RedNoCs: A Runtime Configurable Solution for Cluster-based and Multi-objective System Management in Networks-on-Chip

Philipp Gorski[1], Claas Cornelius[1], Dirk Timmermann[1], Volker Kühn[2]
Institute of Applied Microelectronics and Computer Engineering[1]
Institute of Communication Engineering[2]
University of Rostock
Rostock, Germany
{philipp.gorski2, claas.cornelius, dirk.timmermann, volker.kuehn}@uni-rostock.de

Abstract—**Runtime-based monitoring and adaptation are indispensable system management tasks for efficient operation of complex heterogeneous Multi-Processor-System-on-Chip (MPSoC) under dynamic workloads. Thereby, runtime-adaptive mechanisms like application mapping, adaptive routing or thermal management will have their own parameter, requirements and information flows. The main contribution of this work is the evaluation of a cluster-based, runtime-configurable and multi-objective system management strategy that combines the needed information flows of different runtime-mechanisms and supports the reuse of collected data for higher system-level services. This will increase the benefit-cost-ratio of needed hardware extensions and further tackles the system management aspect in a holistic way.**

Keywords-Network-on-Chip, MPSoC, Monitoring, System Control, HW/SW-Co-Design.

I. Introduction

Networks-on-Chip (NoC) emerged as the next generation of communication infrastructures for the growing number of computational on-chip resources in Multi-Processor-System-on-Chip (MPSoC) [1][2][3]. These complex systems will integrate functionality of various application domains at different regions on a single die. Each domain comes along with specific characteristics regarding the supported degree of parallelism (task-level and/or data-level), typical traffic pattern and loads, use cases, workload timing and constraints. Furthermore, some of these characteristics will change during system-lifetime, because underlying algorithms evolve or user scenarios will be adapted. The efficient operation of such heterogeneous systems depends on the integrated mechanisms for runtime management and their adaptability to the specific requirements of the covered application domains. Typical runtime tasks include application mapping and scheduling, debugging and test, power/energy/thermal management, traffic load management (e.g. adaptive routing in NoC) and fault-tolerance [4][5]. Thereby, the availability and needed quality of monitored system state information is a key concern. Furthermore, the transport of system control data to adjust the system behavior has to be right in time. Both, monitoring and control have their own requirements regarding the current workload, traffic loads and number of parameter needed to be observed/adjusted. Furthermore, the runtime control mechanisms have varying underlying algorithmic scopes (e.g. centralized, distributed or hybrid). This variability makes it hard to design a global solution for a system management, which include the monitoring as well as the control features in a holistic way. Most commonly publicized works present results about single runtime management mechanisms with specifically tailored solutions or utilize simulation-based workload information. Thus, they target a mono-objective management flow for a subset of runtime mechanisms (see Figure 1), which will be optimized for specific problem sizes, application domains or algorithms.

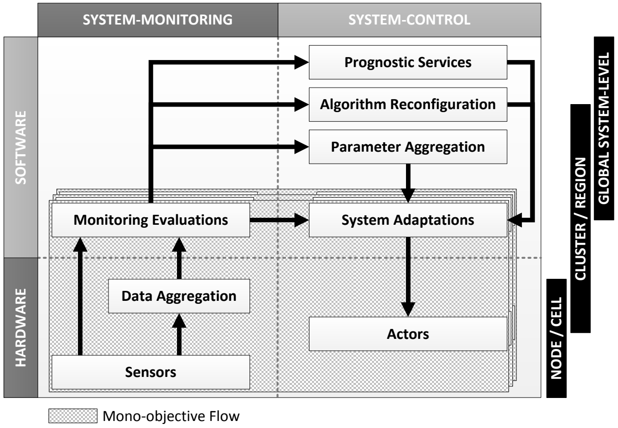


Figure 1. Overview of the multi-objective System Management Flow and its Organization in RedNoCs

The typical mono-objective flow follows the scheme illustrated in Figure 1. At the lowest level on the monitoring side sensors will capture specific parameter data (e.g. traffic or buffer loads for dynamic routing). The data will be further aggregated or fed directly into the unit responsible for monitoring evaluation, which can be implemented in software or hardware with different scopes (locally distributed or centralized). The results of the monitoring evaluation will be used by the system control mechanism to calculate needed adaptations (e.g. changed routing paths or tables) and sends them to its corresponding actors. The main contribution of this work targets the conceptual evaluation of combined information flows from different runtime mechanisms to support a multi-objective system management flow, and the global reuse of the captured data for higher system-level services. This will increase the benefit-cost-ratio of the additionally implemented functionality. Thereby, the presented solution follows four major design criteria. (1) *Redundant NoCs (RedNoCs):* The general separation of application and system data flows to different redundant infrastructure solutions (Data-NoC and System-NoC). Thereby, the design requirements will be separated too and each NoC can be optimized for its own traffic domain. (2) *Multi-objective Clustering:* The utilization of a software-directed and hardware-assisted clustering methodology, where cluster of different flows exist in parallel and can be reconfigured at runtime (sizing or timing of monitoring) to meet the specific constraints of assigned workload fractions. (3) *Reusability:* The proposed reuse of collected monitoring data for higher system-level services like the reconfiguration of system adaptation algorithms or prognostic services. Furthermore, the implementation of hardware intensive parts will focus their reuse for different domains. (4) *HW/SW-Co-Design:* The functional units of the system management, responsible for evaluation and adaptation, will be realized as software agents. Thus, these agents can migrate between resources and will be exchangeable at runtime. The needed hardware parts cover sensing, acting and extended network interfaces to decouple the cluster communication of different system mechanisms.

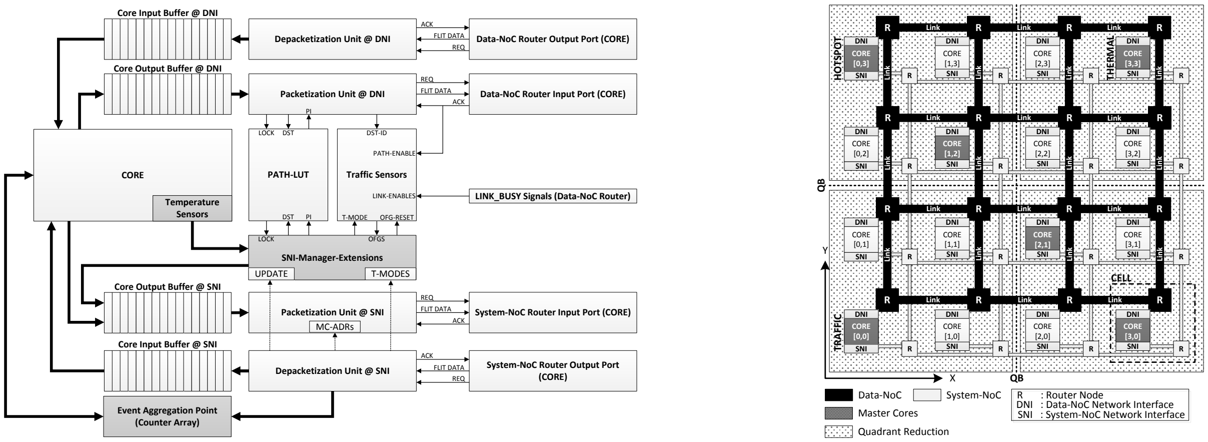
The remainder of this work is organized as follows. Section 2 covers the related work of existing approaches. The general concept and analysis of this work will be provided by Section 3. An evaluation of the experimental results for different corner cases will follow in Section 4. Afterwards, Section 5 gives a final conclusion and an outlook for future investigations.

II. Related Work

The majority of published works propose solutions for specific design cases. They can be mainly classified by the infrastructural integration concept, the operational scope/hierarchy and the targeted runtime mechanisms.

The infrastructural integration can be separated into two main directions [5][6] as follows: (1) Shared Infrastructure Solutions (SIS): The application data and the system information use the same NoC as communication infrastructure, which will be designed for the application data requirements. Additionally, to provide less concurrency between the different data domains special time-division-multiplexing (TDM) or the integration of prioritized virtual channels (VC) can be applied and each kind of data has its own reserved bandwidth or time slots. (2) Exclusive Infrastructure Solutions (EIS): The application data and the system information traffic are assigned to different communication infrastructures, which run in parallel. Each infrastructure is designed for the specific requirements of its data domain and there is no concurrency. Thus, a full separation of domain specific concerns at design- and runtime is achieved.

Different publications cover the integration aspect. In [8] Ciordas et. al. present a monitoring-centric evaluation of different EIS and SIS solutions for the *AEthereal* NoC regarding the design flow. Guang et. al. does a similar evaluation with the major focus on runtime monitoring and operational costs in [5][6]. Further, the requirements of different runtime mechanism and results (power, area, latency) for an 8x8 2D-Mesh topology NoC with 8-Bit data width per link direction are presented. These works show that EIS is the most promising way to handle monitoring traffic in NoCs. Similar results are shown by the infrastructure comparison in [9] for the MNoC using 24-Bit data width per link, four flit buffer depth and two virtual channels for the EIS. MNoC is applied for the objective of thermal/latency monitoring to realize a Dynamic Voltage and Frequency Scaling (DVFS). Adaptive routing in NoC represents the most common runtime mechanism for the utilization of mono-objective system management. The solutions differ from distributed to centralized traffic monitoring in combination with routing table/path adaptations. Ebrahimi et. al. presents two different distributed adaptive routing schemes (LEAR and CATRA) with EIS solutions for the aggregation of traffic congestion information. In LEAR [10] the neighboring router nodes share their congestion states via one additional wire per link direction in the range of one hop. A more complex and irregular congestion information aggregation network for the multi-hop regions is used by CATRA in [11], where the number of additional wires per link corresponds to the width/height of the 2D-Mesh topology. A solution between the complexity of LEAR and CATRA is presented by Rantala et. al. [12] using 2 additional wires per link direction for buffer level information sharing with direct-neighbor nodes. Similar regional congestion information aggregation EIS like CATRA can be found in RCA [13] and DBAR [14]. RCA uses 8 up to 16 additional wires per link, while DBAR will need 8 wires per link for the sharing of congestion information. All of these distributed adaptive routing mechanisms will evaluate the shared traffic information locally at each routing node using special hardware. A centralized adaptive routing called ATDOR is presented in [15]. The centralized path management works as additional hardware resource with a fixed coupling to the NoC though an out-of-band traffic monitoring aggregation network (EIS) using 4 additional wires per link. Further, for the distribution of path updates an additional EIS is suggested, but not applied. The global traffic information or the EIS won’t be reused. A complete multi-objective distributed system management with combined aspects of connection-oriented traffic monitoring, adaptive routing, and application mapping with clustering is tackled by the publications of Faruque et.al. [16][17].



(a) Overview of RedNoCs integration at the CORE

(b) 4x4 2D-Mesh NoC/Cluster with RedNoCs

Figure 2. Overview of the RedNoCs integration strategy at different abstraction levels

The supposed AdNoC solution integrates hierarchical software agents (global, cluster) for dynamic application mapping and reconfigurable clustering, distributed NoC traffic and application event monitoring via hardware probes at each resource/router, and distributed deterministic adaptive routing with buffer size reconfiguration. The monitoring traffic uses an SIS with a prioritized VC for this data domain. The complete system management is event-driven and focusses the runtime optimization of bandwidth utilization. An exploration of MPSoC monitoring and management systems is given in [18]. This work further introduces an own hierarchical concept of agent-based MPSoC management. Similar hierarchical approaches are applied by other publications. In [5][6] a hierarchical monitoring consisting of hardware nodes at the IP cores and at the cluster. Further, higher level software-based platform and application agents are proposed. In [19] this concept is applied for hierarchical power monitoring in NoC. Another approach considers the reuse for DVFS scenarios [20]. A event-based SIS monitoring concept including event taxonomy, probe design/programming, and variable hierarchy (distributed or centralized) for the *AEthereal* NoC is presented in [21]. The targeted use case scenarios are system- and application level debugging at runtime. This work is extended in [22] to support different abstraction-levels. Furthermore, the solutions in [23][24] covers the transaction-based debugging and the integration of special monitoring probes at design time.

In summary, most of the published works present good solutions for specific mono-objective flows, but they do not consider the synergistic potential of the flexible reuse of collected information or of the implemented hardware.

III. REDNOCS CONCEPT

RedNoCs is a HW/SW-based system management solution that covers the infrastructural separation of different traffic domains. The targeted NoC topology is a wormhole-switching 2D-Mesh with applied EIS, where two separated NoCs, called System-NoC and Data-NoC, work in parallel (see Figure 2 (b)). The links between neighboring routers are bidirectional point-to-point connections, which transmit a certain portion of a communication packet in parallel (called flit). The first flit of a packet (header) contains the routing information, while following flits carry the payload. Each packet will finalize with a tail flit that transports payload data too. A packet will enter the NoC at the source node flit-by-flit (wormhole-switching), passes the intermediary router nodes (hops) and finally reaches its destination, where the contained payload data will be processed. The Data-NoC covers the transport of application data, while the System-NoC is used for system management. The System-NoC has the same topology as the Data-NoC to provide the same resource-connectivity, but both infrastructures can be individually adapted to the domain-specific requirements. The Data-NoC integrates more complex routing algorithms, link data width (64-Bit or higher) and resource functionality, while the System-NoC works with reduced link data width (5- or 7-Bit), minimal input buffering (one flit) and dimension-ordered XY/YX-Routing. As illustrated in Figure 2 each NoC-Resource (CORE) is connected independently to both NoCs via Data-Network- and System-Network-Interface (DNI and SNI). The smallest management unit of RedNoCs is a CELL and includes the CORE, DNI, SNI and the connected router nodes (R) of both NoCs. These CELLs are further sub-classified into Slaves and Master. A Master-CELL is suited with special hardware resources and software agents to manage a CLUSTER or global system-level operations. Slave-CELLs are dynamically grouped into a CLUSTER by a corresponding Master-CELL. These CLUSTERs are the fundamental components for the system management and can be configured individually. The following sub-sections describe the specific details of the RedNoCs functionality and its organization.

A. Clustering

The implemented clustering is reconfigurable at runtime and context-based. The creation and management of CLUSTERs is realized via software agents at the Master-

CELLs and utilizes a messaging/organization concept as follows:

CLUSTER-REQUEST (CREQ): For the initial creation of a CLUSTER the Master-CELL sends allocation packets to all CELLs that need to be part of the CLUSTER. These packets consist of the destination routing information (CELL-ADR), the NoC-Address of Master-CELL as source information (MC-ADR), the context identifier of the CLUSTER (CTX-ID) and further context data for Slave-CELL configuration (CTX-DATA). The list of CELLs that will be grouped to a CLUSTER and the corresponding Master-CELL will be selected by global domain agents (e.g. application mapping agent).

CREQ = {CELL-ADR | MC-ADR | CTX-ID | CTX-DATA}

CLUSTER-ACKNOWLEDGE (CACK): The Slave-CELLs receive the request/update of the Master and returns a binding packet as acknowledgement. The packet contains the MC-ADR as routing header, the CELL-ADR as source information and the special CTX-ID to classify the packet.

CACK = {MC-ADR | CELL-ADR | CTX-ID}

CLUSTER-UPDATE (CUP): During CLUSTER operation the configuration data (monitoring periods, routing path updates) need to be adapted, the software agent migrates to another Master-CELL or the CLUSTER will be deleted. Thus, the Slave-CELLs are informed via update packets, which have the same format like the CREQ.

CUP = {CELL-ADR | MC-ADR | CTX-ID | CTX-DATA}

The context of a CLUSTER describes the system management domain (e.g. traffic or thermal monitoring/control) it is used for. Each Slave-CELL can be assigned to multiple CLUSTERs of different CTX-IDs at the same time, but not to different CLUSTERs of the same CTX-ID. Thus, if multiple CLUSTERs of the same CTX-ID coexist, they will be spatial separated and do not share any CELLs. Moreover, this separation concerns the exclusive clustering for different workload fractions/applications and avoids interferences. Each clustering context has its own configuration data, parameter and identifier. Domains that are integrated at design time and supported by special hardware resources (e.g SNI-Manager-Extensions at Figure 2 (a)) have a reserved set of CTX-IDs assigned. The rest of freely available CTX-IDs can be used for software-driven and dynamically defined services (e.g. execution monitoring of individual applications). The allocation of CTX-IDs is managed at runtime by a global domain agent. Before a new service is allowed to start a CTX-ID must be requested by its Master-CELL agent. Furthermore, after finalization of the CLUSTER service the global agent must be informed that the reserved CTX-ID is freed. At this state of research, the restriction of the CTX-ID to one byte seems sufficient and allows the integration of 256 clustering domains. While the CLUSTER will be created and managed by the cluster agent, the planning, resource assignment and placement of CLUSTERs will be processed by upper-level global software agents, which are responsible for specific domains. Furthermore, domain-independent global agents will assembly and process global cross-domain parameter. Below, Figure 3 illustrates the global dataflow concept and separation of concerns for the agent/data architecture of RedNoCs. This concept targets a hierarchical problem separation, where the global scope for parameter adjustments, algorithms and adaptation policies will be configured by the global agents and the cluster agents will work on these globally configured data sets to manage the assigned workload fractions and regional configurations.

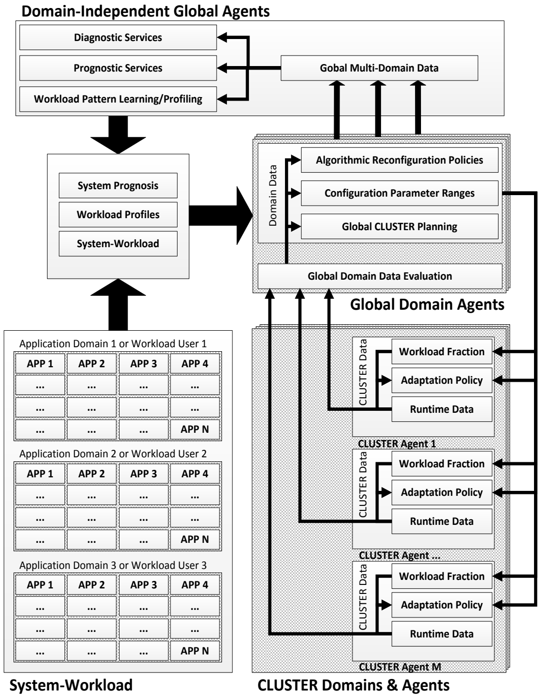


Figure 3. Global software-agent hierarchy in RedNoCs

B. *System-NoC Design and Optimization*

The System-NoC necessitates the integration of additional hardware resources. The System-NoC topology is fully redundant to the Data-NoC, because the support of multi-objective flows needs a general connectivity that covers the potential resource interactions. This design strategy will increase the benefit-cost-ratio. As entry-point for the dimensioning of the System-NoC the minimal data width for the link is adjusted. Furthermore, this parameter defines the number of parallel transmitted data-bits per flit, the sizing of the router node and the number of wires per link direction. In the utilized XY/YX-Routing the header flit contains the complete position [x, y] of the destination CELL as bit-vector and one additional bit for the path selection (XY | YX). Inside a $N_X \times N_Y$ 2D-Mesh NoC the number of addressable CELLs is given by (1) as well as the number of CELLs in a rectangular CLUSTER *Ci*.

$$N^{noc}_{CELLS} = N_X \cdot N_Y \;\; ; \;\; N^{Ci}_{CELLS} = N^{i}_{X} \cdot N^{i}_{Y} \leq N^{C\max}_{CELLS} \qquad (1)$$

Thus, the minimal link data width in bit needs to be at least $ldw_{\min}$ as formulated in (2).

$$ldw_{\min} = \log_2\left(N_{CELLS}^{noc}\right) + 1 \qquad (2)$$

Additional wiring per link direction will be needed by the hop-based REQ/ACK flow control (2-bit/wires) and the flit identifier (2-bit to mark as header '11', payload '01' or tail '00'). Thus, the final link width per direction as number of bit/wires can be obtained from (3).

$$lw_{\min} = ldw_{\min} + 4 \qquad (3)$$

Exemplary, for a 4×4/8×8/16×16 NoC the minimal link data width would be 5/7/9-bit per flit and per link direction 9/11/13 additional wires are needed. The XY/YX-Routing was chosen to support a balanced link utilization over system lifetime and the XY or YX path option will be toggled globally if the System-NoC has no load (e.g. power-on state). The System-NoC packets consist of static and context-based data. The static parts contain the routing information and context identification {DST|SRC|CTX-ID}. The header flit carries the packet destination (DST) and the second flit transports the source node address (SRC) of the packet. These flits will be followed by the context identifier, which needs 8-bit. The additional bits needed for the context data ($n_{CTX-DATA}$) will complete the packet to its final length (4) (in number of flit). The optimal per-hop-latency of the packet header (without congestions) for the targeted System-NoC design is 3 clock cycles (REQ/ACK-handshaking and routing/arbitration delay).

$$l_{packet} = \left\lceil \frac{2 \cdot ldw_{\min} + n_{CTX-DATA} + 8}{ldw} \right\rceil \; ; \; ldw \geq ldw_{\min} \qquad (4)$$

A resource/IP core allows the consumption of one flit in a period of two clock cycles (REQ/ACK) through its network interface. Generally, we can define the reception rate (RR) per IP core as the ratio of received flits to the observed time period in clock cycles and the injection rate (IR) as the ratio of injected flits to the observed time period in clock cycles (see eq. (5)).

$$RR = \frac{sum_{flit}^{in}}{sum_{clock\ cycles}^{period}} \leq 0.5 \; ; \; IR = \frac{sum_{flit}^{out}}{sum_{clock\ cycles}^{period}} \leq 0.5 \qquad (5)$$

The typical traffic pattern inside the System-NoC and its CLUSTERs partially varies from those inside the Data-NoC. For centralized managed CLUSTER the traffic flow will be N_S:1 and 1: N_S. The N_S:1 pattern implies that the Master-CELL of a CLUSTER is the hotspot and all Slave-CELLs (N_S) transmit data to it. This pattern is typical for monitoring tasks. On the other hand, the Master-CELL needs to reconfigure the Slave-CELLs or other runtime mechanisms, which comply with the 1: N_S traffic pattern. Another traffic category is the non-centralized/distributed one from Node-to-Node (N2N). This fraction of System-NoC traffic typically correlates with the traffic pattern of the Data-NoC and concerns distributed mechanisms like flow-control or error management. The standard configuration of the System-NoC, implementing the above mentioned requirements, is called FULL. To optimize the System-NoC design regarding area consumption and/or data-throughput three major strategies can be applied as follows:

QUADRANT REDUCTION: The additional wiring of the System-NoC and the router area are mainly defined by the $ldw_{\min}$ parameter. Thus, a smaller $ldw_{\min}$ would reduce the additional hardware costs, but also increases the needed packet length (4). Resizing $ldw_{\min}$ can be realized via address-space reduction. The address-space-reduction targets the minimization of the coordinate data width. Therefore, the NoC will be divided into quadrants (see Figure 2 (b)) and inside a quadrant each CELL will be addressable directly via a single header flit. If the communication range of a packet exceeds a quadrant, the routing information consists of intermediate header flits ('10') and a final header flit ('11'). The intermediate header flits are needed to pass quadrants that does not contain the final packet destination. Thus, the routing path will be segmented and each time the packet crosses a quadrant border (QB in Figure 2 (b)) the corresponding intermediate header flit can be dropped. The link data width can be reduced by 2-bit (see eq. (6)) and the additional delay correlates to the number of intermediate header (≤ 2).

$$ldw_{\min} = \log_2\left(N_{CELLS}^{quadrante}\right) + 1 \; ; \; N_{CELLS}^{quadrante} = \frac{N_{CELLS}^{noc}}{4} \qquad (6)$$

The System-NoC configuration with applied quadrant reduction is called REDUCED.

DUAL-PORTED MASTER: The N_S:1 traffic pattern inside a CLUSTER results in a bottleneck at the SNI of the Master-CELL, because the RR of the SNI limits the number of packets that can be injected by the Slave-CELLs. If the number of injected packets exceeds the number of receivable packets the System-NoC becomes congested. Thus, the formulated condition of (7) restricts the maximal monitoring traffic. The maximal reception rate of the Master-CELL can be doubled by the integration of a second port at the router that is connected to the SNI. Incoming packets at the router node will be randomly assigned to the first or the second core-port. The additional hardware is restricted to the number of potential Master-CELLS (≤ 50% of all CELLs).

$$RR_{Master}^{\max} \geq \sum_{i=1}^{N_{CELLS}^{Ci}} IR_{Slave\ i}^{\max} \qquad (7)$$

PATTERN SEPARATION: The interferences of concurrent traffic pattern can be reduced by the integration of two virtual channels (VC) [1][2][3]. Thus, the N_S:1 and 1:N_S pattern will share one VC, while the N2N pattern utilizes a different VC. The concurrency of coexisting N_S:1 patterns depends on the placement of the Master-CELLs. If they are nearby to each other the interference of traffic flows will be high. This can be circumvented by symmetric placements as given in the 4x4 CLUSTER example of Figure 2 (b). The traffic monitor (TRAFFIC) is mapped to the Master-CELL [0,0], while the thermal monitor (THERMAL) runs on Master-CELL [3,3]. Regarding the bidirectional links, both traffic flows from all Slave-CELLs to these two hotspots will be independent. Thus, the resulting minimal interference allows the sharing of a single VC. N2N pattern will be more variable in timing and the distribution of source-destination-pairings. Hence, this traffic load will be assigned to the second VC.

C. Monitoring

The availability of current system state information is indispensable for runtime-based application mapping [25][26][27][28][29], power management [6][9][30] and adaptive routing [11][31][32][33][34] in communication-centric complex MPSoC. Therefore, RedNoCs integrates runtime configurable and cluster-based solutions for thermal and traffic monitoring. The dynamic clustering enforces full application/regional isolation [11][14][32][33]. Thus, the monitored data is associated exclusively to the workload fraction inside the CLUSTER and the sampling periods/timing can be configured at runtime under consideration of the current load.

1) Traffic Monitoring

The traffic monitoring of RedNoCs integrates a periodic and centralized mechanism that is hierarchical organized at three different levels (PATH/LINK, CELL, and CLUSTER).

PATH/LINK-LEVEL: The basic traffic sensor is a simple combination of an external triggered binary counter and a configurable comparator (see Figure 4). The counter increments each clock cycle the ENABLE signal is active. In parallel the comparator checks the current counter value against a reference value that is set by the T-MODE. The supported T-MODEs of the presented RedNoCs traffic monitoring can be obtained from Figure 4. If the counter value reaches the configured T-MODE reference, it sets an overflow flag (OFG) that is captured by the register R and external resettable. This unified solution is used in two different ways: (1) *LINK LOAD:* Each output port of a Data-NoC routing node (e.g. North, East, South, West, and Core at 2D-Mesh topology) is connected to a traffic sensor to measure the current link load (LL). The ENABLE signal is connected to the status signal of the port output arbitration unit. The total number of traffic sensors will be 5 per CELL. (2) *INJECTION RATE:* Selected path table entries (DST-ID) of the DNI at each CELL gets a traffic sensor assigned to cover the injection rates at the path level. The assignment of DST-ID to specific application tasks is performed by the mapping algorithm and will be set during the cluster creation. Furthermore, one traffic sensor captures the overall injected traffic. The ENABLE input is connected to the acknowledgment signal (ACK) of the DNI output. The number of needed traffic sensors depends on the maximum sizing of a CLUSTER the monitoring should work path-accurate for. At the current progress, RedNoCs works with 16 path sensors (e.g. 4x4 or 8x2 CLUSTER).

All traffic sensors of a CELL run at the same T-MODE, which is set at the SNI-Manager-Extension by the Master of the corresponding traffic monitoring CLUSTER (Figure 2 (a)). Furthermore, they are grouped and located at the SNI of the CELL (see Figure 5).

CELL-LEVEL: The OFG registers of all traffic sensors inside a CELL are connected to the SNI-Manager-Extension responsible for the RedNoCs Traffic Monitoring (see Figure 5). This functional unit generates the traffic monitoring packets for the System-NoC and works periodically. Thereby, the period (TIMER) is set by the T-MODE value (same as for traffic sensors) of the CELL. For a Data-NoC running on 1 GHz, the traffic situation for each CELL is sampled in intervals configurable from 64 up to 2048 ns. After the expiration of a period, the finite state machine (FSM) tests if at least one OFG is set (OFG-CHECK) all register will be read out and reset at the traffic sensors. If no OFG is active there is no need to generate a traffic monitoring event packet for the expired period. Otherwise the FSM generates a new packet with a defined static order of the OFG-bits (CTX-DATA). The packet destination is the Master-CELL of the corresponding traffic monitoring CLUSTER. Afterwards, the packet is pushed to the output buffer at the SNI of the CELL.

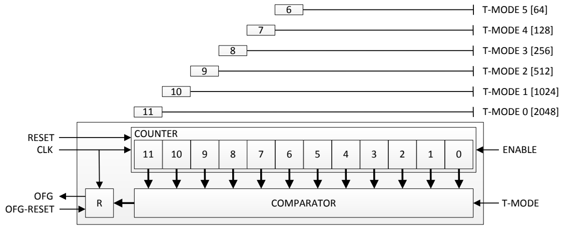


Figure 4. Basic runtime configurable Traffic Sensor of RedNoCs

CLUSTER-LEVEL: At this point, the traffic monitoring packets periodically leaves the CELLs and need to be aggregated by the Master-CELL, after they have passed the System-NoC towards it. Therefore, special Event Aggregation Points (EAPs) are present as exclusive hardware at all Master-CELLs (Figure 6 and Figure 2 (a)). These EAPS are needed to scale the generated OFG data to the final parameter of injection rate (IR) and link load (LL). IR as well as LL will be mapped to scales from 0 up to 100 percent with k_S percentage stepping. Thus, the aggregation for the events of $100/k_S$ traffic monitoring periods is needed. Each period event of a traffic sensor with a reported OFG of ‘1’ represents k_S scale percent of IR or LL. This is done using grouped binary 7-bit counters, where each group is assigned to a monitored CELL of the CLUSTER and each counter inside a group is assigned to the OFG of a specific traffic sensor of this CELL. The counters are triggered by the incoming OFG-DATA and are incremented by one if the corresponding OFG-Bit is ‘1’. The OFG-DATA is fed as fix-ordered parallel bit-vector into the counter group, where the index of each bit corresponds to the traffic sensor identifier. The groups are addressed by the GROUP ID, which is equal to the CELL-ID. The EAP has a buffer at the input and can process the complete OFG-DATA of a traffic

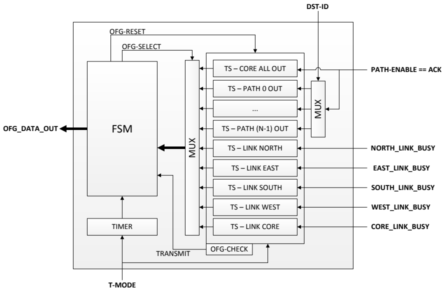


Figure 5. SNI-Manager-Extension for RedNoCs traffic monitoring

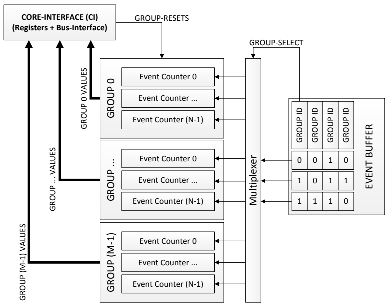


Figure 6. Functional blocks of an Event Aggregation Point

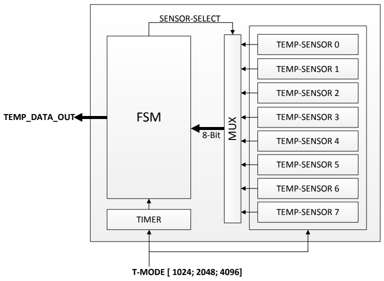


Figure 7. SNI-Manager-Extension for RedNoCs thermal monitoring

monitoring packet in one clock cycle. At the current status of research, a maximal CLUSTER size of 16 CELLS with 21 traffic sensors (5 links/15 paths/1 overall) per CELL was applied. This results in 16 groups with 21 binary 7-Bit counters at each. Moreover, the EAP represents the HW/SW-Interface of the traffic monitoring and the final traffic data can be accessed through the CORE-INTERFACE (CI) that is directly coupled to the internal bus of the Master. The counter values are captured by registers at the CI and the cluster agent will access and store them after a monitoring period has finished or during a current period. The duration of a period can be calculated by eq. (8) and depends on the configured T-MODE period ($r_{T\text{-}MODE}$), the clock frequency of the System-NoC ($f_{System\text{-}NoC}$) and the scale resolution k_S (e.g. k_S =1% or 2%).

$$t_{period}^{Ci} = \frac{r_{T-MODE}}{f_{System-NoC} \cdot k_S} \qquad (8)$$

In example, for a T-MODE of 256 clock cycles at 1GHz the complete path accurate traffic situation of the CLUSTER *Ci* can be capture in periods of 25,6 µs (k_S =1%) or 12,8 µs (k_S =2%). Afterwards, the counter needs to be reset for the next period. Furthermore, the variation of the traffic situation can be recorded by intermediate snapshots during a period without reset. The EAP and the CI are key components to achieve a light-weighted software agent, because the agent is operating on final parameter values and does not need to perform further aggregation steps. Regarding the CLUSTER sizing under consideration of a specified T-MODE the formula of (9) can be deduced from (7) and (4). Thereby, the interference factor k_i reduces the allowed cluster sizing depending on the traffic situation in the System-NoC.

$$N_{CELL}^{Ci\ \max} \leq \left\lfloor \frac{RR_{master}^{\max} \cdot l_{packet}}{r_{T-MODE}} \cdot k_i \right\rfloor \quad with \quad 0 < k_i \leq 1 \qquad (9)$$

2) *Thermal Monitoring*

The integrated thermal monitoring of RedNoCs is a modification of the solution presented in [9]. Each CELL is suited with 8 temperature sensors that offer measured temperature as 8-bit data values. Thus, 64-bit CTX-DATA will be reported by each thermal monitoring packet. These sensors are connected to a special SNI-Management-Extension (see Figure 7) and will be read out periodically (configured by T-MODE). The T-MODE controls a TIMER and RedNoCs works will periods of 1024, 2048 and 4096 clock cycles to enable different temperature resolutions. If the current period is finished, the FSM reads out all temperature sensors (cycle by cycle), generates the monitoring packet and sends it through the System-NoC towards the Master-CELL of the CLUSTER. The software agent at the Master-CELL is able to consume the temperature data directly without further aggregation. The software agent collects these data until all temperature sensors reported its values for the current observation cycle. Afterwards, the new observation cycle starts. The maximal CLUSTER sizing can be calculated by (9).

D. *System Control*

The system control features of the RedNoCs solution targets the utilization of the collected monitoring information at different system-levels (outlined in Figure 3) as well as the reuse of the EAPs. The workload-specific collection of thermal and traffic load data enables the integration of: (1) *Runtime-based selection of dedicated routing algorithms* [35][36] *and/or application mapping strategies* [25] *for different workloads and optimization scenarios* (Algorithm Reconfiguration/Adapdation at Figure 1/3). Moreover, depending on the current and historic workload, the optimization objectives can vary between wear-out minimization, performance/delay or energy consumption (Diagnostic/Prognostic Services at Figure 1/3). Thus, different parameter will be aggregated and combined with dedicated strategy selection policies. (2) *Active Workload/Traffic-Pattern learning at runtime*. This reduces the need of preliminary offline-profiling and data collection as used in [35][36][25]. The attributes of nodes and edges in an application task graphs (like computational costs and traffic injection rates) and their final mappings [4] will be collected/refined during system operation.

Another important system management feature is the observation of task execution [35] to measure application performance or register erroneous interrupts. Therefore, the unused EAPs at the Master-CELLs will be reused. Selected tasks generate events or different types of events, when they process their computation on the IP cores. These events are encoded as special bit-vectors to trigger dedicated counter of an EAP counter group (similar to traffic monitoring). The GROUP-ID and counter assignment is performed by the

TABLE I. SIMULATION RESULTS FOR REDNOCS CORNER-CASES WITH BEST ACHIEVABLE PARAMETER CONFIGURATIONS (APL: # OF CLOCK CYCLES ; MANAGER: # OF CLOCK CYCLES ; CBW: MBIT/S)

		Parameter Configuration				Average Packet Latencies (APL)									
		N2N IR		TSP		TRAFFIC		TEMP		N2N		MANAGER		CBW	
Design	**Pattern**	*4x4*	*8x2*	*4x4*	*8x2*	*4x4*	*8x2*	*4x4*	*8x2*	*4x4*	*8x2*	*4x4*	*8x2*	*4x4*	*8x2*
FULL	hotspot	0.025	0.025	256	512	109.2	117.1	234.1	233.9	122.9	84.9	910	1556	357.4	275.3
	uniform	0.025	0.05	256	512	112.6	113.9	238.3	236.5	85.6	91.3	629	913	357.4	511.2
	bit comp	0.025	0.05	256	512	111.1	114.9	237.2	236.8	79.5	93.9	640	980	357.4	511.2
	transpose	0.025	0.05	256	512	110.1	115.3	237.2	237.2	76.4	95.9	633	935	357.4	511.2
FULL DP	**hotspot**	**0.025**	**0.025**	**256**	**512**	**89.1**	**83.4**	**167.7**	**133.1**	**69.5**	**45.9**	**1000**	**1003**	**357.4**	**275.3**
	uniform	**0.05**	**0.025**	**256**	**256**	**93.8**	**74.7**	**172.2**	**140.4**	**69.4**	**41.8**	**576**	**795**	**593.2**	**357.4**
	bit comp	**0.05**	**0.025**	**256**	**256**	**92.1**	**74.2**	**171.8**	**140.1**	**60.3**	**40.9**	**610**	**767**	**593.2**	**357.4**
	transpose	**0.05**	**0.025**	**256**	**256**	**92.6**	**73.6**	**173.7**	**140.5**	**66.8**	**40.8**	**610**	**779**	**593.2**	**357.4**
FULL 2 VC	hotspot	0.025	0.025	256	512	119.5	127.9	235.1	234.1	**62.9**	**53.4**	668	544	357.4	275.3
	uniform	0.025	0.05	256	512	113.9	120.9	260.7	297.5	**23.7**	**30.1**	658	577	357.4	511.2
	bit comp	0.025	0.05	256	512	114.1	121.5	260.3	296.4	**23.9**	**29.9**	683	589	357.4	511.2
	transpose	0.025	0.05	256	512	115.6	118.9	259.8	297.9	**24.2**	**29.9**	688	578	357.4	511.2
REDUCED DP	hotspot	0.025	0.025	512	512	102.7	94.1	209.3	164.1	66.7	56.7	785	1100	209.1	209.1
	uniform	0.05	0.05	512	512	106.8	89.7	214.2	185.4	57.5	55.6	662	918	373.1	373.1
	bit comp	0.05	0.05	512	512	105.6	89.3	214.3	184.8	57.3	56.6	695	1262	373.1	373.1
	transpose	0.05	0.05	512	512	105.9	90.4	214.9	184.1	58.1	56.3	694	1183	373.1	373.1

software agent responsible for the execution monitoring. After the event generation a packet towards the EAP is send out and the EAP counters will be adjusted according to the event data. The software agent regularly captures the counter values and controls the operation progress of the workload. Thus, if tasks do not generate events a problem might be occurred or the performance is not as expected. Further, measuring the progress in periodic intervals allows the evaluation of application performance and gives additional feedback to the selection strategies of routing and mapping algorithms.

IV. EXPERIMENTAL RESULTS

The evaluation of the RedNoCs solution was realized via system simulations for operational performance and hardware synthesis for the cost approximation. Thereby, the basic System-NoC design parameter configuration can be obtained from TABLE II.

TABLE II. SYSTEM-NOC CONFIGURATION FOR SIMULATION AND SYNTHESIS

Parameter	Value
Clock Rate	1 ns
Topology	2D-Mesh
NoC-Size	8x8
Cluster Size	4x4, 8x2
Input Buffer Depth	1 Flit
# of Master-CELLs	32 (=50%)
Traffic Sensors per CELL	21
7-Bit Counter per EAP	336
N2N Injection Rates	0.025 up to 0.05
N2N Traffic Pattern	random uniform distributed, transpose, hotspot (H=20%) and bit complement

A. System Simulations

The first evaluation step was the corner-case simulations of the maximum CLUSTER size (16 CELLs) at different shapes (4x4 and 8x2) and workloads. Therefore, an own cycle accurate SystemC/TLM-based simulator was used. For different CLUSTER shapes and System-NoC designs the simulated workloads contained a traffic monitoring cluster, thermal monitoring cluster and synthetic traffic patterns for the N2N traffic component. Those three CLUSTERs ran in parallel with full spatial coverage. The placement of the Master-CELL for thermal monitoring was the upper right corner and the Master-CELL of the traffic monitoring was assigned to the lower left corner (see Figure 2 (b)). The configured T-MODE for the thermal monitoring period (TMP) was fixed to 2048 clock cycles. This results in a 30% higher sampling rate than used in the reference of [9].

The T-MODE for the traffic sampling period (TSP) was varied between 256 and 1024 clock cycles. Furthermore, the worst-case of monitoring packet injections per CELL at each sample period was simulated (without the OFG-CHECK). The N2N traffic was simulated under consideration of the random uniform, bit complement, transpose and hotspot pattern [1][2][3]. Thereby, the injection rate (N2N IR) was varied between 0.025 and 0.05 flit/clock cycle per CELL. For the simulated designs these two cases imply that each CELL generates packets with 1 up to 4 byte CTX-DATA in average intervals of 200 and 100 ns. This is more than sufficient for N2N-based transaction- and/or connection management [18][24]. The hotspot pattern furthermore covers the EAP reuse scenario for task observation and performance measurements. The HOTSPOT Master-CELL was placed in the upper left corner of the CLUSTER (see Figure 2 (b)) and receives a 20% fraction of the total injected N2N traffic of all CELLs inside the CLUSTER. Thus, each CELL will generate task events with an average interval of 1 µs or 0.5 µs. This worst-case assumes that all CELLs of the NoC will be active computational nodes running tasks with the periodicity of 0.5-1 µs. For the estimation of the communicational delay if the cluster agent migrates from one Master-CELL to another the cluster agents transmitted *CUP* packets to all Slave-CELL in intervals of 200 µs and the maximum packet latency (in # of clock cycles) over this complete procedure was captured (MANAGER). For the other monitoring (TRAFFIC and TEMP) and N2N traffic loads the average packet delay (in # of clock cycles) was recorded. Thereby, the packet delay is measured as timing interval from the transmission buffer injection of the header flit at the source CELL to the final consumption of the tail flit at the destination CELL. The simulation results for different System-NoC designs are summarized in TABLE I. The listed parameter configurations represent traffic scenarios, where each CLUSTER achieves its timing constraints (packets arrives Master-CELL inside the adjusted sample period) and the highest achievable TSP was focused, because the information about generated and transmitted data per CELL/path/link has the highest weight as activity indicators for traffic, performance and energy management. The resulting average bandwidth utilization per CELL (CBW) per CELL for the System-NoC traffic is given in Mbit/s. The results cover the average of 100 simulation runs per parameter configuration with a system operation time of 1 second at each run. They show that the dual-ported (DP) optimization strategy for the FULL System-NoC configuration performs best in all simulated traffic cases. The average packet latency improvement of the FULL DP over the FULL design case is 23.2% at the 4x4 and 42.9% at the 8x2 CLUSTER shape, while the additional hardware overhead scales with the number of Master-CELLs. The evaluation of the pattern separation optimization (2 VC) proves the better performance of the N2N traffic, if it has its own VC (average latency improvement ~62.5%). But the latencies of the remaining traffic domains will increase because of the bandwidth reduction introduced with the utilization of VCs and the additional hardware costs depends on the number CELLs. Furthermore, the simulations showed up that the packets of the traffic monitoring domain were the first which ran out of their latency constraints and thus the most vulnerable regarding interferences.

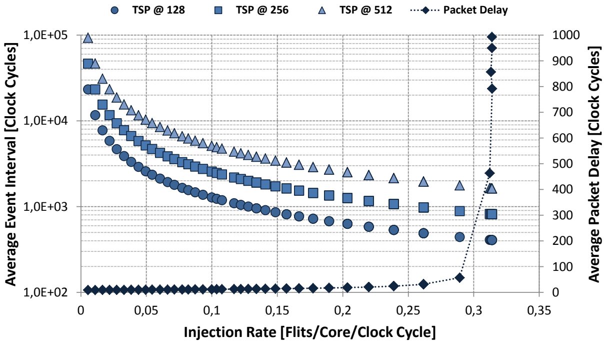


Figure 8. Real traffic monitoring packet injection intervals for varying Data-NoC traffic loads and active OFG-CHECK

To evaluate the traffic reduction caused by the OFG-CHECK a complete 8x8 Data-NoC was simulated under different traffic loads and patterns. The adjusted T-MODEs were 128, 256 and 512 clock cycles. The diagram of Figure 8 shows that the average event interval for the monitoring packet generation by the CELLs is at least three times greater than the adjusted period. In the unsaturated operational region of the Data-NoC (IR < 0.3) the difference is even higher. Thus, the results of TABLE I correlate to the worst-case and demonstrate that real system operation metric will become even better. For the REDUCED design case only the Dual-Ported results are presented in TABLE I. The increased packet lengths, regarding the lower link data width, of the REDUCED designs influence the traffic situation and omit to reach the same performance as observed for the FULL design cases. Thereby, the degradation in achievable sample rates for monitoring will be even higher than the minimization of the System-NoC hardware overhead. The captured overall delay for MANAGER communication of the cluster agents show that reconfiguration or changes of the CLUSTER will take effect after hundreds of clock cycles. In the observed case the needed time would be approximately 0.5 µs up to 1 µs, which is low enough to provide dynamical adaptations for workloads that may change/vary in the order of hundreds of µs up to a few ms.

B. *Hardware Synthesis*

The ASIC design flow was realized with the Synopsys^{TM} DesignCompiler^{TM} using the 45 nm Nangate FreePDK45 Generic Open Cell Library. The presented results in TABLE III show the total cell area (TCA) costs for each of the functional RedNoCs components (SNI-Manager Extensions, Traffic Sensors, EAP) for REDUCED and FULL design.

TABLE III. TOTAL CELL AREA (TCA) HARDWARE COSTS FOR FUNCTIONAL REDNOCS COMPONENTS INSIDE AN 8X8 NOC AT ALL

	TCA [8x8 NoC] [mm²]	
Design Component	*REDUCED*	*FULL*
SNI TEMPERATURE EXT.	0.05311488	0.05452736
SNI TRAFFIC LOAD EXT.	0.06155904	0.09676416
TRAFFIC SENSORS	0.17410176	0.17410176
AGGREGATION POINT	0.71138176	0.71138176
SUM OF ALL UNITS	**1.00015744**	**1.03677504**

TABLE IV contains the hardware costs for the System-NoC routers (TCA) and the complete RedNoCs designs (TCA ALL). The targeted operational frequency was set to 1 GHz and met for all evaluated design cases.

TABLE IV.TOTAL CELL AREA (TCA) HARDWARE COSTS FOR ROUTER UNITS OF AN 8X8 SYSTEM-NOC AND ALL UNITS OF REDNOCS (TCA ALL)

Design	TCA [mm²]	TCA ALL [mm²]	Linkwidth *lw*
FULL	0.17815616	1.2149312	11
FULL DP	**0.19422688**	**1.2310019**	**11**
FULL 2 VC	0.39875328	1.4355283	11
REDUCED DP	0.18442976	1.1845872	9

The main hardware overhead of RedNoCs belongs to the EAPs (>50%) and relativizes the potential savings of the REDUCED design (~3.8% compared to FULL DP). These costs can be reduced if the number of Master-CELLs decreases. Furthermore, the comparison of FULL DP against the FULL design proves the benefits of this optimization strategy. The hardware costs will be only ~1.3% higher, while the operational performance improves by ~23.2% at least. Regarding the hardware costs in the context of the final MPSoC that contain the targeted amount of CELLs on a 45nm silicon die (areas: 280-400 mm² [36][37]) the relative overhead due to RedNoCs will be less than ~2%.

V. CONCLUSION AND FUTURE WORK

The evaluation results show that the presented RedNoCs concept is applicable and supports the integration of multi-objective system management flows at runtime under consideration of affordable costs. Furthermore, the dual-ported optimization strategy was purposed and showed up good performance improvements. The next steps of future investigations target the full integration of adaptive routing and application mapping mechanisms in combination with RedNoCs. Especially, the runtime-based workload pattern learning, prognostic services for long-term reliability improvements, and the scalability analysis of the software agents will be evaluated for scenarios of different application domains.

REFERENCES

[1] E. Salminen, A. Kulmala, and T. D. Hämäläinen, “Survey of Network-on-chip Proposals,” *WHITE PAPER, OCP-IP, MARCH*, no. March, pp. 1–12, 2008.

[2] A. Agarwal, C. Iskander, H. T. Multisystems, and R. Shankar, “Survey of Network on Chip (NoC) Architectures and Contributions,” *scientificjournals.org*, vol. 3, no. 1, 2009.

[3] T. Bjerregaard and S. Mahadevan, “A survey of research and practices of network-on-chip,” *ACM Computing Surveys (CSUR)*, vol. 38, no. 1, p. 1, 2006.

[4] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, “Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3–21, Jan. 2009.

[5] U. Y. Ogras, J. Hu, and R. Marculescu, “Key research problems in NoC design: a holistic perspective,” *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pp. 69–74, 2005.

[6] L. Guang, E. Nigussie, J. Isoaho, P. Rantala, and H. Tenhunen, “Interconnection alternatives for hierarchical monitoring communication in parallel SoCs,” *Microprocessors and Microsystems*, vol. 34, no. 5, pp. 118–128, Aug. 2010.

[7] L. Guang, P. Rantala, E. Nigussie, J. Isoaho, and H. Tenhunen, “Low-latency and Energy-efficient Monitoring Interconnect for Hierarchical-agent-monitored NoCs,” *2008 Norchip*, pp. 227–232, Nov. 2008.

[8] C. Ciordas, K. Goossens, A. Radulescu, and T. Basten, “NoC Monitoring: Impact on the Design Flow,” in *2006 IEEE International Symposium on Circuits and Systems*, 2006, pp. 1981–1984.

[9] J. Zhao, S. Madduri, R. Vadlamani, W. Burleson, and R. Tessier, “A Dedicated Monitoring Infrastructure for Multicore Processors,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 6, pp. 1011–1022, Jun. 2011.

[10] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, “LEAR -- A Low-Weight and Highly Adaptive Routing Method for Distributing Congestions in On-chip Networks,” in *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2012, vol. 1, pp. 520–524.

[11] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, “CATRA-Congestion Aware Trapezoid-based Routing Algorithm for On-Chip Networks,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE’12)*, 2012, pp. 320 – 325.

[12] V. Rantala, T. Lehtonen, P. Liljeberg, and J. Plosila, “Distributed Traffic Monitoring Methods for Adaptive Network-on-Chip,” in *2008 NORCHIP*, 2008, pp. 233–236.

[13] P. Gratz, B. Grot, and S. W. Keckler, “Regional congestion awareness for load balance in networks-on-chip,” *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pp. 203–214, Feb. 2008.

[14] S. Ma, N. Enright Jerger, and Z. Wang, “DBAR: An Efficient Routing Algorithm to Support Multiple Concurrent Applications in Networks-on-Chip,” in *Proceeding of the 38th annual international symposium on Computer architecture - ISCA ’11*, 2011, p. 413.

[15] R. Manevich, I. Cidon, A. Kolodny, I. Walter, and S. Wimer, “A Cost Effective Centralized Adaptive Routing for Networks-on-Chip,” *2011 14th Euromicro Conference on Digital System Design*, vol. 9, no. 2, pp. 39–46, Aug. 2011.

[16] M. A. Al Faruque, T. Ebi, and J. Henkel, “ROAdNoC: Runtime observability for an adaptive network on chip architecture,” *2008 IEEE/ACM International Conference on Computer-Aided Design*, pp. 543–548, Nov. 2008.

[17] M. A. Al Faruque, T. Ebi, and J. Henkel, “AdNoC: Runtime Adaptive Network-on-Chip Architecture,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 2, pp. 257–269, Feb. 2012.

[18] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, “Exploration of MPSoC monitoring and management systems,” in *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, 2011, pp. 1–3.

[19] L. Guang, A. W. Yin, P. Rantala, P. Liljeberg, J. Isoaho, and H. Tenhunen, “Hierarchical Power Monitoring for On-chip Networks.” Turku, Finland, pp. 2–3, 2009.

[20] A. W. Yin, L. Guang, P. Rantala, P. Liljeberg, J. Isoaho, and H. Tenhunen, “Hierarchical Agent Monitoring NoCs: A Design Methodology with Scalability and Variability,” *2008 Norchip*, pp. 202–207, Nov. 2008.

[21] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, and J. Meerbergen, “An event-based network-on-chip monitoring service,” in *Proceedings. Ninth IEEE International High-Level Design Validation and Test Workshop (IEEE Cat. No.04EX940)*, 2004, pp. 149–154.

[22] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, and A. Boon, “Transaction Monitoring in Networks on Chip: The On-Chip Run-Time Perspective,” in *2006 International Symposium on Industrial Embedded Systems*, 2006, pp. 1–10.

[23] H. Yi, S. Park, and S. Kundu, “A Design-for-Debug (DfD) for NoC-Based SoC Debugging via NoC,” *2008 17th Asian Test Symposium*, pp. 289–294, Nov. 2008.

[24] B. Vermeulen and K. Goossens, “A Network-on-Chip monitoring infrastructure for communication-centric debug of embedded multi-processor SoCs,” in *2009 International Symposium on VLSI Design, Automation and Test*, 2009, pp. 183–186.

[25] E. Carvalho, C. Marcon, N. Calazans, and F. Moraes, “Evaluation of Static and Dynamic Task Mapping Algorithms in NoC-Based MPSoCs,” *inf.pucrs.br*, pp. 4–7, 2009.

[26] E. Carvalho, N. Calazans, and F. Moraes, “Heuristics for Dynamic Task Mapping in NoC-based Heterogeneous MPSoCs,” in *18th IEEE/IFIP International Workshop on Rapid System Prototyping (RSP ’07)*, 2007, pp. 34–40.

[27] E. Carvalho and F. Moraes, “Congestion-aware task mapping in heterogeneous MPSoCs,” *2008 International Symposium on System-on-Chip*, pp. 1–4, Nov. 2008.

[28] C.-L. Chou and R. Marculescu, “Run-Time Task Allocation Considering User Behavior in Embedded Multiprocessor Networks-on-Chip,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 1, pp. 78–91, Jan. 2010.

[29] Y. Cui, W. Zhang, and H. Yu, “Decentralized agent based re-clustering for task mapping of tera-scale network-on-chip system,” in *2012 IEEE International Symposium on Circuits and Systems*, 2012, pp. 2437–2440.

[30] J. Plosila, K. Latif, and H. Tenhunen, “Hierarchical power monitoring on NoC - a case study for hierarchical agent monitoring design approach,” in *NORCHIP 2010*, 2010, pp. 1–6.

[31] E. a. Carara and F. G. Moraes, “Flow oriented routing for NOCS,” *23rd IEEE International SOC Conference*, pp. 367–370, Sep. 2010.

[32] M. Palesi, S. Kumar, and V. Catania, “Bandwidth-aware routing algorithms for networks-on-chip platforms,” *IET Computers & Digital Techniques*, vol. 3, no. 5, p. 413, 2009.

[33] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, “Application Specific Routing Algorithms for Networks on Chip,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 3, pp. 316–330, Mar. 2009.

[34] R. Manevich, I. Cidon, A. Kolodny, and I. Walter, “Centralized Adaptive Routing for NoCs,” *IEEE Computer Architecture Letters*, vol. 9, no. 2, pp. 57–60, Feb. 2010.

[35] L. Leem, H. Cho, and J. Bau, “ERSA: error resilient system architecture for probabilistic applications,” in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE’10)*, 2010, vol. 31, no. 4, pp. 546–558.

[36] P. Salihundam, S. Jain, and T. Jacob, “A 2 Tb/s 6× 4 mesh network for a single-chip cloud computer with DVFS in 45 nm CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 46, no. 4, pp. 757–766, 2011.

[37] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, “A 5-GHz Mesh Interconnect for a Teraflops Processor,” *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sep. 2007.