A Flexible Software/Hardware Adaptive Network for Embedded Distributed Architectures

Celine Azar

Department of Physics and Electronics, Faculty of Sciences,
Lebanese University, Beirut, Lebanon

ABSTRACT

Embedded platforms are projected to integrate hundreds of cores in the near future, and expanding the interconnection network remains a key challenge. We propose SNet, a new Scalable NETwork paradigm that extends the NoCs area to include a software/hardware dynamic routing mechanism. To design routing pathways among communicating processes, it uses a distributed, adaptive, non-supervised routing method based on the ACO algorithm (Ant Colony Optimization). A small footprint hardware unit called DMC speeds up data transfer (Direct Management of Communications). SNet has the benefit of being extremely versatile, allowing for the creation of a broad range of routing topologies to meet the needs of various applications. We provide the DMC module in this work and assess SNet performance by executing a large number of test cases.

KEYWORDS

Distributed manycore architectures, scalable network, hardware/software network paradigm.

1. Introduction

The age of giga transistor devices is approaching [1], with 10 nm technology predicted soon, thus allowing to integrate a whole system, including thousands of IPs, onto a single die. The ability of these gigascale SoCs (System-On-Chip) to efficiently link pre-designed functional blocks and satisfy their communication needs will be one of its limitations [2].

Computation has always been expensive, whereas communication has been cheaper, however this has changed with scaling systems [3]. With the advancement of technology, computation is getting more affordable, allowing for the integration of thousands of processors on a single chip. Given their quantity and therefore their capacity to implement a large number of running programs, future processors may be viewed as real transistors. Communication, on the other hand, presents significant problems, which may be summarized in three areas: technology, performance, and design productivity issues [2]. Technology issues arise from non-scalable global wire delays, though gate delays decrease as technology improves. Due to submicron effects (clock skew, power associated with clock distribution trees, etc.), core synchronization will be impossible, and another solution will consist of self-synchronous cores communicating through a network-centric architecture [4]. Interconnection systems based on shared buses are causing performance difficulties and will not satisfy the connectivity needs of future SoCs [5]. In terms of design productivity, because synthesis and compiler technology are not keeping up with IC production [6], and because time-to-market is critical, modular reusable on-chip networks are the solution for dedicated bus-based designs.

While many experts predict gigascale manycores in the near future, with scalability as the ultimate objective, one potential design approach is distributed homogeneous architectures, which prioritize local communication exchanges and links between neighbour cores. However, we have reservations about the capacity to manage communications in platforms with thousands of processors while maintaining strict local routing rules. Designers have developed a network-centric strategy based on Network-On-Chips (NoCs) to solve technological difficulties, integrating techniques established for macro-scale, multi-hop networks into a die. The NoC paradigm's main objective is to increase design productivity and performance by dealing with growing parallelism [7]. Despite the fact that the NoCs area has been extensively explored, we believe there is a need to develop a new technique that provides greater flexibility and dynamicity in order to meet future requirements. In reality, future manycore systems are expected to run sophisticated and dynamic applications with variable computational and communication requirements. In addition, component failures may occur in the system, which must be handled by dynamically modifying the routing pathways without user intervention.

While NoCs are a good answer for today's network topologies, we expect that in future manycore designs with thousands of cores on chip, more efficient interconnection techniques would be provided. We suggest allocating a core to a routing process to add "intelligence" to the routing mechanism. Future manycores will include thousands of cores, as previously stated, and the computing resource will be inexpensive given its abundance. As a result of the large number of cores integrated on the same chip, some will be left idle, and routing processes may be allocated to these unoccupied processors. They will establish a routing network with a completely flexible topology, referred to as routing topology. Routing cores are responsible for creating routing pathways amongst communicating processes in a distributed dynamic manner without any centralized control. Data is transmitted using a network of dedicated modules known as DMC, which minimize the transmission latency once routing pathways have been established. Because it allocates a unique and predetermined routing task to the chosen routing PEs, this technique enhances design productivity and simplifies the programmability issue. As a result, we move away from NoCs' communication-centric strategy and toward a PE-centric approach, with the PE (Processing Element) handling communication management.

The SNet [8] scalable dynamic communication paradigm is discussed in this article. The goal of this proposal is to improve future manycores' flexibility, scalability, and programmability. To design routing pathways, SNet uses an adaptive, bio-inspired routing approach that implements the ACO metaheuristic [9]. After that, we will go through the DMC's hardware design. Data transmission is handled by a network of DMCs. The state of the art on existing NoCs is presented in the next section. We offer an overview of the SNet principle and the DMC architecture in section 3. We compare the SNet routing characteristics to those of existing NoCs in section 4. Section 5 concludes with a summary of future projects.

2. NOCS: A STATE OF THE ART

The NoCs area has been extensively studied, and we will focus in this paper on two features of NoCs: interconnection topology and routing strategy. The arrangement and connections between cores are defined by the interconnection topology. As a result, NoCs can have a fixed interconnection topology or an adjustable topology that can be modified to meet the needs of the applications. Furthermore, the routing strategy uses either deterministic protocols, such as the XY approach, or adaptive rules, which establish routing pathways dynamically based on network demand. Figure 1 illustrates instances of NoCs for various types of routing interconnections and strategies.

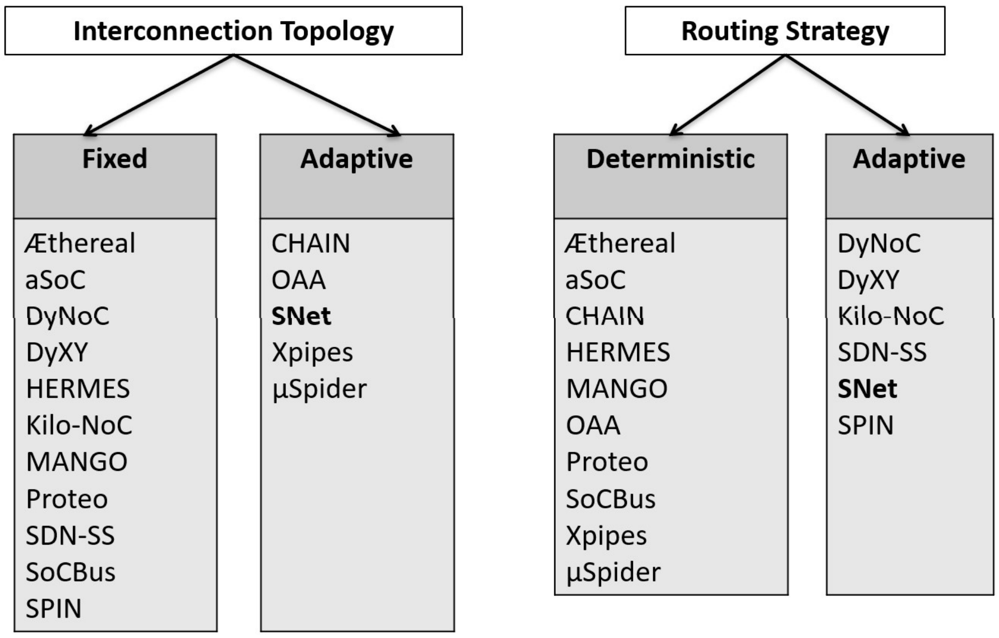


Figure 1. Examples of NoCs for various routing interconnections and strategies.

Fixed NoC topologies (such as mesh networks and fat trees) are differentiated by their tile-based connections, whereas adaptive NoCs have no pre-designed interconnections and instead build a flexible network to map numerous topologies, each fitting the demands of unique application requirements (such as latency or throughput). Figure 1 shows examples of all the above-mentioned categories of NoCs.

Regular NoCs include Æthereal [10], aSoC [11], DyNoC [12], DyXY [13], HERMES [5], MANGO [14], Proteo [15], SDN-SS [18], SoCBus [16], and SPIN [17]. aSoC and Hermes are mesh networks that use n-port switches with $(n - 1)$ ports for connecting to neighbor switches and one port for connecting to the local IP block interface, where n = 4 and 5 respectively. SoCBus and aSoC implement circuit switched deterministic routing and assign one virtual channel. SoCBus uses a minimal path-length routing method to establish the routes statically. In the case of aSoC, communication is pre-established in a reconfigurable statically-scheduled system aimed at data-intensive applications. Hermes uses packet switching instead of circuit switching, employing the XY deterministic approach and assigning one virtual channel. SDN-SS makes a step towards an adaptive routing strategy while implementing a 2D mesh topology. It uses a centralized controller which collects the requests from the system manager and searches for the shortest path between source and target tasks, while avoiding contentions. MANGO not only implements the mesh topology, but it also provides numerous virtual channels to accomplish QoS without affecting bandwidth, and it supports both guaranteed services (GS) and best effort (BE) routing. BE routing covers communications when just accuracy and completeness are ensured, while GS routing adds timing obligations such minimum throughput or maximum delay. The Æthereal channel allows point-to-point communication between two network interfaces and supports both GS and BE services. Offline determined configurations, each corresponding to a needed user feature, are loaded to the network. To deal with network congestion, DyNoC and DyXY construct a mesh topology and develop an adaptive variant of the XY deterministic routing system. Routing in DyNoC includes three modes: N-XY (Normal XY), which acts as a standard XY router, SH-XY (Surrounding Horizontal XY), which is used when the right or left routers are congested, and SV-XY (Surrounding Vertical XY), which is used when the upper and lower routers are overloaded. When it comes to routing, DyXY uses an XY scheme until a network congestion arises. To get past the overloaded zone, it then transfers the packet to the least loaded neighbor.

SoCBus, aSoC, HERMES, MANGO, Æthereal, DyNoC, and DyXY use 2D mesh topologies, whereas Proteo [15] and SPIN [17] use hierarchical networks. Proteo is a hierarchical packet-switching network based on a deterministic hop-by-hop routing algorithm with numerous virtual channels and built from tiny rings. SPIN, on the other hand, features a fat-tree architecture with one virtual channel and adaptive routing. The routers are free to use any of the fat tree's redundant paths, which reduces congestion and improves speed.

Kilo-NoC [19] chooses a MECS (Multidrop Express Channels) topology which uses point-to-multipoint channels connecting each source node to multiple destinations. It uses elastic buffers to optimize QoS while reducing the buffering cost. A customed routing strategy is proposed to exploit the underlying architecture and prevent contentions.

CHAIN [20], OAA [21], µSpider [22], and Xpipes [2], on the other hand, are designs for adaptive networks, each built at a different system level. CHAIN is a deterministic routing system that uses fine grain components to target heterogeneous low-power devices. The µSpider network may be customized using a CAD tool that creates NoC VHDL code at the RTL level, allowing designers to create ad hoc NoCs based on application requirements. The interconnection topology and routing protocol, for example, are design-time adjustable µSpider features. The OAA network may also have its interconnection topology configured at design time. Xpipes, on the other hand, offers a high-level NoC design and implements a library of design-time composable soft macros (switches, network interfaces, and links) that can be used to instantiate and synthesize domain- specific heterogeneous architectures.

When it comes to routing strategies, CHAIN, Xpipes, and µSpider don't provide dynamicity at the protocol level. SPIN, DyNoC, and DyXY all promise this functionality, but they all have a defined interconnection architecture. As a result, we introduce SNet, a unique network that blends flexible connectivity topology with dynamic routing strategy to provide a design that is suited for future applications and system needs. The DMC accelerator decreases the space needed while improving performance. SNet was developed to operate with any interconnection topology.

3. THE SNET PARADIGM

3.1. Description of SNet

Scalability and flexibility are improved using the SNet approach. Local communications between neighboring processors are prioritized, and they can be organized in any regular pattern. After the layout has been established, the neighborhood, or the number of neighbors for each processor, is decided to define the interconnection network. As a result, the mesh topology is a 4-neighbor square arrangement.

Communication requests are routed across shared buffers, with each buffer connecting the two CPUs on either side. The distributed ACO algorithm, which may fit any layout/neighboring setup, is being used as SNet's routing strategy. Furthermore, SNet allows users to assign processing and routing tasks to any of the available cores. As a result, routing PEs create a completely configurable routing architecture. One of the major motivations for using RISC cores in processors is to duplicate small-footprint simple cores.

We chose a manycore platform, called CEDAR [9], to verify the SNet idea, which consists of an array of homogenous RISC cores coupled in a mesh manner via shared buffers. The selected cores have a minimal footprint of 16 $KGates$ and contain a 4 KB instruction memory and a 4 KB local data memory in the platform's actual implementation. [9] has further information about the CEDAR platform and the ACO algorithm. The following are the features of SNet:

- Communication is done by running the ACO algorithm on an arbitrary number of cores using a routing process. As a result, processors are split between routing and calculating PEs. The user has the option of defining any routing topology.

- The ACO algorithm creates dynamic routes between remote communication tasks without the need for human intervention. As a result, the user is free of programming responsibilities. To decrease congestion, paths are spread evenly across routing nodes.

- Data transfers are handled by the DMC module, which is a co-processor linked to each core to speed transmission once routing pathways are dynamically constructed. The DMCs are linked in a mesh manner on the specified platform.

Multitasking is not handled in the current implementation of SNet, but it will be in future works. The goal of multitasking is to enhance the core usage rate. After the routes are created by the routing PEs, the execution moves on to other activities, which might be computation or control tasks. We can also opt to turn off the routing PEs in order to save power consumption. As a result, the SNet concept's routes generation phase may be thought of as an initialization operation that can be revisited during runtime to adjust the routing paths to the current system state and application requirements.

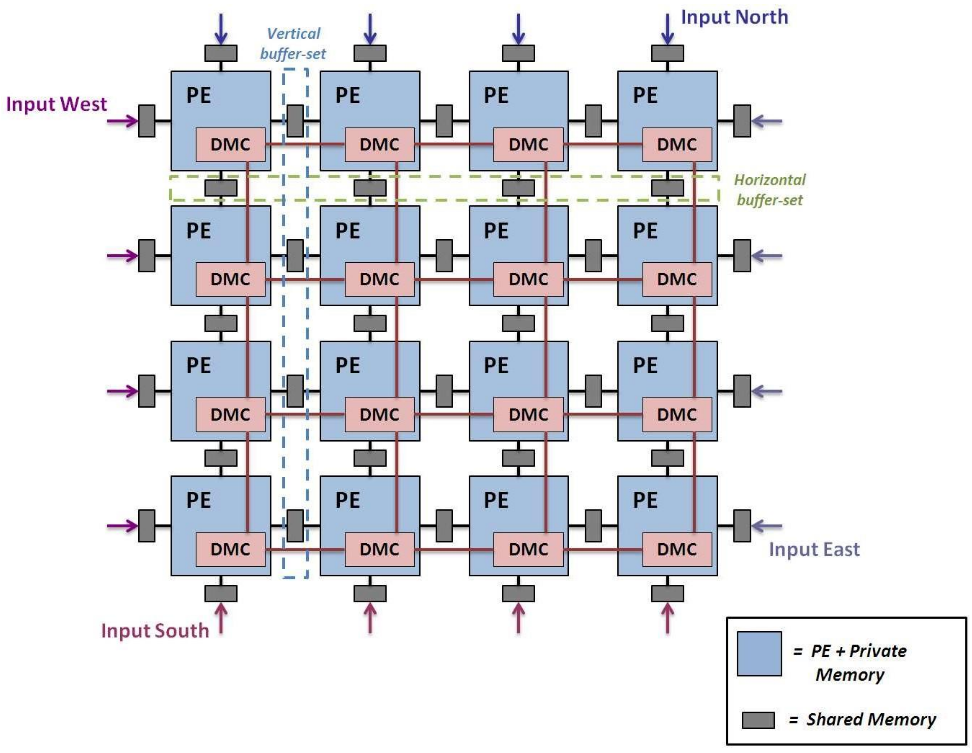


Figure 2. The CEDAR-H platform featuring a network of DMCs.

3.2. The DMC unit

The DMC (Direct Management of Communication) is a brand-new element that has been added to the CEDAR platform [9]. It is tied to each CPU in order to speed up data transfers between distant tasks. As a result, the PEs handle the path construction step in software, while the DMCs handle the data transfers. As illustrated in Figure 2, the DMCs are connected in a point-to-point way, and packets are routed in a wormhole form. To distinguish between the two platforms, we will designate CEDAR-S (as for CEDAR-Software) the original one described in [9]. Furthermore, the platform, which includes the DMCs network, will be known as CEDAR-H (as for CEDAR-Hardware) (cf. Figure 2).

Figure 3 shows the DMC block diagram. It has a link table that is dynamically updated during path construction by the ACO algorithm. As a result, the link table is a local routing table that includes compact data for data routing locally. A Finite State Machine (FSM) is included in the DMC module to manage many requests arriving at the same time while using a round robin checking technique. A switch is added to this to reroute data in the next valid direction. In addition to registers and multiplexers for control, one shifter and five AND gates are added for data processing. Figure 4 shows a closer view of the DMC co-processor.

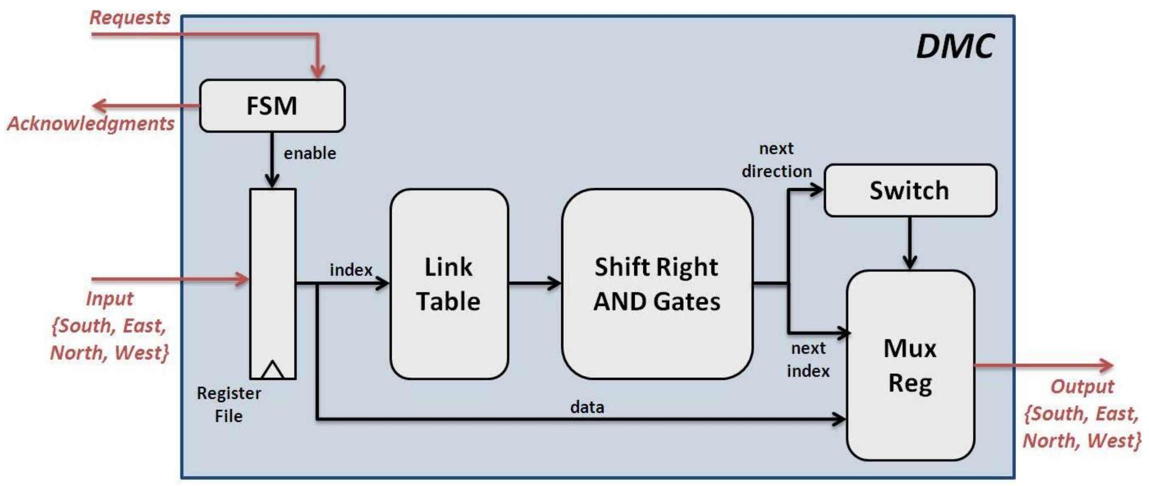


Figure 3. The DMC unit's block diagram.

The DMC unit works like this: It starts by looking for requests arriving from one of four directions: South, East, North, or West. Each request consists of two parts: a 32-bit data to forward and a 6-bit table index that allows the link table to retain 64 simultaneous interactions. The number of communications is parameterizable at compile time and can be changed. When a request is identified, the index is utilized to obtain the next transfer index and direction from the link table. As a result, the tuple $\{data/new\ index\}$ is routed in the following direction.

The DMC module is extremely efficient, just using one cycle for each data transmission. The network has a bandwidth of 9.6 $Gbps$ in its current format. The routing paths established at runtime determine the minimal message delay. It is possible that the selected route is not the shortest. The ACO routing protocol considers two criteria while creating these paths: the distance between the source and destination tasks, and the total of the loads of all crossed routing PEs. The latter option aids in distributing routes equitably across available resources. We compare the SNet approach to state-of-the-art NoCs to assess its flexibility, area, and performance. The overhead incurred by devoting a core to perform routing is discussed in the next section. The average message latencies vs the injection rates are shown to evaluate performance. We also compare the DMC module's performance to that of [8], following data optimization.

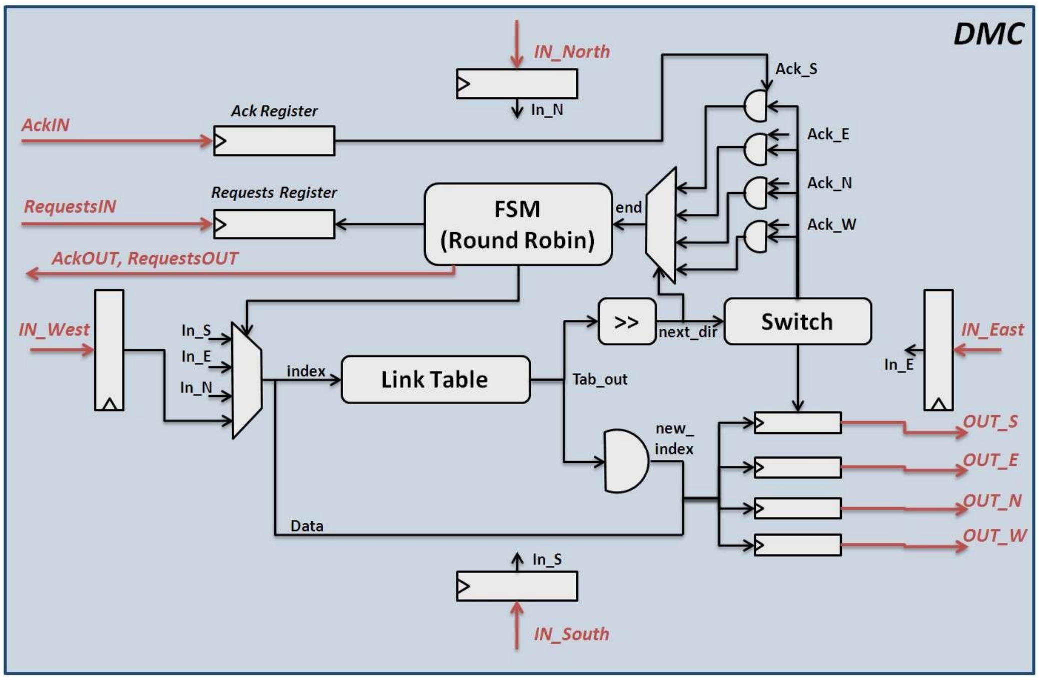


Figure 4. A detailed version of the DMC components.

4. Performance Comparisons

4.1. Area Comparisons

The primary characteristics of NoC designs are their parametrizability and granularity [3]. The ease with which a system-level NoC attribute can be modified at instantiation time is referred to as parametrizability. Thus, it is the NoC's degree of flexibility at the system level, i.e., the number of malleable factors such as the number of virtual channels in the switch, pipeline stages in the connections, and so on. Granularity refers to the level of description of the NoC, which is provided as a single IP at the coarse level and as a series of constructed blocks at the fine level. Figure 5 depicts a diverse set of NoCs that fill the design space defined by these criteria. To this figure from [3], the SNet network has been added.

CHAIN [20], for example, provides a library of fine-grained NoC components that the designer may utilize to construct the appropriate NoC. While the CHAIN NoC is design-flexible, it offers limited system-level flexibility. The network is described as a rather coarse grain system-level module by Æthereal [10], SoCBus [16], and aSoC [11], each with its own set of features. Æthereal has a lot of flexibility when it comes to changing the available slots, whereas aSoC and SoCBus may not provide a lot of options. However, aSoC allows for flexible connection programming after instantiation, making it more versatile than SoCBus. HERMES [5], which has a parameterizable input queue, operates at the same degree of flexibility. With its fixed architecture and protocol, SPIN [17], built as a single IP core, is the least parameterizable. Xpipes [2] offers a collection of fine-grain soft macros of switches and pipelined connections that the Xpipes compiler uses to create an application-specific network automatically. As a result, the granularity and parametrizability levels are in the midrange. Similarly, the Proteo NoC [15] falls into the same category since it has adjustable control and data sizes and is made up of tiny ring networks. In the case of the MANGO NoC [14], its router allows for the assumption of GS and BE service quality, even though the NoC is specified at a very coarse granularity level.

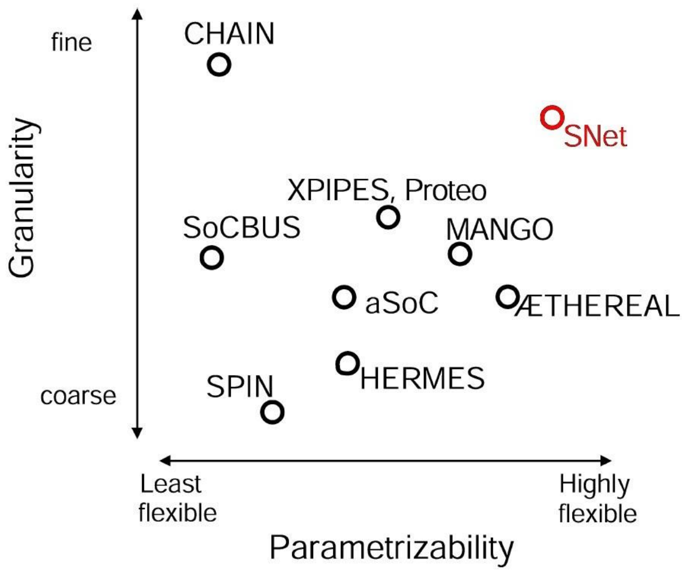


Figure 5. Positioning a collection of NoCs and the SNet paradigm according to the granularity and parametrizability requirements in the NoCs design space [3].

Table 1 [7] lists the router overhead of the previously stated NoCs. Due to FPGA synthesis or lack of information, calculations for the DyNoC, DyXY, Kilo-NoC, OAA and µSpider are not provided. Synthesis methods are performed using various technologies, and we compare them by scaling the dimensions of the specified NoCs to 40 nm TSMC.

Table 1. Routers areas for the studied NoCs.

NoCs	**Tech.** (nm)	**Router area** (mm^2)	**Router area** $(KGates)$	**Est. Router Area,** 40 nm, (mm^2)
Æthereal	120	0.26	–	0.0234
aSoC	180	0.07	–	0.0018
CHAIN	350	–	4.75	0.0029
HERMES	350	–	10 – 49	0.006 – 0.03
MANGO	130	0.19	–	0.0105
Proteo	180	0.20	–	0.0052
SDN-SS	28	0.0085	–	0.0105
SNet	40	0.012 – 0.014	16	0.012 – 0.014
SoCBus	180	0.04 – 0.08	–	0.001 – 0.002
SPIN	130	0.24	–	0.0132
Xpipes	100	0.1	–	0.0104

We first calculate the total number of gates and then evaluate the area at 40 nm TSMC. Buffers are included in the routers zones, and their implementation varies per NoC. The number of virtual channels (VC), the number of ports, the buffer depth, and the flit width are all listed in Table 2 [7]. By multiplying the four buffer parameters, the total buffer size may be computed. The buffer sizes for the selected NoCs range from 12 $bytes$ to 600 $bytes$. SNet's router area is computed by adding one core area to the area of the DMC co-processor (0.162 μm^2 at 40 nm TSMC) and the overhead of the four shared buffers that connect each core to its four closest neighbours. In line with the computed range of buffer sizes, we evaluate two typical memory sizes, 128 $bytes$ and 512 $bytes$, and the two results are shown in Table 1.

Table 2. Organization of buffers for the evaluated NoCs [7].

NoCs	**VC**	**Ports**	**Buffer Depth** ($flits$)	**Flit Width** ($bits$)	**Buffer Size** ($bytes$)
Æthereal	1	5	8	$3 * 32$	480
aSoC	1	4	2	32	32
CHAIN	n/a	n/a	n/a	n/a	n/a
HERMES	1	5	5 – 30	32	100 – 600
MANGO	8	5	1	32	160
Proteo	1	3	7	8	21
SoCBus	1	3	1	16	6
SPIN	1	8	4 ?	32	128 ?
Xpipes	1	4	?	32	n/a

We can conclude that the least flexible NoCs, such as CHAIN and SoCBus, have the most reduced areas. While being part of a very flexible network, the SNet router's area takes a middle range value between Xpipes, SDN-SS and MANGO from one part and SPIN, HERMES, and Æthereal from the other:

- The interconnection architecture is completely configurable, allowing SNet to be designed in a multitude of contexts.
- The number of virtual channels can be increased or decreased as needed. In the actual implementation, it is set to 64 channels, which is the length of the DMC module's link table. This table length is parameterizable and may be adjusted at compile time to meet the needs of the application.
- Pathways between communicating tasks are established at runtime and may alter based on the load at each run.

- Users may configure some PEs as routers to map any routing topology on the platform at instantiation time, allowing them to map any routing topology on the platform. By replacing the common router with a core, the network gains more programmability and data control and synchronization is handed to the network level.

Thus, the benefit of SNet is that it provides a programmable, highly flexible communication network design while still offering a tiny footprint router with a significant memory space.

4.2. Performance comparisons

4.2.1. Simulation Approach

The comparison of performance with state-of-the-art NoCs is a sensitive operation, since different simulation configurations provide a wide range of observed results. As Salminen et al [23] point out, measurement setups in NoC publications are frequently imprecise, resulting in biased comparisons. These can be resumed by the following questions:

- How often are bursts of packets injected at a router, and how large are these bursts?

- How far do packets travel in the network from a source node towards their destination node?

- What proportion of total traffic does each router inject into the network?

By using defined simulation settings and realistic traffic models, we choose an evaluation technique proposed in [24] to extract the features of various routing topologies applied to SNet. As a result, we provide a theoretical study aimed at characterizing the SNet paradigm. 16 computational nodes are assigned to the platform, which size varies between 6×6 and 8×8 processors array, depending on the routing topology. We do handle huge systems in this study, with thousands of cores, but we focus here on constrained dimensions in order to define SNet in a reasonable time frame. Each platform node sends 640 messages with n flits to the network, with a flit width of 32 $bits$. n is changed between 10 and 1000 $flits$, which is equivalent to 25 KB and 2.44 MB respectively, to evaluate the network saturation limits. In order to mimic realistic traffic, such as in application runnings, injection is conducted by source processes rather than by a traffic generator. The simulations are run on the ISS simulator platform, which is based on an ISS array of RISC processors and simulates one instruction every cycle. The Poisson injection distribution is used to simulate traffic, and the tested routing topologies, which consist of different task mappings, are illustrated in Figure 6. Routing PEs are represented with dark grey nodes, computational PEs are in light grey, and unassigned PEs are white nodes. We emphasize the benefit of having invariable source codes regardless of the mapping. Because the ACO algorithm is completely decoupled from the actual location of processes, changing the arrangement of computing/routing nodes has no effect on the executed programs. The number of PEs allocated to execute a routing task is what differ between different topologies, while the number of computing PEs remains the same.

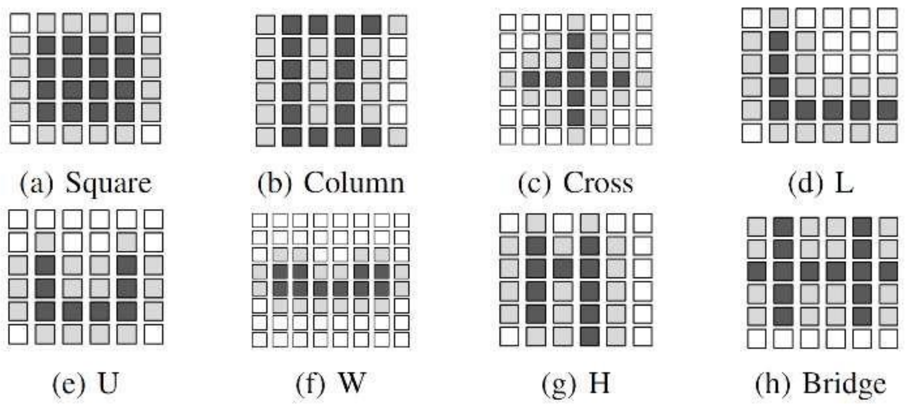


Figure 6. Routing topologies evaluated on the manycore platform.

4.2.2. Conducted Measurements

The injection rate and average message delay parameters are used to evaluate performance. Equation (1) calculates the injection rate:

where $theoreticalDataFlow$ is the bandwidth, which is determined by the number of routers in the topology under study, and $measuredDataFlow$ is calculated using simulation. We look at three possible task distributions across the cores and how they affect latency [24]:

$$injectionRate = \frac{measuredDataFlow}{theoreticalDataFlow} \quad (1)$$

- **Spaced mapping** is the worst case scenario, and it is based on a spatially uniform traffic distribution.

- **Localized mapping** at a rate of 0.5 is an intermediate scenario.

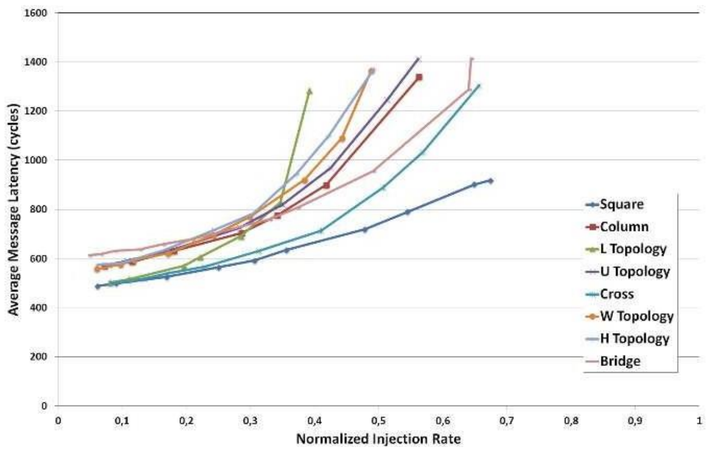
- **Best mapping** puts the communicating tasks as near as possible, aiming LF = 1 if the topology permits it.

The ratio of local traffic to overall traffic is referred to as the localization factor (LF). As per a 0.5 localization factor, 50% of the traffic generated by source PEs is transmitted to destination PEs that are one hop away, while the remaining traffic is divided among destination PEs that are two hops apart.

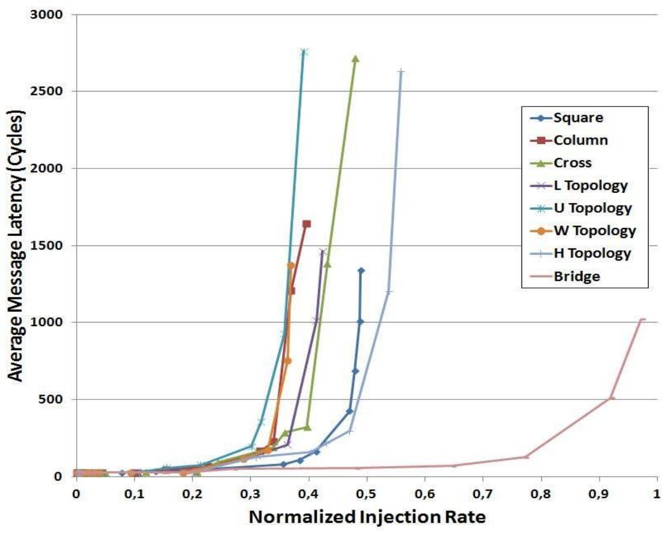
4.2.3. Results

[8] shows measured latencies for the examined routing topologies under spatially uniform traffic distribution and best mapping, whereas Figure 7 (a) and (b) show values for localized mapping with LF = 0.5 for the CEDAR-S and CEDAR-H platforms, respectively. Figure 7 (a) shows that the square topology has the best performance, with a maximum injection rate of 0.67. This is due to a 50% reduction in the number of remote communications requiring more than two hops to reach destinations. The square and bridge topologies may both be observed in this way. The column topology outperforms both L and U topologies. When localized mapping is used, network traffic is decreased, and the relatively large number of routers manages contentions more efficiently than in the L and U topologies, where the number of routers is halved when compared to the column topology. This in no way diminishes the significance of the L and U topologies. It is a decision the user must take based on the application's requirements, a trade-off between performance, area efficiency, and topology which best represents task mapping and communication requirements.

We notice in Figure 7 (b) a shift to the right for the collection of curves when we change to the CEDAR-H platform. There are three distinct groups in the figure. The L, U, W, Cross, and Column topologies are present in the first. These have the lowest injection rates, with the Cross topology reaching a maximum of 0.4. The second set includes the Square and H topologies, which, because of their larger number of routing PEs, are able to better handle network congestions. They yield nearly identical results, with a maximum injection rate of 0.47. The third category is reserved for the bridge topology, which has the highest injection rate equal to 0.8.



(a) Simulations conducted on the CEDAR-S platform.



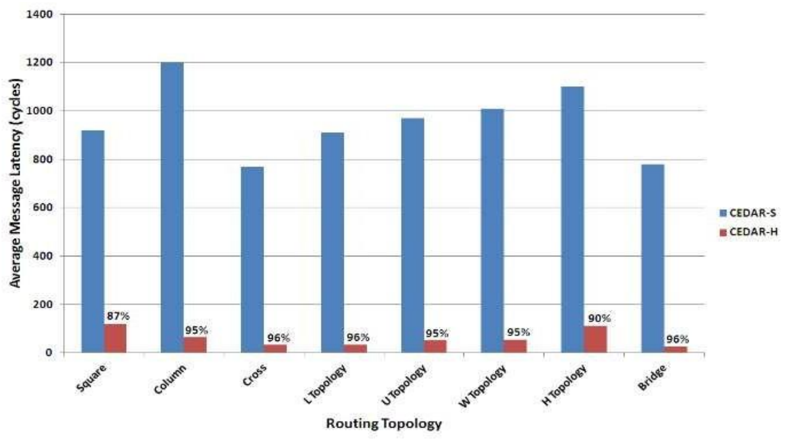
(b) Simulations conducted on the CEDAR-H platform.

Figure 7. Latency variation with Poisson injection distribution for Localized Mapping (LF = 0.5).

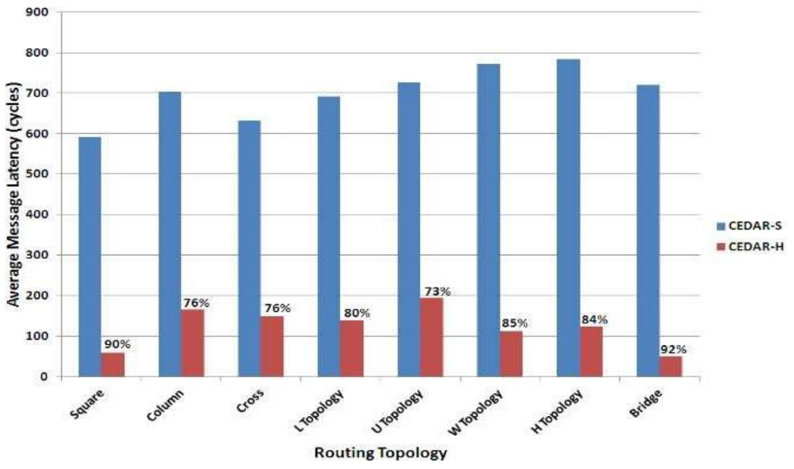
We find that the examined routing topologies span a wide range of injection rate and delay values. Choosing one of these topologies, or any of the many others, is a matter of the implemented application. The CEDAR-H platform outperforms CEDAR-S while only occupying a tiny amount of space overhead. The results indicate that we can get excellent performance even when striving for flexibility and scalability.

4.2.4. DMC performances

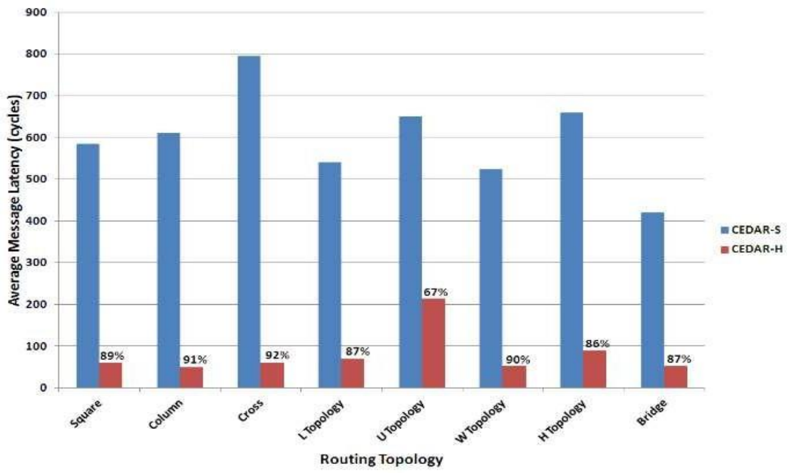
The same simulation experiments are conducted on the CEDAR-S and CEDAR-H platforms to quantify the DMC performance. To compare the two systems, a fixed injection rate is used, and average message latencies for all routing topologies are displayed (cf. Figure 8). An injection rate equal to 0.2 is chosen for spaced mapping, and 0.3 for localized and best mappings. This decision was made in order to calculate an average message delay value for all topologies. On the CEDAR- H platform, for example, the maximum injection rate for the L Topology under spaced mapping is equal to 0.2, hence raising the injection rate will result in no latency for that routing topology.



(a) Spaced Mapping



(b) Localized mapping



(c) Best mapping

Figure 8. Average message latency for CEDAR-S and CEDAR-H platforms under: (a) Spaced mapping, (b) Localized mapping and (c) Best mapping.

As seen in Figure 8 (a), under spaced mapping, the latency values range between 790 and 1200 cycles for CEDAR-S, and between 30 and 120 cycles for CEDAR-H. A performance gain of 94% is achieved by CEDAR-H compared to the CEDAR-S platform. For the localized mapping (cf. Figure 8 (b)), the latency values range between 590 and 790 cycles for the CEDAR-S platform, and between 50 and 195 cycles for the CEDAR-H platform. The latter scores a gain of 82% compared to the CEDAR-S platform. As for the best mapping (cf. Figure 8 (c)), the latency values range between 420 and 795 cycles for CEDAR-S, and between 50 and 210 cycles for CEDAR-H, resulting in an overall gain of 86%. A higher gain percentage is scored for spaced mapping, compared to localized and best mappings, since the DMC manages to reduce drastically the network contentions.

We conclude that the studied routing topologies cover a large scale of injection rate and latency values. CEDAR-S vs. CEDAR-H is a choice between area efficiency and performance.

5. Conclusion and Future Work

This article introduces SNet, a new routing architecture, and its DMC accelerator. Scalability, flexibility, and programmability are all features of this paradigm. SNet may be used to design any interconnection topology and can be applied to any routing topology. It has a hardware module called DMC that speeds up data transmissions. SNet's router offers a medium range area value when compared to state-of-the-art NoCs, while being totally configurable and customizable. This study uses SNet to run over 100 simulations. While performance was not the primary motivation

for this work, the results demonstrated that, despite its high flexibility and scalability, SNet is able to achieve high injection rates of up to 0.9 and low message latencies. We plan to apply the multitasking technique to SNet in the future. To maximize space usage, all cores will be able to dynamically switch between routing and computational tasks. This full dynamic method will be used in the development of an application.

References

[1] Itrs report of the year 2010. [Online]. Available: http://www.itrs.net/links/2010ITRS/Home2010.htm

[2] D. Bertozzi and L. Benini, "Xpipes: a network-on-chip architecture for gigascale systems-on- chip," Circuits and Systems Magazine, IEEE, no. 2, 2004, doi: 10.1109/MCAS.2004.1330747.

[3] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," ACM Comput. Surv., vol. 38, no. 1, p. 1, 2006, doi: 10.1145/1132952.1132953.

[4] Itrs report of the year 2001. [Online]. Available: http://www.itrs.net/Links/2001ITRS/Home.htm

[5] F. Moraes, N. Calazans, A. Mello, L. Moller and L. Ost, "Hermes: an infrastructure for low area overhead packet-switching networks on chip," Integration, the VLSI Journal, vol. 38, no. 1, pp. 69 – 93, 2004, doi: 10.1016/j.vlsi.2004.03.003.

[6] Itrs report of the year 1999. [Online]. Available: http://www.itrs.net/links/1999Winter

[7] E. Salminen, "Survey of network-on-chip proposals," white paper, OCP-IP, pp. 1 – 13, 2008.

[8] C. Azar, S. Chevobbe, Y. Lhuillier and J. Diguet, "SNet, a flexible, Scalable NETwork paradigm for manycore architectures," in 2013Seventh IEEE/ACM International Symposium On Networks- On-Chip (NOCS), Tempe, USA, 2013, pp. 1-2, doi: 10.1109/NoCS.2013.6558414.

[9] C. Azar, S. Chevobbe, Y. Lhuillier and J. Diguet, "Dynamic Routing Strategy for Embedded Distributed Architectures," in 2011 18th IEEE International Conference on Electronics, Circuits, and Systems, 2011, pp. 653-656, doi: 10.1109/ICECS.2011.6122359.

[10] K. Goossens, J. van Meebergen, A. Peeters and R. Wielage, "Networks on silicon: combining best-effort and guaranteed services," Proceedings 2002 Design, Automation, and Test in Europe Conference and Exhibition, 2002, pp. 423 – 425, doi: 10.1109/DATE.2002.998309.

[11] J. Liang, S. Swaminathan and R. Tessier, “ASOC: a scalable, single-chip communications architecture,” Proceedings 2000 International Conference on Parallel Architectures and Compilation Techniques, 2000, pp. 37 – 46, doi: 10.1109/PACT.2000.888329.

[12] C. Bobda, A. Ahmadinia, M. Majer, J. Teich, S. Fekete and J. van der Veen, “DyNoC: A dynamic infrastructure for communication in dynamically reconfigurable devices,” International Conference on Field Programmable Logic and Applications, 2005, 2005, pp. 153 – 158, doi: 10.1109/FPL.2005.1515715.

[13] M. Li, Q. Zeng and W. Jone, “DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip,” 2006 43rd ACM/IEEE Design Automation Conference, 2006, pp. 849 – 852, doi: 10.1109/DAC.2006.229242.

[14] T. Bjerregaard and J. Sparso, “A router architecture for connection-oriented service guarantees in the mango clockless network-on-chip,” Design, Automation and Test in Europe, 2005, pp. 1226 – 1261 Vol. 2, doi: 10.1109/DATE.2005.36.

[15] D. Siguenza-Tortosa, T. Ahonen and J. Nurmi, “Issues in the development of a practical NoC: the proteo concept,” the VLSI Journal, vol. 38, no. 1, pp. 95 – 105, 2004, doi: 10.1016/j.vlsi.2004.07.015.

[16] S. Sathe, D. Wiklund and D. Liu, “Design of a switching node (router) for on-chip networks,” ASIC, 2003. Proceedings. 5th International Conference on, 2003, pp. 75 – 78 Vol. 1, doi: 10.1109/ICASIC.2003.1277494.

[17] P. Guerrier and A. Greiner, “A generic architecture for on-chip packet-switched interconnections,” Proceedings Design, Automation and Test in Europe Conference and Exhibition 2000, 2000, pp. 250 – 256, doi: 10.1109/DATE.2000.840047.

[18] M. Ruaro, L. L. Caimi and F. G. Moraes, “A Systematic and Secure SDN Framework for NoC-Based Many-Cores,” IEEE Access, vol. 8, pp. 105997 – 106008, 2020, doi: 10.1109/ACCESS.2020.3000457.

[19] B. Grot, J. Hestness, S. Keckler and O. Mutlu, “A QoS-Enabled On-Die Interconnect Fabric for Kilo-Node Chips,” IEEE Micro, vol. 32, no. 3, pp. 17 – 35, May-June 2012, doi: 10.1109/MM.2012.18.

[20] J. Bainbridge and S. Furber, “Chain: a delay-insensitive chip area interconnect,” in IEEE Micro, vol. 22, no. 5, pp. 16 – 23, Sept. – Oct. 2002, doi: 10.1109/MM.2002.1044296.

[21] S. Salibekyan and P. Panfilov, “A new approach for distributed computing in embedded systems,” Procedia Engineering, vol. 100, no. 3, pp. 977 – 986, 2015, doi: 10.1016/j.proeng.2015.01.457.

[22] S. Evain, J. Diguet and D. Houzet, “NoC design flow for TDMA and QoS management in a GALS context,” EURASIP Journal on Embedded Systems, vol. 2006, no. 1, pp. 63653, 2006, doi: 10.1155/ES/2006/63656.

[23] E. Salminen, A. Kulmala and T.D. Hamalainen, “On the credibility of load-latency measurement of network-on-chips,” 2008 International Symposium on System-on-Chip, 2008, pp. 1 – 7, doi: 10.1109/ISSOC.2008.4694860.

[24] P. Pande, C. Grecu, M. Jones, A Ivanov and R. Saleh, “Performance evaluation and design trade- offs for network-on-chip interconnect architectures,” in IEEE Transactions on Computes, vol. 54, no. 8, pp. 1025 – 1040, Aug. 2005, doi: 10.1109/TC.2005.134.