The Invasive Network on Chip - A Multi-Objective Many-Core Communication Infrastructure

Jan Heisswolf *, Aurang Zaib †, Andreas Weichsgartner ‡, Martin Karle *, Maximilian Singh *, Thomas Wild †, Jürgen Teich ‡, Andreas Herkersdorf †, Jürgen Becker *

* Karlsruhe Institute of Technology, Germany, {heisswolf, becker}@kit.edu, {martin.karle, maximilian.singh}@student.kit.edu
† Technical University Munich, Germany, {aurang.zaib, thomas.wild, herkersdorf}@tum.de
‡ University of Erlangen-Nuremberg, Germany, {andreas.weichsgartner, teich}@cs.fau.de

Abstract—Invasive computing is a novel paradigm for massively parallel computing systems. The core objective of invasive computing is to support resource-aware programming through all layers of a system. An invasive application has the ability to explore and dynamically spawn its computation to neighboring processors, taking into account the status of the underlying architecture. The applications can exclusively request the reservation of hardware resources in a phase named invasion. The Invasive Networks on Chip (i-NoC) supports this paradigm by enabling the invasion of communication resources according to the requirements of applications. The monitoring infrastructure of the i-NoC is used by the software layers of the system to determine the location and availability of the resources that are invaded. This work gives an overview on the i-NoC hardware which is required to support invasive computing from the communication perspective. It introduces the i-NoC as a scalable, multi-objective many-core communication infrastructure and presents a concept that takes communication monitoring information into account for the invasion of communication resources. The presented results show how invasion of communication resources can help to improve the performance of a parallel application and reduce its communication energy footprint. Synthesis results investigate the implementation costs of the monitoring infrastructure introduced to take proper invasion decisions.

I. INTRODUCTION

In recent times, continuously increasing transistor densities enabled the integration of large numbers of cores on a single chip. Multi-processor System on Chip (MPSoC) architectures, originating from this development, have been in the market for almost one decade. Most of the available Systems on Chip (SoC) utilize buses and point-to-point connections for on-chip communication. Centralized arbitration of such interconnects limits their scalability and thus their usability as communication infrastructure of next generation many-core architectures with hundreds of cores [1]. Thus, Networks on Chip (NoC) have been introduced to mitigate the communication bottleneck. They offer better scalability by deploying distributed arbitration schemes for on-chip communication [2]. NoC based multi-core architectures have already proven their effectiveness [3], [4], [5]. Intel’s "Single-chip Cloud Computer" (SCC) is one example for an existing NoC based architecture [6]. It is a multi-core architecture that consists of 24 tiles connected through a mesh network with two Pentium cores per tile. The major challenge from the software perspective is the exploitation of the massive parallelism provided by NoC-based many-core architectures. New strategies need to be devised for managing and programming such systems in an efficient way. Invasive computing [7], [8] introduces a novel computing paradigm to facilitate application level parallelism in many-core architectures. Its main idea and novelty is to support resource-aware programming from application, operating system as well as architectural aspects. An invasive application can dynamically adapt its resource requirements by keeping in view not only the application level parallelism but the current utilization of the architecture as well. Invasive applications can dynamically reserve computation, communication and memory resources in a phase called invasion. The concepts conjunct for invasive computing target large systems, thus the underlying hardware architecture needs to be scalable. More details on invasive computing are given later. Extensions in the state of the art architecture are required to support the principles of invasive computing. The invasive Networks on Chip (i-NoC) enables the support for invasive principles with respect to communication. Therefore the i-NoC needs to provide a scalable communication infrastructure. The invasion of communication resources is realized through exclusive reservation of communication resources for an application. In addition, the status of the communication infrastructure is taken into account for the decisions taken during the invasion phase. Therefore a communication monitoring system is provided by the i-NoC. This work gives an overview on the invasive principles from the communication perspective. It describes the i-NoC as a scalable communication system that combines different concepts for reservation of communication resources with respect to invasive computing. The i-NoC monitoring infrastructure is introduced to enable appropriate decision making for resource allocation during the invasion phase of an application. This work does not claim to introduce novel hardware mechanisms for NoCs. It shows how existing concepts of connection oriented communication can be used to enable the invasion of communication resources within a scalable multi-objective many-core NoC. A monitoring infrastructure is used to enable distributed decision making during the invasion phase. The presented results show how the proposed concept for communication resource invasion can be used to improve application performance and reduce the communication and energy footprint of a distributed application. The rest of this paper is organized as follows: In Section II background and related work are described. Section III gives a short introduction on invasive computing, the invasive architecture and the role of the i-NoC within the invasive computing concept. The hardware implementation of the i-NoC including its monitoring infrastructure is detailed in Section IV. A case
study using a matrix multiplication on a prototype of an invasive architecture as well as ASIC and FPGA synthesis results are presented in Section V. Section VI gives a conclusion and summarizes our next steps.

II. RELATED WORK

NoCs provide a scalable interconnect for many-core architectures. While bus based interconnects using centralized arbitrated scheme do not scale beyond a certain number of cores, the arbitration inside the NoC can be distributed to the level of individual routers. In addition, different network topologies, like the widely used mesh topology, can be easily scaled to higher network sizes. Besides the idea to connect the different computing resources through a network, the implementation and realization can be quite flexible. This can range from packet to circuit switching, different routing protocols, network topologies or Quality of Service (QoS) guarantees. In the following text, some of the state of the art NoC architectures are described with their specific characteristics. Millberg et al. present the Nostrum NoC in which the Guaranteed Bandwidth (GB) support is augmented in addition to the Best Effort (BE) traffic delivery [9]. A concept referred as Looped Containers is applied to give guarantees for throughput. The approach provides limited run-time flexibility because the Virtual Channels (VCs) used for the Looped Containers are not established in a fully dynamic manner. Goossens et al. present the Æthereal NoC architecture which also offers both Guaranteed Service (GS) and BE traffic support [10]. GS support requires resource reservation to provide worst case bandwidth guarantees with bounded latency. BE traffic exploits the NoC capacity which is not utilized by GS traffic. Nostrum and Æthereal use Time-division multiplexing (TDM) based global synchronization, which leads to unused time slots and therefore reduces the overall throughput. Bolotin et al. present QNoC that provides QoS traffic guarantees with four service levels of different priorities [11]. Preemptive scheduling is used to schedule traffic corresponding to the respective service level. In such a scheduling scheme, the traffic of higher priority always limits the one with lower priority. Bjerregaard and Sparso propose a clock-less NoC named MANGO, which offers GS and BE support through two separately implemented routers [12]. The GS routers forward data streams on statically programmed point-to-point connections, whereas the BE routers are responsible for connection-less packet based traffic.

In contrast to static assignment of GS and BE traffic to communication resources, which is dominant in existing work, the i-NoC provides a flexible communication infrastructure that enables run-time resource sharing between GS and BE communication. This in turn enables an efficient utilization of the communication infrastructure independent of the amount of GS and BE traffic. In contrast to existing work, support for GS communication is not addressed isolated. It is combined with a monitoring infrastructure that helps to observe the communication in a distributed way and enables distributed decision making while establishing new GS connections that are used for the invasion of communication resources. Existing work on GS communication was only demonstrated using simulation results based on either synthetic applications or only discussed from a theoretical perspective. To the best of our knowledge this is the first work that demonstrates the benefits of GS communication on a real NoC-based architecture prototype using a real world application.

III. CONCEPT

In order to envision the functional requirements for the i-NoC, an overview on the invasive principles, the invasive architecture and the invasive communication concept is given in the following.

A. Invasive Computing

The main idea and novelty of invasive computing [8] is to introduce resource-aware programming. A given program gets the ability to explore and dynamically spread its computations to neighbor processors. The application requests for additional resources in a phase called invasion. During the invasion phase the run-time system needs to search for the resources that fit to the application requirements. Therefore, it takes the status of the underlying hardware into account. This includes the status of the computation, communication as well as memory resources, since all of these resources can be invaded. OS internal information and hardware monitors are used to get the required status information in a large distributed architecture. The last step of the invasion phase is the reservation of the resources for the application. If the invasion request was successful, the application can spread its computation in a phase called infection. Once the program terminates or if the degree of parallelism decreases, the program may enter in a phase called retreat to release resources and resume execution with reduced parallelism or even sequentially.

B. Invasive Hardware Architecture

An incarnation of an hardware architecture supporting invasive computing is named InvasIC. It is a heterogeneous tile based MPSoC architecture consisting of processing tiles including loosely coupled RISC processors, Tightly-Coupled Processor Arrays (TCPAs) [13] and reconfigurable RISC cores (i-Cores). It also includes memory and I/O tiles, which establish the interface of processing tiles to the external memory and I/O peripherals. In order to reduce the latency of memory accesses, a hierarchy of caches as well as Tile Local Memories (TLM) are realized. External memory accesses, I/O communication and direct communication between the processing elements are directed through the i-NoC to the respective tiles. The invasive Network Adapter (i-NA) is the interface between the tile local bus and the i-NoC routers. Figure 1 gives an overview of the envisioned architecture. A detailed overview of an invasive architecture is given in [14]. All aspects of the architecture, especially the i-NoC are designed with focus on scalability and decentralized resource management. Details of the i-NoC and i-NA are provided in Section IV.

C. Invasive Communication Concept

In order to facilitate the invasive principles from the communication perspective, the communication infrastructure needs to support the invasion of communication resources by applications. To ensure appropriate application mapping and invasion decisions, a monitoring infrastructure is required that provides the current status of the communication system. This information is taken into account while mapping of applications to decide which resources shall be invaded by the application. Taking the status of the communication hardware into account helps to ensure successful invasion of communication resources.
Figure 2 gives an overview about the process required for invasion of communication resources. An invasion phase is initiated by an invasion request, raised by a new application which wants to start its execution or by an existing application which needs to increase its level of parallelism. The invasion request is forwarded to an instance of the invasive Operating System (OS) named OctoPOS [15]. The request includes a description of the requested resources. The OS instance now needs to search resources that fit to the requirements of the application. At that point, OS internal occupancy information can be taken into account for the utilization of computation and memory resources. These information are known from the history of previous invasion phases and from the resource status after system initialization. The NoC load cannot be easily obtained through OS internal information, since the utilization of the NoC depends on the number of communication resources already invaded by other applications, i-NoC internal decisions (e.g. routing) and the communication behavior of the applications. To take proper mapping decisions from the communication perspective, monitoring information need to be taken into account. Therefore, the i-NoC provides a monitoring infrastructure which is described in Section IV-D. This information is taken into account while searching for resources that fit to the requirements of the application as shown in Figure 2. To enable scalability, an operating system instance called agent system [16] is held responsible for an application and the hardware resources claimed by the application. It only accesses the monitoring information in the surrounding region of the application. If appropriate resources are not available within that region, resources might be searched in the surroundings. Once resources are found, they are allocated for the application. For the allocation of communication resources, a reservation request is forwarded to the i-NoC through the i-NA as described in Section IV-B. If such an invasion request is successful, the application can be executed by infecting the computation resources.

The resource search phase as well as the allocation phase can fail (e.g. if a connection cannot be established successfully). This again leads to a fail of the invasion phase. To deal with such a situation, invasive applications are malleable and can continue processing with lesser number of resources. If an application has no resources at all, execution has to be delayed. Taking the i-NoC monitoring information into account helps to increase the chance of successful invasion.

IV. REALIZATION

The invasive network on chip is composed of the i-NoC routers and i-NA. The realization of these components is discussed in the following text. In Section IV-C, the communication mechanism through i-NA and i-NoC routers is illustrated through an example. The i-NoC monitoring infrastructure, used for decisions making during invasion phase, is discussed afterward.

A. Router

The i-NoC is a packet-switched NoC supporting both connectionless as well as connection-oriented communication. Connection-oriented GS support is used to provide hard guarantees with respect to throughput and latency. From the perspective of invasive computing, it enables the invasion of communication resources exclusively by an application. Connectionless BE transmission is used for communication over NoC which takes place without a preceding invasion of communication resources (e.g. main memory communication at OS boot up). In case there are no further communication resources which are available for invasion, BE provides a fallback solution for data access via the i-NoC. The two transmission strategies (GS and BE) share the router resources dynamically to enable management of NoC resources in a flexible manner. Decentralized routing is chosen due to its better scalability compared to source routing. Currently XY-routing and adaptive odd-even turn routing are available in the i-NoC to be selected at design-time. If odd-even turn is used, output port selection [17] is based on i-NoC monitoring.

The i-NoC uses wormhole switching based flow control to reduce the amount of buffers required inside the routers. The packets are segmented into equal sized flits. Inter-router flow control is credit-based. The basic structure of the i-NoC router is shown in Figure 3. The buffers of the i-NoC routers are located at the input ports. VCs are used to utilize the physical link bandwidth efficiently.
as proposed in [18]. Assignment of VCs is done dynamically by the router based on local decisions for both traffic types. A VC is assigned to an output port once it is allocated to a BE or GS transmission by adding an entry to the reservation table of the respective output port based on the routing decision. The arbitration between the VCs assigned to an output port of the router is done in a round robin manner. Weights can be assigned to increase the bandwidth of a GS connection, as described in [19]. The VCs are taken into account for arbitration only if data is available for transmission. By using this scheduling scheme, link utilization is maximized.

Fig. 3. Structure of the i-NoC router including its monitoring infrastructure

Scheduling of VCs either allocated for BE or GS communication is not prioritized, which ensures that GS connections may not completely block BE traffic unlike in most QoS supporting NoCs discussed in Section III. Therefore, it is ensured that GS connections never occupy all VCs. This provides a worst case bandwidth guarantee for BE communication. The round robin scheduling of each allocated VC provides delay and throughput guarantees for reserved connections. The used weighted round robin scheduling enables to adjust bandwidth and latency of reserved connections, as described in [19], [20]. The weight of each connection is thereby provided by the software as part of a communication resource invasion request.

B. Network Adapter

The i-NA provides an interface between the processing tiles and the i-NoC routers. Processing tiles have a local bus to facilitate communication between different peripherals within the tile. i-NA is also attached to this local bus. In order to access data over the i-NoC, the i-NA performs a protocol translation from bus to network packets. The basic structure of the i-NA is shown in Figure 4. The i-NA provides the necessary support for initiating the request for invading the communication resources over the i-NoC. For this purpose, i-NA contains Connection Manager which is responsible for triggering invasion of communication resources by setting up GS connections upon request of applications. In addition, there is a table named as VC Reservation Table to maintain the status of on-going GS connections. Traffic classifier is a component which directs the incoming traffic from tiles over network through GS or BE communication mechanism depending on the existence of a GS connection to the destination tile. In order to invade communication resources, the application writes a memory mapped register inside the connection manager module of the i-NA. Connection manager then generates a header flit and injects it into an available VC to establish an end-to-end GS connection over i-NoC. In case of successful reservation, the GS connection is granted to the application and the appropriate status is stored in VC reservation table. When an application owns the GS connection over i-NoC, the subsequent traffic is routed over the reserved connection by the traffic classifier to provide hard guarantees and improve communication performance. When an application wants to retreat from previously invaded GS connection, it writes to the memory mapped registers of the connection manager accordingly. Connection manager initiates the retreat of invaded communication resource by issuing a tail flit and updates the VC reservation table accordingly. Section IV-C contains further details of this mechanism.

In order to support the decision making for invading appropriate resources, the communication monitoring information is sampled in the i-NoC. i-NA provides the interface of the i-NoC monitoring registers to applications. It is accomplished by mapping monitoring registers values into the local address space of each tile. Monitoring information of other tiles are accessible via the i-NoC. This enables applications to access the monitoring registers and then decide on which resources should be invaded depending on i-NoC utilization. Details about the monitoring infrastructure are provided in the Section IV-D.

C. Data Transmission

The i-NoC supports communication between tiles through both GS and BE data transmission. An application running on a processing tile requests for a GS connection as part of an invasion request. The tile’s network adapter receives this request and triggers the reservation of the requested connection. Therefore, a special GS header flit, containing the destination address of the connection, is inserted into the input queue of the reserved VC in the i-NA and then forwarded to the connected router via the local port. The router receives the header, takes a routing decision and reserves a VC at the output port determined by the routing unit.
This mechanism is repeated in each subsequent router. Fully distributed decision making within the i-NoC reduces protocol overhead and enables scalability. Once the GS header arrives at the destination, i-NA sends a flow control message to the source (initiator) of the connection using a BE packet to acknowledge the connection establishment. After receiving this flow control acknowledge message, the initiator is allowed to inject data into the reserved connection. If a GS connection is denied (e.g., due to unavailability of a free VC inside a router), the outstanding acknowledge leads to a timeout in the source i-NA. The source NA now issues a tail flit which releases the reserved resources of the partly established connection. Then, the NA gives appropriate feedback to the corresponding application depending on the result of the invasion request by status registers located inside the i-NA. Connectionless BE traffic uses standard packet-based transmission without requiring any resource reservation. For BE traffic, a head, several body and a tail flit are injected cohesively without requiring an end-to-end flow control. The VC allocation policy implemented inside the i-NoC routers ensures that at least one VC is not reserved for GS communication, as detailed in [21]. This mechanism ensures that the communication between all tiles is always possible through BE traffic as a fallback solution. An extension of the allocation unit inside the router even enables to establish private sub-networks for specific applications at run-time. The realization of such virtual networks supported by the i-NoC is detailed in [22]. Figure 5 illustrates an example of GS and BE data transmissions in the i-NoC. Transmission 1 shows a GS connection setup through a GS header flit. Communication between two tiles through an established GS connection is shown by Transmission 2. It utilizes different VCs at different links as explained before. This increases the degrees of freedom during the connection setup which improves the chances of successful invasion. Transmission 3 represents a BE packet on its way to the destination through the i-NoC. As shown, BE packets only utilize a VC for a very short period.

![Example of BE- and GS data transmissions within the i-NoC](Image)

### D. Monitoring

The monitoring infrastructure of the i-NoC is an important feature, which is required to take appropriate mapping decisions with respect to communication during the invasion phase. The OS needs to search for a region within the invasive architecture where the communication requirements of the application are satisfied. Depending on the communication requirements, a region with a low utilization of the i-NoC may be selected by taking the i-NoC monitoring information into account. This can help to improve the communication performance of the application that needs to be executed. If communication resources need to be invaded by the application, the monitoring information can be used to search for an appropriate region within the architecture and for estimating the chances for successful invasion. Monitors of three different types are envisaged for providing the required information to the OS:

- **Link Utilization (LU):** Provide the information about the utilization of the links between routers and to the tiles. They are used to obtain the available bandwidth.
- **Virtual Channel Utilization (VCU):** Provide information about the utilization and availability of virtual channels. They are used to estimate if additional GS connections, used for invasion of communication resources, can be established.
- **Buffer Utilization (BU):** Provide the status of the buffer fill level. They can be used to detect over utilization and back pressure of existing GS connections. The BU monitoring is only used for detailed analysis of the i-NoC performance.

All monitors measure the respective utilization for a predefined period using counters which probe the respective signals within the i-NoC routers as shown in Figure 3. The captured values give an average of the utilization within the measurement period $P$. The value of $P$ can be defined at run-time between 1 and $P_{\text{max}}$. This enables to adjust the monitoring period according to the application behavior and OS requirements. In addition to this average monitoring, the VCU and BU monitors measure the peak utilization. This is the highest number of used VCs or buffer slots being utilized at the same time. A history of the last $N$ periods is stored within the monitoring units. This enables to access not only the latest monitoring values but also provides information about the evolution of the utilization during the past periods. The size $N$ of the history can be defined at design time. All monitoring information of a router is accessible via memory mapped registers, as explained in Section IV-B.

## V. Evaluation

The presented concept for invasion of communication resources in the i-NoC and the proposed monitoring infrastructure is investigated with respect to performance and energy consumption. Therefore, an FPGA-based prototype of the invasive architecture and the i-NoC was realized. A parallel integer matrix multiplication is taken as an example application. ASIC and FPGA synthesis results of a single i-NoC router and i-NA are used to investigate the implementation overhead of the monitoring infrastructure.

### A. FPGA prototype

An FPGA prototype of the invasive architecture was realized on the Xilinx ML605 development board. This board contains a Xilinx Virtex-6 LX240T FPGA. Due to the complexity of the invasive architecture, the realizable size of the architecture...
is quite small. A homogeneous two-tile architecture with two RISC cores per tile could be realized. The used RISC cores are LEON3 SPARC V8 processors [23]. A structure of the prototype is given in Figure 6. Each core has a separate L1 data- and instruction cache. Both cores share an L2 cache that is used for holding tile-external data. The tile local memory located in each tile is currently used to store the binary executable and frequently used data. A DDR memory controller is located in one of the tiles. The TLM of other tiles and the DDR memory can be accessed through i-NoC.

The parameters of the realized prototype are given in Table I. Due to the limited amount of the available BRAMs in the used FPGA, the size of the caches and memories was very limited. To have enough BRAMs available for the tile local memory, the sizes of the L1- and L2-caches had to be strictly limited. A large TLM is required to store the executable and to have some additional space left for frequently accessed data.

![Fig. 6. Homogeneous architecture prototype realized on the Xilinx ML605 development board](image)

### Table I. Settings of the Two Tile FPGA Prototype

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1D-Cache (Sets, LineSize, Lines)</td>
<td>2, 4, 4</td>
</tr>
<tr>
<td>L1I-Cache (Sets, LineSize, Lines)</td>
<td>2, 8, 8</td>
</tr>
<tr>
<td>L2-Cache (Sets, LineSize, Lines)</td>
<td>2, 32, 1</td>
</tr>
<tr>
<td>Tile Local Memory</td>
<td>512 KB</td>
</tr>
<tr>
<td>DDR Memory Size</td>
<td>1024 MB</td>
</tr>
<tr>
<td>NoC Link Size</td>
<td>32 bit</td>
</tr>
<tr>
<td>NoC VCs</td>
<td>4</td>
</tr>
<tr>
<td>NoC Buffer Size</td>
<td>128 Byte</td>
</tr>
</tbody>
</table>

The prototype was synthesized using the Xilinx PlanAhead flow. The target clock frequency of the prototype is 50 MHz. The i-NoC components could achieve much higher frequencies as shown later. The design consumes only 17% of the slice registers available on the LX240T FPGA. 51% of the FPGA Look up Tables (LUTs) are required for the design. The limiting factor for further increasing the number of tiles is the BRAM utilization, which is 85% for the two-tile prototype. A larger prototype is under development and will be realized using a larger prototyping platform, as detailed in [24].

### B. Case study

A parallel implementation of the matrix multiplication is used to show the benefits of communication resource invasion using the two-tile prototype of the invasive architecture. The matrix multiplication is written in a parametrizable way. Four variants of the algorithm are executed: The BE variant uses best effort communication, it does not invade any communication resources. GS uses invasion of communication resources in the i-NoC by guaranteed service connection that are established before data transmission. For both versions the source as well as the destination matrix is located in the DDR memory during processing. The two other variants are BE-PF and GS-PF. They also use BE or GS communication in the i-NoC respectively, but the data required for the current computation are prefetched to the tile local memory through a load/store software implementation. The prefetching procedure itself is also taken into account for the following measurements with respect to computation time and communication overhead.

Figure 7 summarizes the results obtained from executing the four variants of the integer matrix multiplication on the two-tile prototype. All variants are executed for four different sizes of the matrices up to 128 x 128 elements. A further increase of the matrix size is restricted by the TLM size of the prototype. For the presented results, the BE variant which does not invade communication resources is taken as a reference and compared to the GS version which invades the communication resources. Figure 7(a) compares the execution time of the different variants. The longest execution time was obtained for the 128x128 BE variant with an absolute value of 158.7 x 10^6 cycles. The invasion of communication resources (GS variant) has a speedup of up to 5.9% for larger matrices compared to the variant without invasion (BE). The improvement is significant, taking into account that the speedup only results from the invasion of communication resources. For small matrices where all data fit to the L2 cache, the speedup of GS communication is small due to the low amount of NoC communication. If prefetching is used, the NoC utilization and thus the performance by invasion of communication resources is less. A speedup of 0.5% can be obtained with prefetching of small matrices, if communication resources are invaded. For larger matrices, the performance improved through GS communication during prefetching has no consequence, due to the long computation time which follows. The benefit in the application performance results from the invasion of communication resources. This invasion reduces the communication latency due to the use of GS connections.

Another advantage of GS communication is the reduced network protocol overhead. This can be seen from the results presented in Figure 7(b). They show the communication resource utilization for the same scenarios. These results are measured using the proposed monitoring infrastructure of the i-NoC, introduced in Section IV-D. The monitors deliver accurate information about the utilization of the i-NoC. Consequently, they are used to observe the communication on the architecture prototype. It can be seen from the results that the invasion of communication resources reduces the utilization of the communication infrastructure heavily. For the variants where the source matrix is located in the DDR memory, (BE and GS) the invasion of communication resources can reduce the communication overhead by 27% to 32%. If the source matrices are prefetched to the TLM, a communication reduction of 32% can be achieved independent of the matrix size. For large matrices, prefetching can strongly reduce the communication overhead which cannot be seen from the relative numbers in the diagram (this reduction is not in the focus of this work). In general, the reduced communication overhead of GS communication compared to BE results from the fact that GS communication has no protocol overhead during transmission. This in turn results from the fact that no control information, such as source and destination address as well as packet length, need to be transmitted, if a connection is established.

Finally, the energy consumption for data transmission is analyzed for the given scenarios. Therefore, the Communication

---

**TABLE I. SETTINGS OF THE TWO TILE FPGA prototype**

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1D-Cache (Sets, LineSize, Lines)</td>
<td>2, 4, 4</td>
</tr>
<tr>
<td>L1I-Cache (Sets, LineSize, Lines)</td>
<td>2, 8, 8</td>
</tr>
<tr>
<td>L2-Cache (Sets, LineSize, Lines)</td>
<td>2, 32, 1</td>
</tr>
<tr>
<td>Tile Local Memory</td>
<td>512 KB</td>
</tr>
<tr>
<td>DDR Memory Size</td>
<td>1024 MB</td>
</tr>
<tr>
<td>NoC Link Size</td>
<td>32 bit</td>
</tr>
<tr>
<td>NoC VCs</td>
<td>4</td>
</tr>
<tr>
<td>NoC Buffer Size</td>
<td>128 Byte</td>
</tr>
</tbody>
</table>
Related Energy (CRE) consumption is introduced. CRE is defined as the energy consumption which is directly related to data transmission. The energy consumption of the idle router, which is 7.94 mW, is not taken into account due to the fact that this static amount depends heavily on the use of power saving techniques such as clock or power gating which are not in the scope of this work. To estimate the CRE, the dynamic energy consumed of an i-NoC router is measured by post-synthesis power analysis using value change dump (VCD) files resulting from accurately modeled GS and BE transmission scenarios to get the toggle rates. Afterward, Synopsys Power Compiler is used to estimate the power using the generated VCDs.

Figure 7(c) shows the power analysis for the GS transmission scenarios relative to the BE equivalent. If the source matrices are located in the DDR memory, the CRE can be reduced by 22% (large matrix) to 27% (small matrix) by invading the communication resources. A constant reduction of 27% in energy consumption could be achieved for the prefetching scenario, independent of the matrix size.

The presented results clearly show the benefit of invading communication resources in terms of application speedup, NoC utilization and power consumption. The i-NoC monitoring was used for detailed communication analysis.

### C. Synthesis Results

The benefits of invading communication resources have been clearly shown by the previous case study. In the following, the implementation overhead of the monitoring infrastructure which is required to realize the concept introduced in Section III, is investigated. Therefore, two baseline versions of the i-NoC router are used to investigate the implementation costs of the monitoring infrastructure: Base32 has a link size of 32 bit. It is the router version used for our FPGA prototype in combination with LU and VCU monitoring. However, that link size and the resulting bandwidth would be too small for an ASIC implementation. Thus, a second variant (Base256) with 256 bit link size is used to estimate the overhead of a router that fits to the requirements of an ASIC design with a large number of cores, such as Intel’s Single-Chip Cloud Computer [25], which has also a link size of 256 bit. In addition, the area estimates for a 32 bit i-NA are provided. The considered i-NA contains the support for communication resource invasion and monitoring infrastructure support. The current i-NA implementation only supports a link size of 32 bit due to its AHB interface used for the FPGA prototype.

The synthesis results for an ASIC and FPGA realization of a single 5-port router and a network adapter are given in Table II. For the ASIC synthesis a 45 nm standard cell library from TSMC (tcbn45gsbwpcw) with worst case operation conditions is used. Synopsys Design Compiler (F-2011.09-SP4) is taken for synthesis. The Power Compiler is used to estimate the energy consumption based on realistic VCDs. The target device for the FPGA realization is the Xilinx Virtex-6 LX240T FPGA which is also used for our prototype.

The results for the small router (Base32) show that adding link utilization (LU) and virtual channel utilization (VCU) monitoring increases the area consumption by 7.5% and the power consumption between 12.3% and 16.3% for the ASIC implementation. The critical path and thus the clock frequency is not affected by the monitoring infrastructure. For the 32 bit FPGAs realization clock frequency slightly drops when adding LU and VCU monitoring. The number of registers required for the monitoring units increases the total number by 29.8%. The i-NoC router is implemented in a way not to use any BRAMs to save them for other components of the prototype.

The more meaningful results are the ones for a link size of 256 bit. Such a link size better reflects the implementation of a large and scalable architecture addressed by the i-NoC. The synthesis results in Table II show that for the Base256 ASIC implementation, which is the most realistic variant, LU and VCU monitoring increase the area consumption of an i-NoC router by only 2.1%. Power consumption is increased by 5.1% at maximum. The achievable clock frequency of 1.5 GHz is not affected. For an FPGA prototype the clock frequency drops slightly if LU and VCU monitoring is added. The register utilization is increased by 16.6%. The LUT utilization remains constant.

Buffer utilization monitoring has a much higher implementation overhead compared to LU and VCU monitoring. In contrast to LU and VCU monitoring it is not required to take appropriate invasion decisions at run-time. These decisions can be taken based on link and VC utilization monitoring. Thus, BU monitors should only be included into an architecture if detailed back-pressure analysis for individual applications is required.

In general, it can be summarized that LU and VCU monitoring have an acceptable implementation overhead for a realistic ASIC implementation of a 256 bit i-NoC router. In addition synthesis results of a 32 bit i-NA for both FPGA and ASIC are given in Table II. The NA has a slightly lower area and power consumption than the ASIC implementation of the router itself.
VI. CONCLUSION & FUTURE WORK

In this work a concept for the invasion of communication resources is presented which takes monitoring information into account. The invasive network on chip is introduced as a scalable general-purpose communication infrastructure for many-core architectures that supports the concept of communication resource invasion. The introduced communication monitoring infrastructure is used to observe the utilization of the \textit{i}-NoC. The invasive network adapter provides a software interface for the invasion of the \textit{i}-NoC resources and enables efficient access of the monitoring data.

An FPGA based prototype was realized to investigate the presented concept using a parallel implementation of a matrix multiplication. The presented results obtained from execution on the prototype show that invasion of communication resources can accelerate the application performance by up to 6\% and reduce the energy consumption for communication by up to 27\%. ASIC synthesis results obtained for an \textit{i}-NoC router with 256 bit link size depict a low area overhead of only 2.1\% and an additional power consumption of only 5.1\% for the monitoring infrastructure. The clock frequency as well as the performance of the \textit{i}-NoC is not affected by the monitoring infrastructure.

The processing of \textit{i}-NoC monitoring information within the invasive operating system is currently under development. The work for building a large heterogeneous prototype of an invasive architecture is in progress. Investigating the \textit{i}-NoC monitoring based decision making and the resulting invasion of communication resources by the OS on a larger architecture including more complex applications will be done in the near future.

ACKNOWLEDGMENT

This work was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center “Invasive Computing” (SFB/TR 89).

REFERENCES