Centralized Traffic Monitoring for online-resizable Clusters in Networks-on-Chip

Philipp Gorski, Dirk Timmermann
Institute of Applied Microelectronics and Computer Engineering
University of Rostock
Rostock, Germany
{philipp.gorski2, dirk.timmermann}@uni-rostock.de

***Abstract*—Runtime-based traffic monitoring is one of the most essential system services for modern many-core platforms. It ensures self-awareness of the current system load and enables other runtime mechanisms, like application mapping and adaptive routing, to optimize workload operations. While Networks-on-Chip introduced a scalable and massively parallel communication infrastructure for the growing number of on-chip resources, scalable traffic monitoring becomes a key concern. A high degree of runtime adaptivity at different system layers comes with the costs that monitored system states need to be available at different locations, scopes and resolutions. Furthermore, many-core systems with hundreds of resources will be less single purpose and run workloads composed of various application domains, traffic and timing characteristics. Thus, the monitoring operations should be adaptive to consider this variability and provide customizable behavior. In the work at hand, a runtime configurable and cluster-based traffic monitoring is proposed. Our experiments show that in maximum 2% hardware overhead per tile in an 8x8 NoC is needed to enable online resizable clustering of up to 64 IP cores, centralized traffic monitoring of all path and link loads inside these clusters at resolutions of 1%, and configurable monitoring timing adjustment in a range of 10^3 to 10^5 clock cycles.**

Keywords—Networks-on-Chip; Monitoring; Many-Core; Runtime Reconfiguration; HW/SW-Co-Design

I. INTRODUCTION

Networks-on-Chip (NoC) emerged as the next generation of communication infrastructures for modern many-core systems, applying packet-based communication inside a networked topology of routers interconnected by bidirectional point-to-point links [1][2]. Thereby, the most commonly used class of NoC is the two-dimensional $N_X \times N_Y$ mesh, which supports a regular physical layout, scalability and enough inherent parallelism [1]. Such NoC-based many-core systems will integrate functional resources or run software of various application domains 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, timing and constraints. Hence, runtime adaptivity of management mechanisms like application or routing is indispensable for optimized operations of these workloads [2]. Therefore, the system must be aware of its current loads and provide this information to entities at different system-levels. For distributed workload processing on NoC-based platforms the availability of accurate traffic load information is a key concern regarding the impact of the communication on the performance.

In the work at hand, a cluster-based and runtime configurable traffic monitoring solution for mesh-based NoC is presented. The monitoring data aggregation works with a centralized structure at the cluster level and is managed by a software module (called cluster agent). Thereby, a cluster represents a resizable group of cores inside a closed rectangular shape that will be placed dynamically inside the NoC and offers flexible monitoring timing adjustments. This allows regional customization for dedicated workload fractions inside the NoC and the activation of traffic monitoring in those parts of the NoC where it is needed. Furthermore, the global data collection can be established by requesting the partial cluster information at few dedicated entities. The proposed traffic monitoring captures the loads of links, complete minimal paths with source-destination-pairs inside the cluster, and overall data in/out loads of the IP cores. Thereby, the final load values, accessed by the cluster agent, are available in time periods of 10^3 to 10^5 clock cycles and scaled from 0-100% of the maximum limits with configurable scale resolutions of 1%, 2% and 4%. This avoids costly data processing in software and our experiments show that a tradeoff in data accuracy versus timing can be reached. For data transport of the monitoring and management information a unified dedicated infrastructure is established (see Figure 1). The many-core system contains one NoC for the exchange of workload/application data (Data-NoC/DNoC) and another one for the transfer of system management data (System-NoC/SNoC). This communication strategy offers minimal interference of the different traffic domains, separates the design constraints and the unified SNoC can be reused by other management mechanisms [3][4][5][6]. Both NoC apply input-buffered wormhole-switching, minimal oblivious routing algorithms, round robin link arbitration and hop-based REQ/ACK flow control.

The remainder of this work is organized as follows. Section 2 discusses the related work of existing approaches. The general concept of the proposed traffic monitoring is introduced in Section 3. Section 4 highlights specific design and implementation aspects. The evaluation of the experimental results for different corner cases and an analysis of the expected hardware overhead follow in Section 5. Afterwards, Section 6 gives a final conclusion and an outlook for future investigations.

II. Related Work

The majority of published works about traffic monitoring in NoC propose dedicated solutions for debugging, adaptive routing and task mapping mechanisms. The infrastructural integration can be separated into three main concepts: the shared infrastructure (SI), dedicated infrastructure (DI) and the unified infrastructure solution (UI) [3][4][5][6]. SI uses the same NoC as communication infrastructure for application and monitoring data. DI integrates additional physical interconnects and in UI the application and the monitoring traffic are assigned to different communication infrastructures, as applied in here by the DNoC and SNoC. Most of the distributed packet-level adaptation mechanisms in NoC apply regional congestion/traffic monitoring, like implemented by CATRA[7], RCA [8] and DBAR [9]. Contrary, the ATDOR approach in [10] utilizes a globally centralized traffic monitoring concept. But these DI-based single-purpose approaches implement the traffic monitoring with a focus on strong information reduction for low hardware costs and without global reuse of the monitored states. Other approaches target the event-based traffic monitoring via dedicated hardware probes at the router and network interfaces. In [11], the SI-based RoadNoC applies such a solution for a distributed and connection-oriented traffic state monitoring. Another event-based monitoring concept via SI for the *AEthereal* NoC is presented in [12]. This work tackles event taxonomy, probe design/programming, and a variable hierarchy (distributed or centralized). It is extended in [13] to support different abstraction-levels of events. Furthermore, the solutions in [14][15] covers the transaction-based debugging and the integration of special monitoring probes at design time. Other works target the optimized design of the event probes and their system integration.

A general exploration of monitoring and management strategies for NoC is given in [6][5][6][17]. These works further introduce an own hierarchical concepts of agent-based management and clustering.

III. Concept

In general, centralized system management schemes impose the requirement for the decision evaluating entities to operate on a broader system information context and thereby offer optimization with global awareness. But the corresponding global data monitoring and distribution represents a bottleneck which makes such solutions less scalable than distributed approaches. Thus, the hierarchical organization via flexible sized regions/clusters and information flows can be used to tackle this issue. In the work at hand, such a flexible clustered solution will be achieved by a HW/SW-Co-Design that implements traffic load sensing as well as data aggregation in hardware, while cluster management and monitoring data evaluation is processed by a software module (cluster agent). According to Figure 1, the smallest entity for the traffic monitoring is a CELL/TILE that integrates an IP core, the Network Interfaces for the DNoC (DNI) and the SNoC (SNI), and the router (R) and links for both NoC. Thereby, the IP cores are separated into normal and master cores. The master cores are sparsely distributed and represent resources where the cluster agents operate. Therefore, they will preserve a free resource budget and only one master core per cluster will be needed, while additional unused ones inside a cluster will serve as optional migration points for the cluster agent in case of an error or at a cluster reconfiguration. Furthermore, they will implement additional hardware components for monitoring data aggregation and fast software access to it. The ratio of master to normal cores and the distribution scheme are defined at design time.

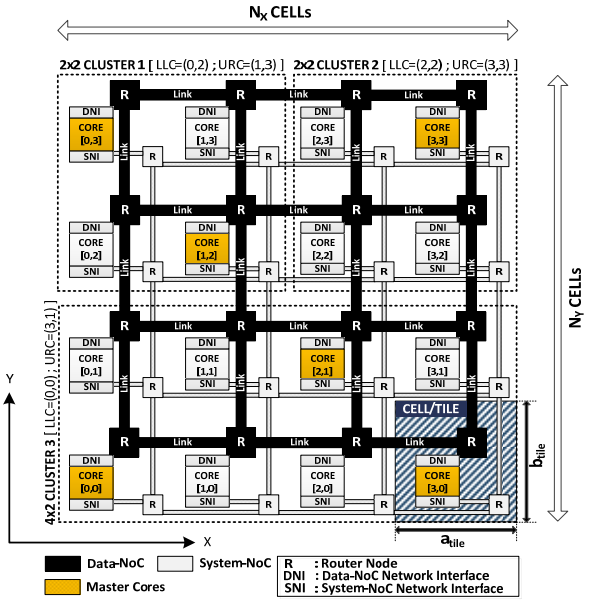

Figure 1. Exemplary 4x4 mesh-based many-core system with DNoC and SNoC infrastructure

A. Dynamic Clustering

A cluster is a rectangular shaped group of cells, inside the $N_X \times N_Y$ mesh, that is managed by a centralized administration. Thereby, the number of cells at a single cluster i is defined by n_{Ci} inside a variable range of 1 to N_{CLmax} with $N_{CLmax} < N_X \cdot N_Y$. The maximum cluster size for the traffic monitoring (N_{CLmax}) is defined at design time and the proposed solution is evaluated for 16 up to 64 cell cluster. The n_{Ci} cells are arranged inside a closed region with a shape of $n_{Cxi} \times n_{Cyi}$ and the requirement that this region contains at least one master core. The rectangular region will be defined by the global NoC {x, y}-addresses of its lower left corner cell (*LLC*) and the upper right corner cell (*URC*). Each cell is exclusively assigned to one traffic monitoring cluster inside the many-core system and thus no overlapping regions exist. Figure 1 depicts an exemplary case of three coexisting clusters (CLUSTER 1-3). The flexible sizing and positioning of clusters require a consistent cluster-internal identification of source-destination-paths and cells, which does not depend on the global xy-coordinate address space of the NoC (= $log_2(N_X)$+ $log_2(N_Y)$ bit). Thus, each cell of the cluster gets an internal identifier (*GROUP-ID*) with an address space of $log_2(N_{CLmax})$ bit. The decisions for the creation, sizing and placement of a cluster are processed at a global system level. Therefore, the application mapping entity represents a good candidate, because of its knowledge about the workload behavior and the binding of the traffic monitoring

to the requirements of mapped workload fractions. Afterwards, the cluster building, maintenance and information services will be delegated to the cluster agent. This entity requests, organizes and maintains the assigned cells via the following messaging scheme:

I. CLUSTER-CREATION-ASSIGNMENT (*CCA*): Initially, the cluster agent receives its task definition for the new cluster creation process. The corresponding packet addresses the selected master core and contains a unique context identifier (*CTX-ID*) to classify the contents, configuration data (*CTX-D*), and the region specification for the cluster (*LLC*, *URC*). The cluster agent acknowledges this packet and immediately starts the cluster setup via the transmission of cluster requests (*CRQ*).

II. CLUSTER-REQUEST (*CRQ*): The master core sends n_{Ci} request packets to all cells inside the spanned region between *LLC* and *URC*. Thereby, a request packet contains the destination address of the cell, the NoC address of the master core (*MC-ADR*), the *GROUP-ID* of the destination cell, the context information (*CTX-ID*, *CTX-D*), and the cluster size specifications (*LLC*, *URC*). Afterwards, it waits for the response packets (*CSP*).

III. CLUSTER-RESPONSE (*CSP*): The cluster cells receive request or update packets of the master core, stores the defined changes and returns a response packet to the master core as acknowledgement. The packet addresses the *MC-ADR* and contains the *GROUP-ID* as well as the *CTX-ID* to classify the packet. After the reception of n_{Ci} *CSP* packets the cluster agent has established or updated the full cluster.

IV. CLUSTER-UPDATE (*CUP*): During cluster operation the configuration data (i.e. monitoring periods) need to be adapted, the cluster agent migrates to another master core or the cluster will be deleted. Thus, the n_{Ci} cells are informed via update packets, which have the same format like the *CRQ*.

B. Traffic Monitoring Flow

The traffic monitoring has a hierarchical aggregation structure of three stages (link/path, cell, master) inside the cluster, which requires an equal clock rate at each active cluster cell but no globally synchronized timing or timestamps. Thus, it follows the globally asynchronous locally synchronous design (GALS) paradigm of NoC. For sensing and aggregation of traffic information at these stages the functions of equation (1) to (4) are applied.

$$cnt_j(t_{i+1}) = \begin{cases} cnt_j(t_i)+1 & (s_{ctrl}=1) \wedge cnt_j(t_i) < b_{T-MODE} \\ cnt_j(t_i) & (s_{ctrl}=0) \\ 0 & (s_{ctrl}=1) \wedge cnt_j(t_i) = b_{T-MODE} \end{cases} \quad (1)$$

$$OFG(t_{i+1}) = \begin{cases} 1 & (s_{ctrl}=1) \wedge cnt(t_i) = b_{T-MODE} \\ 0 & (rst=1) \\ OFG(t_i) & else \end{cases} \quad (2)$$

$$check(t_{sp}) = \sum_{j=0}^{N_S-1} OFG_j(t_{sp}) > 0 \quad (3)$$

$$load_j(t_{sp+1}) = \begin{cases} load_j(t_{sp})+1 & (OFG_j(t_{sp+1})=1) \wedge (rst=0) \\ load_j(t_{sp}) & (OFG_j(t_{sp+1})=0) \wedge (rst=0) \\ 0 & rst=1 \end{cases} \quad (4)$$

STAGE 1: The first stage covers the activity sensing of paths and links at each cell. Therefore, basic traffic sensors work as configurable counter cnt_j via function (1). These sensors will be triggered by a control signal $s_{crtl}=\{0,1\}$ that indicates the active transmission of data at a path or link. The time base Δt for signal changes is the clock cycle t_{clk} of the DNoC and thus the sensors count activity events in time steps of $t_{i+1}=t_i+ t_{clk}$. For each time step the signal indicates activity, the counter will be incremented by one up to a defined bound $b_{T\text{-}MODE}$. If this bound is reached the sensor j sets an overflow flag OFG_j as given by (2) and restarts. This reduces a defined amount of $b_{T\text{-}MODE}$ observed activity events per sensor to a single bit. Furthermore, it defines that the minimum time period between two overflows will be $t_{sp}=b_{T\text{-}MODE} \cdot t_{clk}$. The $b_{T\text{-}MODE}$ value is set by the cluster agent (as part of the *CTX-D*), configurable at runtime and equal at all cluster cells.

Each cell contains $N_S=N_{PS}+N_{LS}$ traffic sensor instances. For path traffic measurements $N_{PS}=N_{CLmax}$ sensors are needed. One sensor for the overall output of the IP core and (N_{CLmax}-1) sensors to cover all connections to the other cells inside the cluster at its maximum size. As trigger the REQ/ACK flow control signals at the DNI will be used. This ensures that a completed data transmission is measured. Each time the IP core transmits data to the DNoC router the overall output sensor is activated and, if the destination cell of the data relies inside the cluster, the corresponding path sensor j is incremented too. To measure the link traffic load at the DNoC router of the cell N_{LS}=5 traffic sensors are needed. In the focused mesh topology each router has five output ports (*NORTH, EAST, SOUTH, WEST, CORE*) and their arbitration status signals trigger if the outgoing links at these ports are active or not. Thereby, active link arbitration includes data transmissions and busy waiting cycles. This ensures the correct load sensing and detection of congested situations, where no real data transmission happens but bandwidth is lost.

STAGE 2: The second stage periodically checks the *OFG* bits of all N_S traffic sensors at a cell as given in (3) with the periodicity of t_{sp}. If overflows occurred at the current period ($check(t_{sp}) > 0$) a bit-vector of all *OFGs* will be read out and send as monitoring packet via the SNoC to the master core (*MC-ADR*) of the cluster agent. Furthermore, the *OFGs* have to be reset ($rst=1$) in that case. The packet additionally contains the *CTX-ID* and the *GROUP-ID* of the cell to be correctly processed at the master core.

STAGE 3: The third stage contains the central aggregation of the transmitted *OFG* bit-vectors at the master core via the function described by (4). For each of the n_{Ci} cluster cells a vector of load counter exist. The vector will be addressed by the *GROUP-ID* of the cell and the counter value $load_j$ by the sensor id j ($0 \leq j \leq (N_S - 1)$). A counter value will be incremented by one if its corresponding sensor reported a positive overflow ($OFG_j=1$) for a t_{sp} period.

The correct mapping of $load_j$ to a scale of 0-100% of the maximum path/link load with a defined resolution of k_s is established by the local data capture timing of the cluster

agent. A sensor is enabled to report overflows with a minimum period of t_{sp} and if k_s represents one scale step between 0-100% it describes the weight of each reported *OFG* as well. Thus, the cluster agents need to capture and modify the load counts after $N_{SP}=100/k_s$ full t_{sp} periods. The resulting monitoring cycle T_{MC} at the cluster agent will be $T_{MC}=N_{SP}\cdot t_{sp}$. Hence, the selection of k_s at the cluster agent offers another option for timing control of the traffic monitoring (in addition to the adjustment of the $b_{T\text{-}MODE}$ value at the cell level). After capturing procedure the $load_j$ counter are reset and the read values need modifications as described in (5). This operation depends on the selection of k_s and our solution supports the interval $k_s=\{1, 2, 4\}$. The $0{:}k_s{:}100\%$ scaling of $load_j$ to $sload_j$ is reached by the product $ks\cdot load_j$. But through the selection of k_s as 2^p this step can be reduced to a simple left shift of p bits for the binary representation of $load_j$.

$$sload_j = \begin{cases} load_j & k_s = 1 \Rightarrow N_{SP} = 100 \\ load_j << 1 & k_s = 2 \Rightarrow N_{SP} = 50 \\ load_j << 2 & k_s = 4 \Rightarrow N_{SP} = 25 \end{cases} \quad (5)$$

Each $sload_j$ represents an integer value from 0 to 100 and can be stored in a single byte. Thus, the maximum memory consumption of the traffic data per monitoring cycle and cluster is $N_S \cdot N_{CLmax}$ bytes.

C. Traffic Monitoring Configuration

The traffic monitoring enables an adaptive timing behavior by the selection of two parameters ($b_{T\text{-}MODE}$, k_s). TABLE I contains the resulting monitoring cycles for test design case used in this work. With an adjustable monitoring timing in a range of 10^3 to 10^5 clock cycles the complete traffic situation inside the cluster at the transaction- or message-exchange-level can be captured in a central instance. Thus, this monitoring does not enable a locally distributed packet-by-packet adaptivity and rather focusses the path based data stream adjustments at higher levels.

TABLE I. RESULTING MONITORING CYCLES T_{MS} IN µS FOR t_{clk}=1NS AND VARIATION OF $b_{T\text{-}MODE}$ AND AT POSSIBLE k_s

	Monitoring Cycle T_{MS} [µs] for t_{clk}=1ns		
$b_{T\text{-}MODE}$	k_s=1	k_s=2	k_s=4
64	6.4	3.2	1.6
128	**12.8**	**6.4**	**3.2**
256	25.6	12.8	6.4
512	51.2	25.6	12.8
1024	**102.4**	**51.2**	**25.6**
2048	204.8	102.4	51.2

For the adjustment of the configuration parameter the dependencies formulated in (6) and (7) have to be considered.

$$\left\lceil \frac{n_{Ci}}{b_{T-MODE}} \right\rceil \le c_f \cdot r_{\max} \; with \; (0 < c_f \le 1) \quad (6)$$

$$sload_i \pm e_{\max} \; with \; e_{\max} \le 2 \cdot k_s \quad (7)$$

The $b_{T\text{-}MODE}$ is set at the cells and the adaption effort correlates to n_{Ci}. As mentioned, the $b_{T\text{-}MODE}$ further determines the worst-case injection period of monitoring packets (t_{sp}) and thus its selection range depends on the number of observed cells as well. The regional clustering only reduces the bottleneck of the applied centralization, but the master core still represents the hotspot of the underlying $n_{Ci}{:}1$ traffic pattern (worst-case is $N_{CLmax}{:}1$). Thus, the packet reception capability of its SNI (r_{max}), the resulting queuing delay and congestions of the monitoring packets are the limiting factors for the $b_{T\text{-}MODE}$ adjustment. The selected $b_{T\text{-}MODE}$ has to fit the worst-case condition formulated in (6) that n_{Ci} monitoring packets injected in $b_{T\text{-}MODE}$ clock cycles must fit the maximum packet reception rate r_{max} at the master core SNI. Thereby, c_f represents a correction factor to consider related traffic interferences and dependencies in the SNoC. For the presented scenarios of this work c_f =0.7 has proven as good conditioning.

The adaptation of k_s is done at the cluster agent and defines the timing of the traffic monitoring cycle T_{MC}. Moreover, k_s determines the scale resolution and accuracy of the monitored loads. Thereby, each missed OFG_j from a monitoring packet results in an error at $sload_j$ of k_s. Our worst-case experiments showed that the condition of (7) is valid if the $b_{T\text{-}MODE}$ is selected accordingly to (6). The maximal error $e_{max} \le 2 \cdot k_s$ results from the situation that the last reported monitoring packet does not reach the master core inside the current monitoring cycle (OFG_j miss) and the affected traffic sensors have further counted events ($0 < cnt_j < b_{T\text{-}MODE}$) that remains as undetected rest. Thus, k_s allows a fast adaptation of the monitoring timing under consideration of the needed load data accuracy.

IV. DESIGN

The traffic monitoring concept presented above requires additional hardware at each cell and the dimensioning of some parameter at design time. Figure 2 depicts a schematic of the Ip core interfaces SNI/DNI to the SNoC/DNoC and highlights the established hardware modifications.

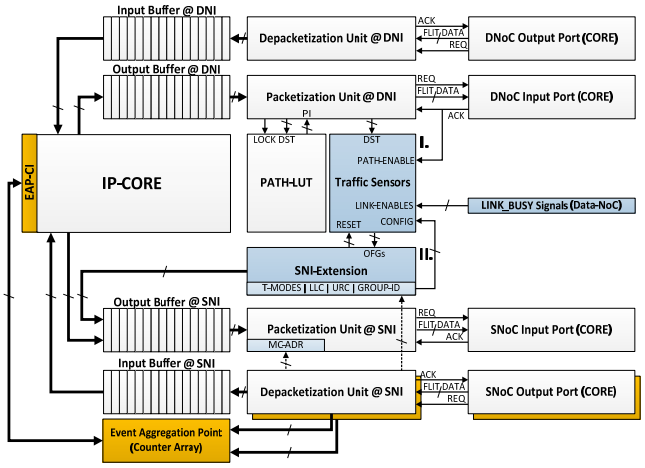

Figure 2. Schematic overview of DNI and SNI elements at the IP core

Each cell implements the N_S traffic sensors (I) for monitoring stage 1 and a SNI-Extension (II) that processes the periodic *OFG* packet generation of stage 2. The master cores will further integrate the load counter to aggregate the reported *OFGs* as needed by stage 3 (Event Aggregation Point). The

number of traffic sensors and load counter is dominated by the maximum cluster size N_{CLmax}, while the width of a traffic sensor is defined by the maximum supported $b_{T\text{-}MODE}$ and the upper load counter bound is constrained to 100. The traffic monitoring is evaluated with maxima of 16 and 64 cell clusters at different $n_{Cxi} \times n_{Cyi}$ shapes.

According to equation (6), the adaptivity of the monitoring cycles is constrained by the current cluster size n_{Ci} and the packet reception capability r_{max} of the master core SNI. Thereby, the selected linkwidth w_L of the SNoC defines the number of datawords/flit per packet for the given amount of transmission data and thus the consumption delay per packet at the SNI. Due to the REQ/ACK flow control each flit need two clock cycles to be consumed. To enable lower $b_{T\text{-}MODE}$ values for greater clusters two major design time options exist. First, increasing w_L reduces the consumption delay per monitoring packet at the SNI and raises r_{max}. But the resulting hardware overhead affects the complete SNoC. Alternatively, the master core SNI and the connected SNoC router can be suited with a dual ported (DP) design. This enables the consumption of two packets of different input ports in parallel at the SNI. Thus, r_{max} can be doubled and the hardware overhead is reduced to the number of master cores. This work applies the second option to double r_{max}.

TABLE II. IMPORTANT DESIGN PARAMETER AND SETTING FOR THE TRAFFIC MONITORING AND EXPERIMENTAL EVALUATION

Parameter	Value	
NoC	*SNoC*	*DNoC*
Linkwidth w_L	8-, 16-bit	64-bit
Clock Rate t_{clk}	1ns	
Port Buffer Depth	1 flit	5 flit
$b_{T\text{-}MODE}$	64 - 2048	
Cluster Size N_{CLmax}	16 and 64	
Cluster Shape $[n_{Cxi} \times n_{Cyi}]$	4×4, 8×2 and 16×4, 8×8	
Tile Area A_{tile}	3mm × 3mm	

TABLE II provides a collection of parameter settings used for the synthesized and simulated test cases of this work..

A. Traffic Sensors

The design of the traffic sensors (TS), for the path and link activity counter of stage 1, is realized with a combination of a binary counter and comparator (see Figure 3). The supported $b_{T\text{-}MODE}$ interval is *{64, 128, 256, 512, 1024, 2048}* and this demands a data width of 12-bit at the counter/comparator.

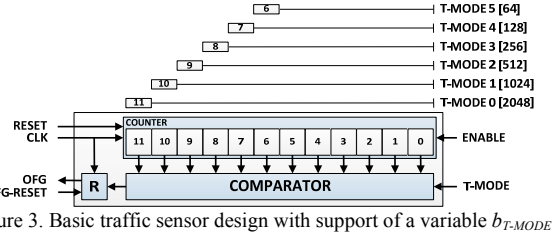

Figure 3. Basic traffic sensor design with support of a variable $b_{T\text{-}MODE}$

According to s_{ctrl}, the counter is triggered by an ENABLE signal and checked by the comparator if the defined bound (T-MODE=3-bit select signal) is reached. The generated *OFG* bit is written to the register R, and can be externally read and reset by the SNI-Extension. Thus, the presented unit implements the functions defined (1) and (2). All N_S traffic sensors of a cell are grouped and integrated as part of the DNI (see Figure 2 and Figure 4).

Furthermore, a component (see Figure 4: CLUSTER-CHECK+ID-GENERATOR) is integrated that solves the issue of the validation and unique addressing of path sensors for destinations inside the defined cluster (*LLC*, *URC*). This component implements the functions given in (8) and (9). If a packet is going to be send through the DNI the packetization unit sets the destination *DST* (x_{DST}, y_{DST}) at its header. First, the destination coordinates will be checked by (8) if they will be located in the defined cluster. Therefore, four comparators with the size of $log_2(N_X)$ and $log_2(N_Y)$ bit are needed. The second step generates a valid *GROUP-ID* after the conditions of (8) were met. This implies a zero shift of *DST* with the *LLC* as new arbitrary origin and a final bitwise xor operation on the resulting coordinate vectors (x_{CC}, y_{CC}). For the zero shift two ripple carry adder for $log_2(N_X)$ and $log_2(N_Y)$ bit operands are implemented. Afterwards, the shifted coordinates are used to generate the *GROUP-ID* with a width of $w_{ID}=log_2(N_{CLmax})$ bit, where $w_{ID} \leq (log_2(N_Y) + log_2(N_X))$.

$$x_{LLC} \leq x_{DST} \leq x_{URC}$$
$$y_{LLC} \leq y_{DST} \leq y_{URC} \quad (8)$$

$$x_{CC} = x_{DST} - x_{LLC}$$
$$y_{CC} = y_{DST} - y_{LLC} \quad (9)$$
$$GROUP-ID = x_{CC}[(w_{ID}-1) \rightarrow 0] \otimes y_{CC}[0 \rightarrow (w_{ID}-1)]$$

Therefore, the calculated cluster coordinates are cut down to the w_{ID} lower bits and a bitwise xor of the x_{CC} sub-vector with the reversed y_{CC} sub-vector results in a unique *GROUP-ID* for the inbound destination cell. The result serves as selection input for a multiplexer to enable the corresponding traffic sensor. The same *GROUP-ID* calculation scheme is performed by the cluster agent during the cluster creation to assign each cell its unique cluster identifier.

B. SNI-Extension

The periodic *OFG* checking and monitoring packet generation of stage 2 is done by the SNI-Extension (see Figure 2 and Figure 4). This component further stores the configuration data (*LLC*, *URC*, *T-MODES*, *GROUP-ID*). The selected *T-MODE* controls the traffic sensors and a 12-bit *TIMER*, which triggers the periodic operations of a finite state machine (*FSM*). Each t_{sp} period the *FSM* tests if the *OFG-CHECK* enables the generation of a monitoring packet for the cluster agent. The *OFG-CHECK* implements the function in (3) via xor over all *OFG* registers of the connected traffic sensors. If the packet generation is enabled the *OFGs* of all N_S sensors are read out and reset. The *FSM* composes the packet data, pushes it flit by flit into the output buffer of the SNI and afterwards the packet is transmitted through the SNoC to the cluster agent. Thereby, the monitoring packet consists of a header flit containing the routing information, another flit for *GROUP-ID* and *CTX-ID*, and the N_S/w_L flit for the *OFG* bit-vector. The master core address (*MC-ADR*) of the header flit is added by the packetization unit of the SNI, while the rest remains for the *FSM*.

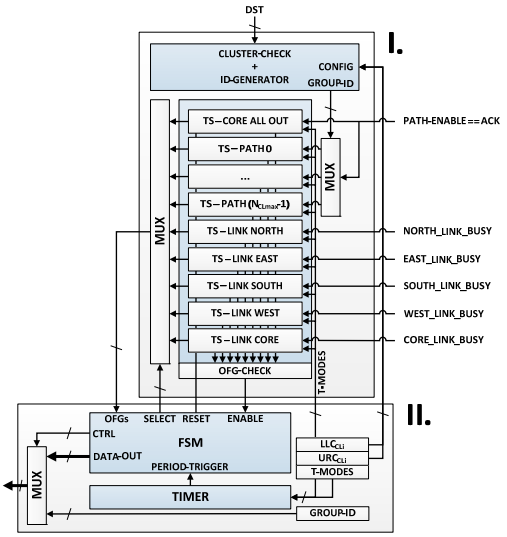

Figure 4. Detailed schematic of I. grouped traffic sensors (TS) and II. SNI-Extension at each cell

C. Event Aggregation Point

The final aggregation of reported *OFGs* at the master core is established as special hardware component, called Event Aggregation Point (EAP), which is part of the SNI (see Figure 2 and Figure 5). It consists of N_{CLmax} groups of 7-bit counter with N_S counter per group and implements the functionality defined by (4). The EAP is sourced by a buffer that takes the *OFG* vector and *GROUP-ID* from incoming monitoring packets. The *GROUP-ID* defines the counter group and the index of the *OFG* inside the vector represents the identifier of the traffic sensor.

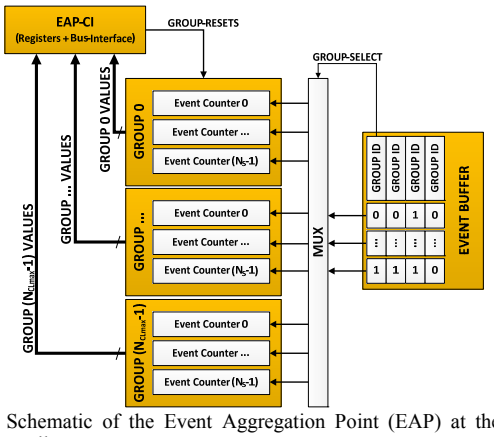

Figure 5. Schematic of the Event Aggregation Point (EAP) at the SNI of master core cells

The EAP is connected through a core interface (EAP-CI) to the internal bus of the IP core and directly accessible. This EAP integration enables the cluster agent to access the aggregated load directly and avoids costly *OFG* aggregation in software. The cluster agent can read the counter values with or without reset. This allows two different capturing strategies. After a full monitoring cycle the load values are read and the counter must be reset. But during a monitoring cycle the values can be captured without reset. This allows the tracing of load/traffic curves for dedicated paths or links in smaller time resolutions.

V. Experimental Results

The experimental evaluation was established by full system simulations and the hardware synthesis for cost approximation. The applied parameter settings of clock rate, port buffer depth and cluster shape as well as sizing can be obtained from Table II. Thereby, the SNoC and DNoC operated on the same clock rate. Furthermore, the synthesized and simulated SNoC implemented the XY-routing with a minimal port buffer depth of one flit. The DNoC was simulated with XY-routing and an adaptive minimal XY/YX-routing [18] for each parameter configuration. The port buffer depth was five flit and the output buffers at DNI as well as SNI were constrained to 4096 flit.

A. Hardware Synthesis

The ASIC design flow was realized with the Synopsys Design Compiler using the 45nm Nangate FreePDK45 Generic Open Cell Library. Table III contains the results for different parameter configurations. For each of the hardware components the total cell area is extracted and the summed area costs for both cell types (normal and master) are calculated. Thereby, the area costs for the SNoC links considers two bidirectional links per cell (see Figure 1) with a wire length l_W of 2900 µm and a wire pitch w_P of 0.14 µm. A single bidirectional link has $2 \cdot w_L$ wires for data transmission and 4 control wires per direction. The area per link is calculated by the equation $2 \cdot (w_L+4) \cdot l_W \cdot w_P$.

Table III. Hardware Synthesis Results of the Traffic Monitoring Components as Total Cell Area and Relative Overhead per Tile

Total Cell Area @ 45nm [µm²] for t_{clk}=1ns, N_X=8 and N_Y=8				
N_{CLmax}	*16 cells*		*64 cells*	
SNoC w_L	*8-bit*	*16-bit*	*8-bit*	*16-bit*
1. SNoC Router	2436.53	3166.19	2436.53	3166.19
2. DP SNoC Router	3121.78	4039.21	3121.78	4039.21
3. Traffic Sensors (I)	2765.33	2765.33	8807.53	8807.53
4. SNI Extension (II)	1208.24	1208.24	2598.5	2598.5
5. EAP	19288.19	19288.19	245486.08	245486.08
6. SNoC Links	19488	32480	19488	32480
$\sum_{master}$ **[2,3,4,5,6]**	45871.54	59780.97	279501.89	293411.32
$\sum_{master} / A_{tile}$	0.51%	0.66%	3.11%	3.26%
$\sum_{normal}$ **[1,3,4,6]**	25898.1	39619.76	33330.56	47052.22
$\sum_{normal} / A_{tile}$	0.29%	0.44%	0.37%	0.52%

The relative overhead is defined by the ratio of the summed area costs to the available area per cell/tile (A_{tile}). Assuming a homogeneous layout of $N_X \times N_Y$ tiles, this ratio can be used to estimate the total system level overhead as well. The selected sizing for A_{tile} of 3 mm × 3mm was deviated from the 45nm Intel SCC platform in [19]. The results show that the implementations costs for the additional monitoring hardware units imposes a feasible overhead and will be dominated by the sizing N_{CLmax}. Under consideration of a high master core ratio of 50% the average overhead estimate per tile at 16/64 cell clusters would be below 0.6/2.0 percent.

B. Simulation Results

The traffic monitoring performance was evaluated with a cycle-accurate system simulation environment written in SystemC. For the experimental prove of condition (6) and (7) a worst-case scenario of parameter settings and cluster agent location was applied to varying workloads. Therefore, maximal cluster size of 16/64 cells was adjusted and the

cluster agent ran at the *LLC*. This corner cell placement results in a traffic situation with the maximum average packet delay for the monitored cells (maximum average hop distance to master core). For the 16/64 cell configuration a SNoC linkwidth w_L of 8/16 bit was applied and the resulting size of the monitoring packets is 5/7 flit. With applied dual-ported (DP) master core and a consumption delay of two clock cycles per flit at the SNI, r_{max} can be calculated to 0.2/0.143 packets per clock cycle for the 16/64 cell cluster. According to (6), the resulting minimal allowed $b_{T\text{-}MODE}$ is 128 for n_{Ci}=16 and 1024 for n_{Ci}=64. These monitoring configurations of max sizes at minimal possible $b_{T\text{-}MODE}$ values were simulated with two different traffic scenarios and at all possible k_s of 1, 2 and 4. Each case was simulated 10 times with 10 full monitoring cycles per run for both DNoC routing configurations. Thereby, the real loads of all paths (PL) and links (LL) were logged and compared to the captured data of the proposed monitoring after each finalized monitoring cycle. The resulting deltas represent the absolute monitoring errors and the maximum as well as the average error for the PL and LL were extracted from these deltas.

The first simulated scenario covers the traffic injection up to a saturated DNoC for the evaluation of the traffic monitoring stability. Therefore, the uniform random traffic pattern was applied for the DNoC communication, because it enables highest saturation levels and the balanced traffic distribution activates most traffic sensors. Each IP core injected packets with a uniform selection of the destination address and a size in the range of 5 to 15 flits. Figure 6 shows the simulation results for the maximum monitoring error of a 4x4 shaped cluster over the full traffic injection range and all k_s settings. The increased result density at the injection rate levels around 0.25 indicate that the DNoC became fully saturated. The analysis of link and path activity showed that up to 100% of all traffic sensors were active in that region.

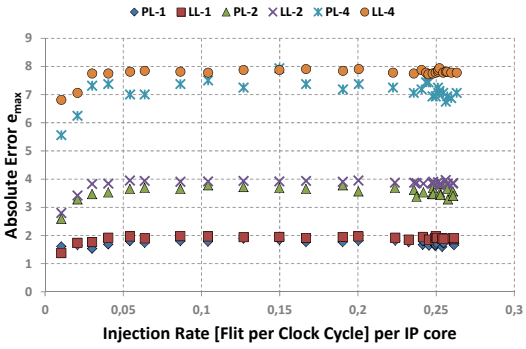

Figure 6. Resulting absolute error for path (PL) and link loads (LL) in a 4x4 shaped 16 cell cluster at k_s={1, 2, 4}, $b_{T\text{-}MODE}$=128 and increasing traffic injection rates [PL/LL-k_s]

Figure 7 and Figure 8 summarizes the results of all simulated monitoring configurations. Thereby, the maximum errors for PL and LL proves the e_{max}=±2·k_s statement of (7) for both cluster sizes at all simulated shapes if $b_{T\text{-}MODE}$ adjustments follow (6). Furthermore, the maximum average error taken from the simulated traffic load range at each monitoring configuration indicates a practical bound of ≤0.25· e_{max}.

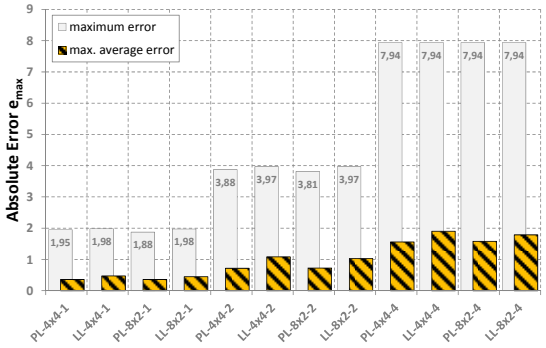

Figure 7. Monitoring error statistics for all simulated 16 cell cluster configurations at DNoC saturation scenario [PL/LL-Shape-k_s]

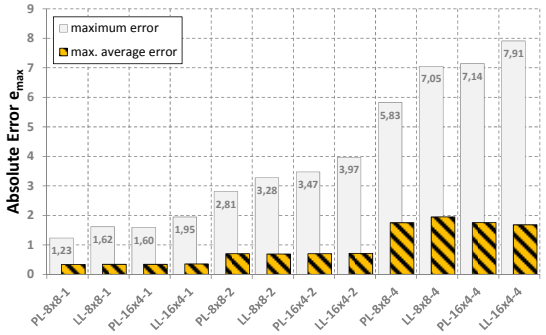

Figure 8. Monitoring error statistics for all simulated 64 cell cluster configurations at DNoC saturation scenario [PL/LL-Shape-k_s]

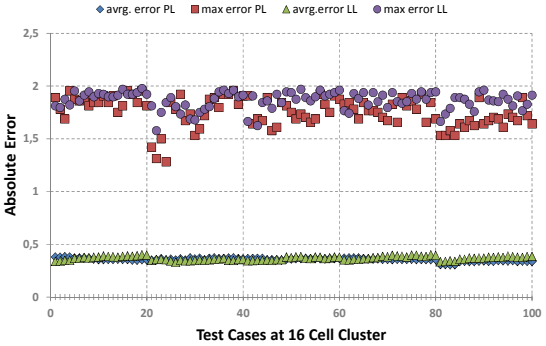

Figure 9. Monitoring error statistics for all mixed application scenarios in a 16 cell cluster with k_s=1 and $b_{T\text{-}MODE}$=128

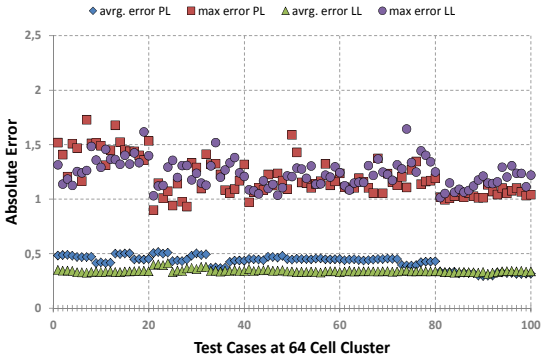

Figure 10. Monitoring error statistics for all mixed application scenarios in a 64 cell cluster with k_s=1 and $b_{T\text{-}MODE}$=1024

The second scenario targets the simulation of workloads with mixed traffic characteristics. Therefore, a set of random

application task graphs with 7 to 70 tasks per graph was generated. Afterwards, 100 random workloads (test cases) with 20 to 400 tasks (2 to 10 task graphs) per workload were composed of this set and simulated with a random mapping. The tasks communicate via packets with a uniform random size from 5 to 50 flits that is selected for each communication edge at the beginning of a simulation run. A task is triggered periodically in a uniform random range of 100 to 500 clock cycles and sends a packet to one of its successor nodes. The successor selection is random uniform as well. The resulting traffic in the DNoC is unbalanced with a high spread of the loads at the paths and links. Thus, it has a complementary behavior to the first simulation scenario. The resulting maximal and average monitoring errors for PL/LL for the highest scale resolution at k_s=1 are depicted in Figure 9 for the 16 cell cluster and in Figure 10 for the 64 cell cluster. The results confirm the upper error limit of (7) and indicate an average monitoring error in the range of $0.2 \cdot e_{max}$ to $0.25 \cdot e_{max}$.

VI. Conclusion and Future Work

This work introduced a flexible centralized traffic monitoring for NoC that enables the full coverage of traffic situations inside specific regions with 16 up to 64 IP cores at the path and link level under consideration of feasible additional hardware efforts. The monitoring offers a high degree of adaptivity regarding the timing, cluster sizing and cluster position. The achievable monitoring cycles of 10^3 to 10^5 clock cycles with load data resolutions of 1% to 4% support the use for cluster-based hierarchical application mapping and path-based routing adaptations. Furthermore, we could offer a practical strategy for the monitoring cluster configuration and show that the absolute error relies below a defined bound if this strategy is applied. The next steps of our investigations target the coupling to other runtime mechanisms and further optimizations of the monitoring as well as design refinements. Especially, a combined application mapping that incorporate the global cluster planning for the workloads and the local cluster mapping for assigned workload fractions.

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] R. Marculescu, U. Y. Ogras, L. 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.

[3] 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.

[4] 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.

[5] 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.

[6] A. W. Yin, P. Rantala, E. Nigussie, J. Isoaho, and H. Tenhunen, "Hierarchical agent based NoC with dynamic online services," *2009 4th IEEE Conference on Industrial Electronics and Applications*, pp. 434–439, May 2009.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] P. Gorski, C. Cornelius, D. Timmermann, and V. Kühn, "Centralized Adaptive Source-Routing for Networks-on-Chip as HW/SW-Solution with Cluster-based Workload Isolation," in *8th International Conference on Systems (ICONS'13)*, 2013, no. c, pp. 207–215.

[19] J. Howard, S. Dighe, S. R. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. K. De, and R. Van Der Wijngaart, "A 48-Core IA-32 Processor in 45 nm CMOS Using On-Die Message-Passing and DVFS for Performance and Power Scaling," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, Jan. 2011.