Centralized Adaptive Source-Routing for Networks-on-Chip as HW/SW-Solution with Cluster-based Workload Isolation

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—**The growing number of applications and processing units in modern MPSoCs comes along with dynamic and diverse workload characteristics at runtime. Thus, the communication infrastructure, e.g., Networks-on-Chip (NoC), operation on time dependent dynamic traffic loads makes adaptive congestion and load management indispensable. This paper introduces a centralized adaptive path management for oblivious source routing. Thereby, a cluster-based, runtime-configurable software solution continuously monitors the global traffic situation and calculates the needed routing adaptations for each active source-destination pair of the current workload inside a cluster. Contrary to other published solutions, a HW/SW-Co-Design for configurable clustering, traffic monitoring and path calculation is applied. Furthermore, the isolation of workload fractions by the spatial clustering allows application specific configurations.**

Keywords-Network-on-Chip, MPSoC, Adaptive Routing, Traffic Monitoring, Clustering, 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][4][5][6][7]. 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. Thereby, the selected routing policy in NoC is one of the major design aspects, as it defines the degree of parallelism and redundancy of the NoC that will be utilized for the communication of the workload applications. Most commonly oblivious, minimal and dimension ordered (DOR) algorithms, like the XY-Routing, are used. This class of routing algorithms results in low hardware costs, deterministic behavior, and, shows good performance results for static workloads. But concerning unpredictable dynamic workloads including the presence time dependent and domain-specific traffic, they suffer from the absence of path adaptability for load balancing and system stabilization. Thus, adaptive routing strategies became research focus, which adjust the routing paths at runtime based on current traffic information. Thereby, the proposed algorithms mainly differ in the criteria, where the routing path will be adjusted, if the paths are minimal or not, and the scope of traffic information used for the decisions. The majority of existing works about adaptive routing for NoC treat the routing adjustment as encapsulated self-adaptation mechanism at runtime, which is integrated at all router nodes and/or the network interfaces of IP cores. The mechanisms are distributed and each node itself performs adaptations using local or regional traffic/failure information without global coordination. Contrary, centralized solutions will operate on global system information, but may suffer from long latencies for traffic monitoring and routing updates. The research of this work was mainly inspired by the ATDOR solution presented in [8]. To the best knowledge of the authors, ATDOR is the first fully evaluated approach of centralized adaptive source routing for MPSoC. It offers a good solution to avoid high latencies of centralized path adaptations, but has different limitations. The presented solution of this work goes beyond these limitations and offers more flexibility, reuse, and fault tolerance. The main contribution of the presented approach targets the following changes and improvements: (1) *Redundant NoCs:* The ATDOR approach integrates a dedicated traffic aggregation network to provide global load information for the exclusive out-of-band hardware processor that calculates the path updates. This work generalizes approach to an exclusive infrastructure for system management information. Two redundant NoC work in parallel (Data-NoC and System-NoC) and will fully separate application data and system management data transport. Thereby, the design requirements will be separated too and each NoC can be optimized for its own traffic domain. (2) *Runtime-Configurable Clustering and Monitoring:* The centralized traffic monitoring and path adaptation is applied to spatial separated clusters and not as global approach. The dynamic clustering is managed at runtime by software agents and each cluster gets a defined fraction of the runtime workload assigned. Furthermore, the monitoring at the cluster-level is

 ISBN: 978-1-61208-246-2

configurable regarding the data capturing periods. Thus, monitoring as well as path adaptation can be configured for specific workload fractions. (3) *HW/SW-CoDesign:* To minimize the hardware overhead and raise the flexibility/configurability, a modular HW/SW-Co-Design is applied. The hardware only consists of needed mechanisms to realize the source routing and the monitoring of traffic information. The evaluation of monitoring information and calculation of routing paths is realized in software as centralized software agent inside a cluster. The combinations of clustering and migratable software further avoid the integration of a single point of failure. (4) *Full Runtime Integration:* The ATDOR solution is a two-phased approach that calculates the path updates in parallel to normal operations and afterwards distributes the path updates. Thereby, static workloads are assumed and calculation phases of 0.5 up to 1 ms. The proposed solution works continuously and intermediately distributes calculated path updates to provide more dynamic workload conditions.

The remainder of this work is organized as follows. Section 2 covers the related work. The third Section describes the conceptual part of the proposed solution and its global design aspects. A detailed evaluation regarding hardware costs, software timing, and simulated runtime characteristics will be given in Section 4. Afterwards, this work will be finalized with a conclusion and outlook for future investigations in Section 5.

II. RELATED WORK

Adaptive routing mechanisms in Networks-on-Chip mainly differ in the criteria, where the routing path will be adjusted, if the paths are minimal or not, and the scope of traffic information used for the decisions [1][2][3][4][5][6][7]. The majority of works focus a distributed adaptation at the router nodes under consideration of aggregated traffic information, targeting the direct proximity or regional scopes [3]. Thus, the router calculates the direction of the next hop of an incoming packet. The degree of adaptivity further varies between minimal and non-minimal route selection.

Ebrahimi et. al. presents two different distributed adaptive routing schemes (LEAR and CATRA) with exclusive solutions for the aggregation of traffic congestion information. In LEAR [9] the neighboring router nodes share their congestion states via one additional wire per link direction in the range of one hop. The adaptive routing is non-minimal. A more complex and irregular congestion information aggregation network for trapezoidal multi-hop regions is used by CATRA in [10], where the number of additional wires per link corresponds to the width/height of the 2D-Mesh topology. The applied routing is minimal. A solution between the complexity of LEAR and CATRA is presented by Rantala et. al. [11] using 2 additional wires per link direction for buffer level information sharing with direct-neighbor nodes to find non-congested minimal paths. Similar regional congestion information aggregation like CATRA can be found in the minimal adaptive routing strategies of RCA [12] and DBAR [13]. RCA offers different regional scopes and uses 8 up to 16 additional wires per link, while DBAR needs 8 wires per link for the sharing of congestion information. 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. in [14]. 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 complete system management is event-driven and focusses the runtime optimization of bandwidth utilization. A centralized adaptive routing called ATDOR is presented in [8]. The centralized path management works as additional hardware resource with a fixed coupling to the NoC through an out-of-band traffic monitoring aggregation network using 4 additional wires per link. Path updates will be calculated as a hole and in parallel to NoC operation under consideration of the global traffic situation and include the end-to-end (E2E) switching between XY and YX routes at the network interfaces of the IP cores. Path updates will be distributed over the normal NoC or an exclusive infrastructure. In [15], Cho et. al. suggests a Path-Based, Randomized, Oblivious, Minimal Routing (PROM) strategy. This solution contains a distributed local path adaptation at every router node with randomized decisions, based on probabilities. Especially, the simple O1TURN [15] technique of randomly toggling between XY and YX paths at the network interface shows optimal load balancing results. Kim et. al. provides a two-staged solution of source-initiated distributed path discovery and maintenance to increase the fault tolerance of NoC-based MPSoCs [16]. In [17], Asad et. al. presents a quite similar predominant routing approach using flooding based path exploration mechanisms. Routes between source-destination-pairings will be discovered and maintained using hop-restricted flooding techniques. Traffic optimization is out of scope and no hardware costs are presented. All of these adaptive routing mechanisms will evaluate the shared traffic information locally at each routing node using special hardware. Furthermore, the integration of virtual channels (VC) is needed to provide deadlock-freedom.

A comparable solution to LEAR, that applies minimal routing without VCs but using proximity traffic information is DyXY, presented in [18]. A similar solution, called DyAD, can be found in [19]. A different approach targets the selection of different routing algorithms for specific application workloads or flows at design time. At runtime the routings tables will be changed globally or regional optimized for the current of workload. In [20], Moreno et. al. proposes a routing scheme called planned source routing (PSR), which is a design time solution for traffic optimization. In [14][15] Palesi et. al. provides the APSRA technique including precompiled runtime adaptation of routing algorithms for workload specific demands.

 ISBN: 978-1-61208-246-2

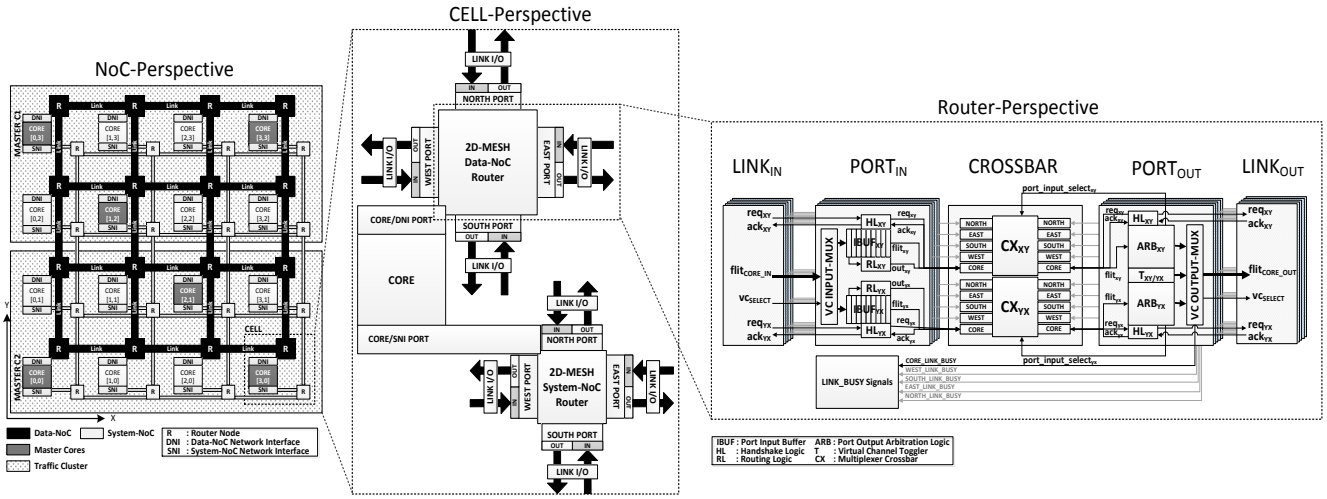


Figure 1. Overview of the targeted Network-on-Chip solution at different implementation levels

III. ADAPTIVE ROUTING CONECPT

The targeted NoC topology is a wormhole-switching 2D-Mesh, where two separated NoCs, called System-NoC and Data-NoC, work in parallel (see Figure 1). 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) of a path and finally reaches its destination, where the contained payload data will be processed. As flow control a hop-based request/acknowledge-scheme (req/ack) is applied. 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 a XY/YX-Routing that will be adapted at the network interface (PATH-LUT at DNI), higher link data width (64-Bit or higher) and more complex resource functionality, while the System-NoC works with reduced link data width (7-Bit for a 8x8 NoC), minimal input buffering (one flit) and XY-Routing. To provide deadlock-freedom for the path adaptations at the Data-NoC at least two virtual channels (VC_{XY} and VC_{YX}) for both selectable path configurations must be integrated [8]. As illustrated in Figure 1 and 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 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 Slave and Master. A Master-CELL is suited with special hardware resources and software agents to manage a CLUSTER regarding the traffic monitoring and path updates. Slave-CELLs are dynamically grouped into a CLUSTER by a corresponding Master-CELL (see 4x2 C1 and C2 at Figure 1). These CLUSTERs are the fundamental components. Inside a CLUSTER the software agent is able to migrate between the existing Master-CELLs. The number of potential Master-CELLs is defined at design time.

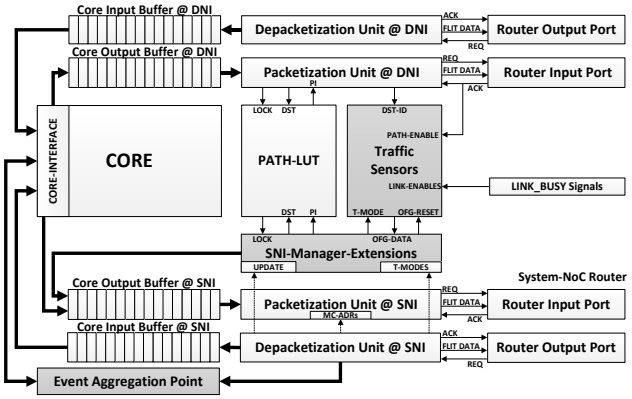


Figure 2. Overview of the Network Interfaces and IP core integration

The communication at the Data-NoC integrates an end-to-end based path adaptation concept, which is coordinated by the software agent of the CLUSTER. If the CORE sends a data through the Data-NoC it pushes a packet to the output buffer of the DNI. The packet header contains the position information of the source (SRC) and the destination (DST) as 2D-Mesh position. The information if the packet should take the XY or the YX path to the DST is stored at the PATH-LUT and needs one bit per destination in the NoC. Thus, for the n×n NoC the PATH-LUT contains (n^2-1) registers. At the packetization unit of the DNI the direction bit will be read out from the PATH-LUT and added to the packet header. Afterwards, the packet will be pushed to the corresponding VC and this won't change until the packet has traversed all router and links on its path to the DST. Thus, the VC assignment is static. At the router-level both

 ISBN: 978-1-61208-246-2

VC have an equal prioritization with a flit-based time-division-multiplexing (TDM). This will be managed by a special TDM unit (see $T_{XY/YX}$ in Figure 1). If two concurring packet of different VC will be routed to the same output port, the TDM unit toggles the selection signal at the output multiplexer every 2 clock cycles, because each flit needs this period to pass the link to the input port of the next router due to the hop-based req/ack flow control. Thus, each VC gets assigned the half of the link bandwidth and the transported packets will not run into a blocking situation. If only one VC will access the output port it will get the full bandwidth assigned by the TDM unit.

The adaptive source-routing works centralized and is applied to the spatial scope of the CLUSTER. Thus, each software agent of a CLUSTER monitors the traffic situation and performs the path updates. The following sub-sections describe the specific details of the adaptive source-routing 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 via the System-NoC 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).

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. At this specific case for the adaptive source-routing uses one dedicated CTX-ID. 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 (see C1 and C2 at Figure 1). Moreover, this separation concerns the exclusive clustering for different workload fractions/applications and avoids interferences. Thus, for the traffic monitoring and adaptive source-routing inside the CLUSTER, full spatial workload fraction isolation can be realized. Each clustering context has its own configuration data. 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. The workload fractions and clustering is precalculated at the global agent that covers the runtime-based application mapping.

B. Traffic Monitoring

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

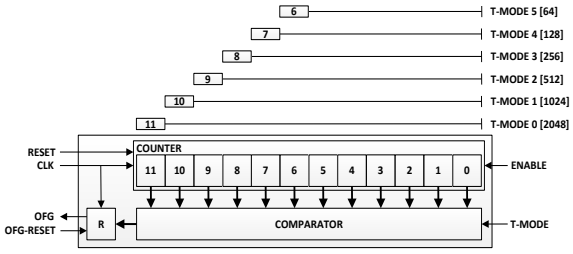


Figure 3. Basic Traffic Sensor concept

PATH/LINK-LEVEL: The basic traffic sensor is a simple combination of an external triggered binary counter and a configurable comparator (see Figure 3). 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 traffic monitoring can be obtained from Figure 3. 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 sensor solution is used in two different ways: (1) *LINK LOAD*: Each output port of a Data-NoC routing node (NORTH, EAST, SOUTH, WEST, and CORE) 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 multiplexing unit (see LINK_BUSY Signals). 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 (IR) at the path level inside the CLUSTER. Furthermore, one traffic sensor captures the overall injected traffic of the CELL. The ENABLE input is connected to the acknowledgment signal (ACK) of the DNI output (PATH-ENABLE). The number of needed traffic sensors depends on the maximum sizing of a CLUSTER the monitoring should work path-accurate for.

 ISBN: 978-1-61208-246-2

At the current progress, it 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 CLUSTER (Figure 2). Furthermore, they are grouped and located at the SNI of the CELL.

CELL-LEVEL: The OFG registers of all traffic sensors inside a CELL are connected to the SNI-Manager-Extension responsible for the Traffic Monitoring (see Figure 2). This functional unit generates the traffic monitoring packets for the System-NoC and works periodically. Thereby, the period 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, a finite state machine tests if at least one OFG is set. If true then 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.

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. 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 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 cycle has finished or during a current cycle. The duration of a cycle can be calculated by eq. (1) 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%).

$$t_{cycle} = \frac{r_{T-MODE}}{f_{System-NoC} \cdot k_S} \quad (1)$$

In example, for a T-MODE of 256 clock cycles at 1GHz the complete path accurate traffic situation of the CLUSTER can be capture in cycles 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. At this point the software agent at the Master-CELL has full information about the traffic situation inside the CLUSTER and is able to perform its path update evaluations.

C. Source-Routing

The routing path adaptation exclusively runs in software at the Master-CELL and focuses an incremental improvement for the traffic load of the current assigned workload fraction of the CLUSTER. After a finalized traffic monitoring cycle, the path adaptation starts and runs through all CELLs of the CLUSTER in a fixed order. For each CELL all observed paths will be evaluated in a fixed order and the following optimization procedure will be processed in the given sequence:

1. TEST: Initially the path will be tested for shared position coordinates, because if the path targets CELLs at the same row/column of the 2D-Mesh no minimal path alternatives are available and the path will be removed from the optimization list.

2. PREPARATION: The monitored injection rate (IR) of the current path is used to prepare the monitoring data for the new path evaluation run. The IR will be removed from the link load (LL) values of each link traversed by the path in its current assigned configuration (XY or YX).

3. EVALUATION: At this step the fitness for the XY and the YX configuration of the path will be calculated, which is simply the sum of LL values for all links traversed by a path. Afterwards, the lowest sum will be selected and its corresponding path configuration (XY or YX) will be assigned (minimal path load strategy.

4. UPDATE: After the evaluation of the path, the monitoring data will be updated with the injection rate of the path at the traversed link positions in the resulting configuration (XY or YX). This ensures the operation on the new traffic constellation for subsequent optimization runs of

 ISBN: 978-1-61208-246-2

other paths. Furthermore, if the path has changed an update packet will be sent to the CELL via the System-NoC to register the new path configuration at the lookup-table of the DNI (PATH-LUT). The update will be performed by the SNI and contains the bit flip at the PATH-LUT for the DST for the evaluated path.

Contrary to ATDOR [8], this solution only works continuously and triggers updates of single paths intermediately after processing. Furthermore, the timing depends on the monitoring interval, which can be configured individually for each CLUSTER and thus allows full adaptation for the traffic of the assigned workload fraction. The fixed ordered processing should lead to an incremental optimization of the traffic load balancing in case of static traffic patterns. For dynamic variations the traffic balancing will be slightly improved. The configurability of path updates for specific traffic pattern of the assigned workload fraction may be further improved by more selective ordering during path processing. This will be part of future investigations.

IV. EXPERIMENTAL EVALUATION

The evaluation of the presented solution was realized via software profiling and system simulations for operational performance as well as hardware synthesis for the cost approximation. Thereby, the basic Data-NoC design parameter configuration can be obtained from TABLE I.

TABLE I. DATA-NOC CONFIGURATION FOR SIMULATION AND SYNTHESIS

Parameter	Value
NoC Clock Rate	1 ns
Master-CELL CPU Clock Rate	0.5 ns
NoC Topology	2D-Mesh
Synthesized NoC-Size	8x8
Simulated Cluster Size	16 CELLS at 4x4 spatial shape
Data-NoC Input Buffer Depth	4 Flit
System-NoC Input Buffer Depth	1 Flit
Data-NoC Link data width	64 Bit
System-NoC Link data width	7 Bit
Average Link wire length [23]	0.83 mm
Wire width/spacing [23]	140/140 nm
# of I/O-Links in 8x8 NoC	224
# of Master-CELLs in NoC	32 (=50%)
Traffic Sensors per CELL	21
7-Bit Counter per EAP	336 (=16*21)
Simulated Data-NoC Packet Sizes	2 up to 9 Flit (uniform)
Traffic Pattern	random uniform distributed, transpose, hotspot (H=20%) and bit complement Hotspot Position = CORE [0,0]

A. Software Profiling

The main parameter for the evaluation of the ANSI-C based software agent is the computational latency/delay for the routing path calculation. Hence, the software profiling for the UPDATE/PREPARATION steps and the EVALUATION was carried out. As reference platform, a PowerPC 440 32-bit RISC CPU was used, which is implemented as hardcore on a Xilinx Virtex5 device. It is suited with 32-kB Data and Instruction Cache. The minimalistic Xilkernel from Xilinx served as basic integration environment for the centralized source adaptation software, which is implemented as standalone thread. To provide accurate timing information, the duration of path computations was measured in clock ticks via special timer commands.

TABLE II. SOFTWARE PROFILING RESULTS, GIVEN AS AVERAGE DELAY IN NUMBER OF CPU CLOCK CYCLES, FOR DIFFERENT NOC SIZES, AND MAXIMAL ALLOWED HOPS (MAXHOP) FOR ROUTING PATHS CALCULATIONS

NoC Size	Maxhop	PREPARE/UPDATE	XY/YX
4 x 4	-	36	132
4 x 4	4	34	118
4 x 4	3	32	104
8 x 8	-	46	174
8 x 8	8	42	157
8 x 8	4	34	119
8 x 8	3	32	105

For different 2D-Mesh NoC sizes (4x4 and 8x8), one million source-destination-pairings were generated and the routing path calculation processed. Further, uniform distributions with defined maximal hop distances (Maxhop) between these pairings were profiled. Afterwards, the average calculation delay, as number of passed CPU clock cycles, over all pairings was recorded. The results are included in Table II. The delay values for the full path calculation at different parameter variations (XY/YX) already contain the latencies for the PREPARE/UPDATE procedures. Observing the case of a 2GHz CPU running the thread with a budget of 20 %, the full XY/YX path calculation at the 4x4 CLUSTER without hop constraints is able to provide ~3000 single path updates per millisecond. The obtained profiling results were used for the timing accurate simulation of a complete workload simulation inside a CLUSTER.

B. Hardware Synthesis

The ASIC design flow was realized with the Synopsys™ DesignCompiler™ using the 45 nm Nangate FreePDK45 Generic Open Cell Library. The presented results in TABLE III show the total cell area costs for each of the functional components (SNI-Manager Extension, Traffic Sensors, EAP) and the router nodes of both NoC. Further, the NoC area costs of the complete NoC solution are given

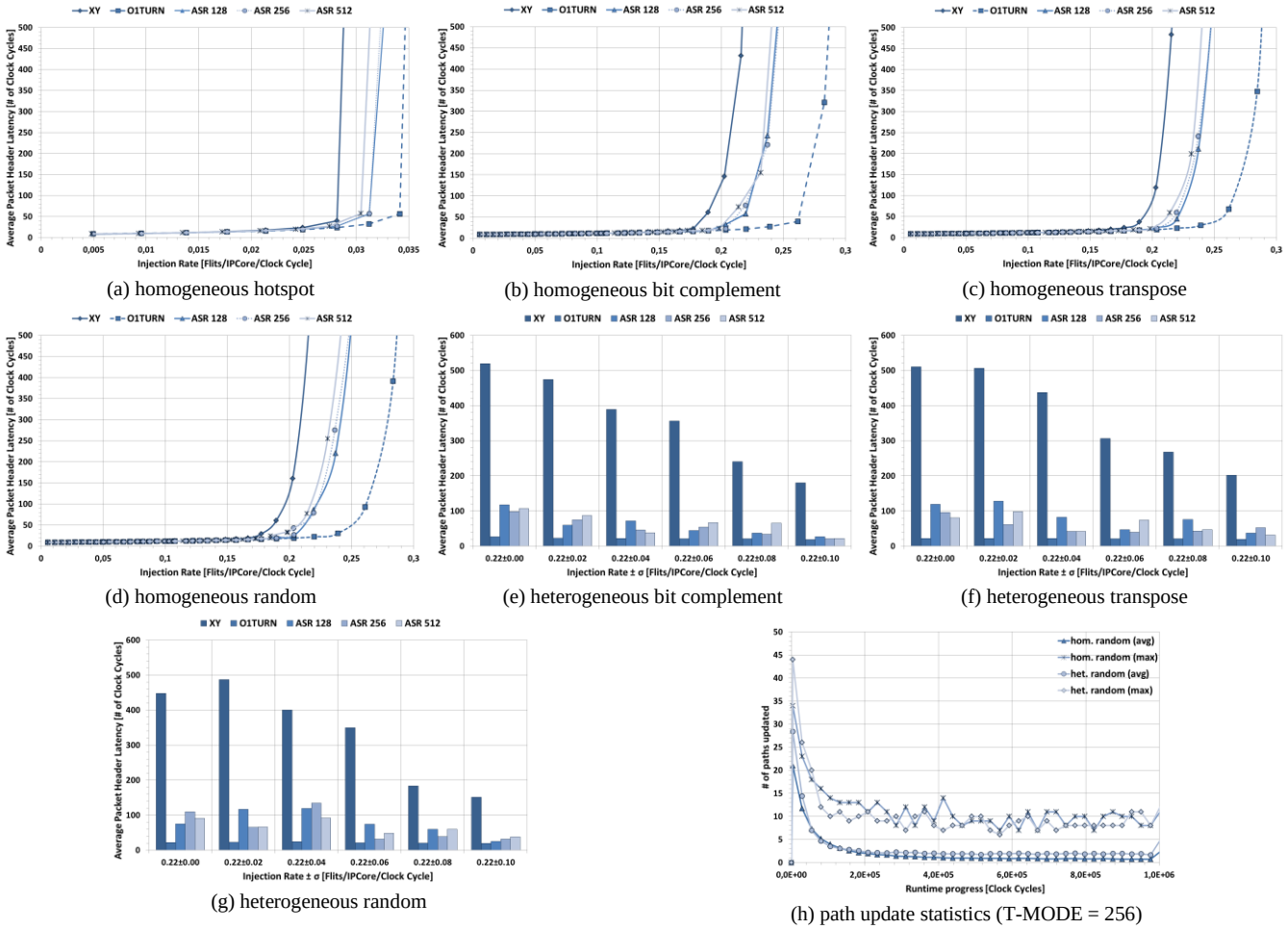
TABLE III. TOTAL CELL AREA (TCA) HARDWARE COSTS FOR SINGLE DESIGN COMPONENTS INSIDE A CELL AND AN 8X8 NOC AT ALL

	Total Cell Area	
Design Component	*CELL [µm²]*	*8x8 NoC [mm²]*
SYSTEM-NOC ROUTER	2783.69	0.178156
SYSTEM-NOC LINK	1510,6*	0.3383744
DATA-NOC ROUTER	40174.51	2.571169
DATA-NOC LINK	14641.2*	3.279628
SNI TRAFFIC LOAD EXT.	1511.94	0.096764
TRAFFIC SENSORS	2720.34	0.174102
AGGREGATION POINT	22230.68	0.711382
SUM OF ALL UNITS	**85572.96**	**7.3495754**

**area of a single I/O NoC-Link*

The targeted operational frequency was set to 1 GHz and met for all evaluated design cases. Regarding the hardware costs in the context of the final MPSoC that contain the

 ISBN: 978-1-61208-246-2



(a) homogeneous hotspot (b) homogeneous bit complement (c) homogeneous transpose

(d) homogeneous random (e) heterogeneous bit complement (f) heterogeneous transpose

(g) heterogeneous random (h) path update statistics (T-MODE = 256)

Figure 4. Summarized results of the 4x4 CLUSTER simulations for homogeneous task loads [(a), (b), (c), (d)], heterogeneous task loads [(e), (f), (g)], and path update statistics (h)

targeted amount of CELLs on a 45nm silicon die (areas: 280-400 mm² [24][25]) the relative overhead due to both complete NoC will be less than circa 2.6% down to 1.8%. Thereby, the total link area costs was estimated using the configuration of [23] as presented in TABLE I.

C. Simulation Results

The full system simulations target the maximum CLUSTER size (16 CELLs) at a spatial 4x4 shape with different workload configurations and monitoring cycles. This allows the fully path-accurate monitoring of the traffic situation inside a CLUSTER. Therefore, an own cycle accurate SystemC/TLM-based simulator was used. The dynamic workload consists of 10 tasks per CORE/CELL with a uniform random scheduling. Each task gets a fixed packet destination assigned depending on the four simulated synthetic traffic pattern as mentioned in TABLE I. For the hotspot pattern, H=20% of the tasks per core are communicating with the assigned CORE [0,0] in the lower left corner. Furthermore, two different traffic load strategies at the task-level were applied. At the homogeneous strategy each tasks has the same traffic injection rate, while the heterogeneous case works with an injection rate (IR) interval [IR±σ] and each task gets assigned a uniform random distributed injection rate out of this defined range. This allows the general evaluation of different traffic situations with more balanced or mixed loads at the path level. The monitoring was configured to work with different T-MODEs of 128, 256 and 512 at a scale resolution k_S=1%. This results in monitoring/path evaluation cycles of 12.8 µs, 25.6 µs and 51.2 µs. As references, the simple dimension-ordered XY-Routing and the probability-based adaptive O1TURN-Routing [15] were simulated. O1TURN toggles the VC (XY or YX) for each injection packet of a task and thus enforces a fully balanced VC utilization. The evaluation in [15] showed, that O1TURN offers the best average case throughput and latency. The newly introduced adaptive source-routing is called ASR <T-MODE>. The summarized results of all simulated parameter variations are depicted in Figure 4. The diagrams (a)-(g) contain the average packet header latency over varying traffic injection rates. The results were calculated as average from 100 simulations runs of workloads at each parameter constellation. Thereby, each simulation runs covers the

 ISBN: 978-1-61208-246-2

system operation time of 1 millisecond. For the homogeneous traffic load strategies in (a)-(d) the proposed ASR clearly outperforms the XY-Routing, but for all simulated traffic pattern it does not reach the performance of O1TURN and the latency improvements over XY-Routing ranges at circa 50% of the O1TURN results. Furthermore, the impact of reduced monitoring/evaluation cycles is smaller than expected. The simulation results for the heterogeneous tasks load strategy (see Figure 4 (e) - (g)) show that the advantage of O1TURN over ASR decreases with the increasing spread of task injection rates at highly intensive overall traffic loads (Traffic Load > 20 %) inside the CLUSTER. The difference between ASR traffic optimization performance of homogeneous and heterogeneous spread task loads relies on the resulting overall distribution of the traffic load inside the CLUSTER. At homogeneous task injection rates the traffic situation inside the CLUSTER will be more balanced and thus the dedicated assignment of specific path configurations is harder, because XY and YX path only slightly differs in the resulting fitness values (sum of link loads for each path). The higher the spread of task injection rates becomes the delta of these values increase, because the overall traffic situation becomes more heterogeneous. The diagram in Figure 4 (h) plots the registered path updates (average and maximum) at the homogeneous and heterogeneous case for the random traffic pattern over the proceeding operation time of the workload. Both cases show a similar convergence behavior at the maximum and the average case. The results for the other simulated traffic pattern are quite similar, but the random pattern is the most interesting, because it generates the most equal balanced overall traffic load inside the CLUSTER. As expected, the number of path updates reaches its peak at the beginning of operations. After the exponential decrease it works continuously at low update rates and the resulting computational demand for the software agent is suitable. The performing CPU at the Master-CELL of the software-agent was configured to operate with a clock frequency of 2 GHz.

V. CONCLUSION AND FUTURE WORK

The presented HW/SW-ASR concept, including clustering and path-accurate traffic monitoring, and its evaluation show that the intended runtime-configurable approach is feasible and superior to dimension-ordered XY-Routing. But the comparison to the O1TURN solution outlines its limitation as standalone solution. The additional hardware overhead is comparable to the solutions of [8][10][12][13], but has a more general focus on system management communication and makes workload specific traffic information available for global reuse. Thus, the presented cluster-based centralized ASR is only one integration aspect of the clustering and the System-NoC approach. The presented results brings the focus of future investigation on ASR to direction targeted by [14][15][26][27][28][29][30]. Specifically, the following aspects will be highly interesting:

Evaluate hybrid solutions that combine ASR and O1TURN. Both routing algorithms work at the end-to-end path level and can be merged without high additional hardware efforts. Thus, a more pattern-specific assignment to different CLUSTERs and inside these CLUSTER workloads may results in an attractive combination. Thereby, ASR will be pushed to support the planning of guaranteed bandwidth connections for specific application tasks or paths. Furthermore, the integration of failure handling for reliable NoC communication is a desired extension.

The globally available traffic information allows a better evaluation of the current system states and makes an interaction of runtime-based application mapping, routing strategy selection and spatial clustering more valuable. Furthermore, the integrated software-agents of ASR will support a better integration of these aspects, which targets runtime optimization at different abstraction levels and operational aspects.

REFERENCES

[1] W. J. Dally and T. B, "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 684–689.

[2] A. Jantsch and H. Tenhunen, *Networks on chip*, 1st ed. Kluwer Academic Publishers, 2003, p. 312.

[3] 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.

[4] E. Salminen, A. Kulmala, and T. D. Hamalainen, "On network-on-chip comparison," *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pp. 503–510, Aug. 2007.

[5] 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.

[6] 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.

[7] B. A. Abderazek, "Basic Network-on-Chip Interconnection for Future Gigascale MCSoCs Applications: Communication and Computation Orthogonalization," *Information Systems*, pp. 1–7, 2006.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

 ISBN: 978-1-61208-246-2

[12] 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.

[13] 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.

[14] 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.

[15] M. H. Cho, M. Lis, K. S. Shim, M. Kinsy, and S. Devadas, "Path-based, randomized, oblivious, minimal routing," in *Proceedings of the 2nd International Workshop on Network on Chip Architectures - NoCArc '09*, 2009, p. 23.

[16] Y. B. Kim and Y. Kim, "Fault Tolerant Source Routing for Network-on-chip," in *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, 2007, pp. 12–20.

[17] A. Asad, M. Seyrafi, A. E. Zonouz, M. Soryani, and M. Fathy, "A Predominant Routing for on-chip networks," in *2009 4th International Design and Test Workshop (IDT)*, 2009, pp. 1–6.

[18] M. Li, Q. Zeng, and W. Jone, "DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip," *2006 43rd ACM/IEEE Design Automation Conference*, pp. 849–852, 2006.

[19] J. Hu and R. Marculescu, "DyAD: smart routing for networks-on-chip," in *Proceedings of the 41st annual Design Automation Conference*, 2004, vol. 04, p. 263.

[20] E. I. Moreno, C. A. M. Marcon, N. V. Calazans, and F. G. Moraes, "Arbitration and routing impact on NoC design," in *2011 22nd IEEE International Symposium on Rapid System Prototyping*, 2011, vol. 3, pp. 193–198.

[21] 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.

[22] 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.

[23] C. Hernandez, F. Silla, and J. Duato, "A methodology for the characterization of process variation in NoC links," in *Design, Automation & Test in Europe Conference & Exhibition (DATE'10)*, 2010, pp. 685–690.

[24] 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.

[25] 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.

[26] S. Azampanah, A. Khademzadeh, N. Bagherzadeh, M. Janidarmian, and R. Shojaee, "LATEX: New Selection Policy for Adaptive Routing in Application-Specific NoC," *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pp. 515–519, Feb. 2012.

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

[28] 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.

[29] 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.

[30] A. Kumar, B. Mesman, B. Theelen, H. Corporaal, and Y. Ha, "Analyzing composability of applications on MPSoC platforms," *Journal of Systems Architecture*, vol. 54, no. 3–4, pp. 369–383, Mar. 2008.

 ISBN: 978-1-61208-246-2