanni De Micheli. Networks on chips: A new soc paradigm. *Computer-IEEE Computer Society-*, 35(EPFL-ARTICLE-165542):70–78, 2002.

[3] Konstantin Berestizshevsky, Guy Even, Yaniv Fais, and Jonatan Ostrometzky. SDNoC: Software defined network on a chip. *Microprocessors and Microsystems*, 50:138–153, 2017.

[4] E Beyne and AL Manna. High-bandwidth chip-to-chip interfaces: 3d stacking, interposers and optical i/o. In *IMEC technology forum, Taiwan*, 2013.

[5] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

[6] Cadence. Imec and Cadence Tape Out Industry's First 3nm Test Chip, Press Release. https://www.cadence.com/content/cadence-www/global/en_US/home/company/newsroom/press-releases/pr/2018/imec-and-cadence-tape-out-industry-s-first-3nm-test-chip.html. Accessed: April 2019.

[7] Ge-Ming Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on parallel and distributed systems*, 11(7):729–738, 2000.

[8] Liu Cong, Wang Wen, and Wang Zhiying. A configurable, programmable and software-defined network on chip. In *Advanced Research and Technology in Industry Applications (WARTIA), 2014 IEEE Workshop on*, pages 813–816. IEEE, 2014.

[9] Intel Corporation. "New Intel Core Processor Combines Highperformance CPU with Custom Discrete Graphics From AMD to Enable Sleeker, Thinner Devices,". http://newsroom.intel.com.

[10] Marvell Corporation. Marvell ARMADA 8040 Quad-Core CA72 Processor with Marvell MoChi and FLC Architecture, Tech. Rep. http://www.marvell.com/embedded-processors/assets/Armada8040PB-Jan2016.pdf.

[11] Soultana Ellinidou, Gaurav Sharma, Jean-Michel Dricot, and Olivier Markowitch. A SDN solution for system-on-chip world. In *Software Defined Systems (SDS), 2018 Fifth International Conference on*, pages 14–19. IEEE, 2018.

[12] Subramanian S Iyer. Heterogeneous integration for performance and scaling. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 6(7):973–982, 2016.

[13] Ajaykumar Kannan, Natalie Enright Jerger, and Gabriel H Loh. Enabling interposer-based disintegration of multi-core processors. In *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 546–558. IEEE, 2015.

[14] KEVIN Lepak, GERRY Talbot, SEAN White, NOAH Beck, S Naffziger, SENIOR FELLOW, et al. The next generation amd enterprise server product architecture. In *Proc. Hot Chips*, pages 1–22, 2017.

[15] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[16] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[17] Tesla NVIDIA. P100 white paper. *NVIDIA Corporation*, 2016.

[18] R Sandoval-Arechiga, JL Vazquez-Avila, R Parra-Michel, J Flores-Troncoso, and S Ibarra-Delgado. Shifting the network-on-chip paradigm towards a software defined network architecture. In *Computational Science and Computational Intelligence (CSCI), 2015 International Conference on*, pages 869–870. IEEE, 2015.

[19] Remberto Sandoval-Arechiga, Ramón Parra-Michel, JL Vazquez-Avila, Jorge Flores-Troncoso, and Salvador Ibarra-Delgado. Software defined networks-on-chip for multi/many-core systems: A performance evaluation. In *Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems*, pages 129–130. ACM, 2016.

[20] Alberto Scionti, Somnath Mazumdar, and Antoni Portero. Software defined network-on-chip for scalable cmps. In *High Performance Computing & Simulation (HPCS), 2016 International Conference on*, pages 112–115. IEEE, 2016.

[21] Daniel P Seemuth, Azadeh Davoodi, and Katherine Morrow. Automatic die placement and flexible i/o assignment in 2.5 d ic design. In *Sixteenth International Symposium on Quality Electronic Design*, pages 524–527. IEEE, 2015.

[22] Théo Rigas Tristan Vanspouwen Olivier Markowitch Soultana Ellinidou, Gaurav Sharma and Jean-Michel Dricot. "sspsoc: A secure sdn-based protocol over mpsoc.". volume 2019, 2019.

[23] Sehat Sutardja. 1.2 the future of ic design innovation. In *2015 IEEE International Solid-State Circuits Conference-(ISSCC) Digest of Technical Papers*, pages 1–6. IEEE, 2015.

[24] Thiruvengadam Vijayaraghavan, Yasuko Eckert, Gabriel H Loh, Michael J Schulte, Mike Ignatowski, Bradford M Beckmann, William C Brantley, Joseph L Greathouse, Wei Huang, Arun Karunanithi, et al. Design and analysis of an apu for exascale computing. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 85–96. IEEE, 2017.

[25] Shien-Yang Wu, CY Lin, MC Chiang, JJ Liaw, JY Cheng, SH Yang, CH Tsai, PN Chen, T Miyashita, CH Chang, et al. A 7nm cmos platform technology featuring 4 th generation finfet transistors with a 0.027 um 2 high density 6-t sram cell for mobile soc applications. In *2016 IEEE International Electron Devices Meeting (IEDM)*, pages 2–6. IEEE, 2016.

[26] Jieming Yin, Zhifeng Lin, Onur Kayiran, Matthew Poremba, Muhammad Shoaib Bin Altaf, Natalie Enright Jerger, and Gabriel H Loh. Modular routing design for chiplet-based systems. In *Proceedings of the 45th Annual International Symposium on Computer Architecture*, pages 726–738. IEEE Press, 2018.