Service-oriented Approaches for the Operation of large on-chip Networks

Claas Cornelius Hendrik Bohn Dirk Timmermann
Institute of Applied Microelectronics and Computer Engineering
University of Rostock, Germany
{claas.cornelius | hendrik.bohn | dirk.timmermann}@uni-rostock.de

***Abstract*—Service-Oriented Architectures (SOA) specify standards that allow the connection and interaction of heterogeneous hardware and software components in networks of any kind. Similar aims have to be encountered by the emerging Network-On-Chip (NOC) approaches to design future integrated circuits. That is, diverse components (called resources) offer a large number of services which have to be managed efficiently during the operation of such a chip. The arising problems, like for instance service discovery or availability, are addressed by SOAs. Hence, this paper discusses whether and to what extent service-oriented approaches can be adopted for the operation of large on-chip networks with heterogeneous resources and its services. Thereto, simulation results are presented to evaluate different architectural implementations.**

I. INTRODUCTION

The continuous reduction of transistor dimensions has enabled the semiconductor industry to achieve ever higher logic complexity and increased performance, at reduced costs per transistor. The International Technology Roadmap for Semiconductors [1] projects that this so called scaling is about to continue. Several billion transistors on a single chip with feature sizes below 50 nm, operating at frequencies in the 10 GHz range are expected by the end of this decade. Unfortunately, the scaling does not affect all properties likewise which causes new problems and challenges that need to be faced at various abstraction levels and in different design steps.

Networks-On-Chip (NOC) have been proposed as a new design methodology to overcome the large number of issues of nanometer technologies [2][3]. A NOC consists of miscellaneous independent resources - e. g. general purpose processors, memory, decoders, or I/O - that are connected to a global on-chip network. Interfaces encapsulate the global network and the resources from another and enable the communication among the various resources. To this day, lots of effort has been put into designing efficient links [4], working out advantages of different topologies, and developing diverse types of routers [5][6]. First prototypes of functional NOC implementations have also been presented with a relatively small number of resources for testing or specific applications [7][8]. However, a large number of independent resources is expected to be implemented on a single chip in the future [9] which comes along with new challenges for the operation of such chips. Therefore, to simply apply the widely used mechanisms of single-core systems does not seem promising for NOCs. To this day, only little effort has been put into questions of handling very large general purpose NOCs and managing the plenty of services provided by the resources.

A possible solution might be service-oriented approaches – that are well-known and established in computer science – which offer interesting characteristics that could also ease and improve the operation of NOCs. Service-Oriented Architectures (SOA) specify standards for the seamless connection and interaction between heterogeneous hardware and software components [10]. Abstract interfaces and determined communication patterns are used to deal with issues like finding, addressing, and using remote services as well as managing connections. Such issues have to be addressed in NOCs as well. The main difference is that NOCs have strong requirements on the network and the additional need for the awareness of physical limitations like power consumption, rather low bandwidth, or thermal distribution. By contrast, SOAs rely on widely distributed networks with weak requirements such as the Internet. For example web services, the most well known application of an SOA, use a set of several protocols and communication stacks which can be applied in numerous ways to achieve a certain aim. Indeed, this allows the coverage of issues ranging from addressing over management to security but also generates a large overhead needed for the implementation.

This paper elaborates the basic concepts of SOAs and discusses the possibility to adopt those mechanisms for the operation of large NOCs with its plenty of available services. Thereto, section II introduces related work in connection with on-chip networks and service-oriented architectures. A discussion of properties and possible advantages follows in section III before section IV concludes the paper.

II. RELATED WORK

This section introduces the basic ideas of NOCs and SOAs to ease the understanding of the discussion in section III where both concepts will be brought together.

A. Networks-on-Chip

Networks-On-Chip (NOC) have been suggested as an option to cope with issues of bus-based or point-to-point connected systems in future technologies. This paper focuses on open questions for the management and operation of such NOCs, as a multitude of ideas and solutions has already been proposed for the lower layers and its functionality been demonstrated. For instance repeater, booster, and current sensing have been suggested

This work is supported by the German Research Foundation (DFG) under grant number GRK-466, the SIRENA project in the framework of the cooperative R&D program ITEA, and BMBF.

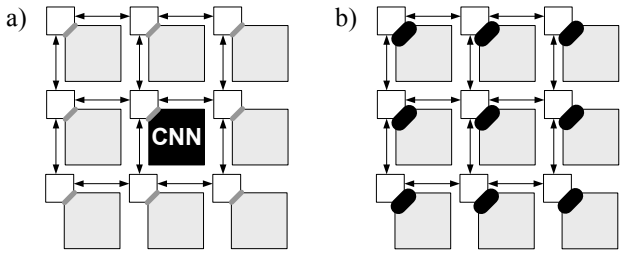

Figure 1. Simple example of a 3x3 mesh network a) with a Central Coordination Node (CCN) and b) with distributed system control

for signal transmission [4], coding and error correction schemes have been analyzed as well as virtual channel routers [5][6] have been integrated in various topologies (e.g. mesh, torus, fat-tree) applying diverse routing schemes (e.g. XY, adaptive, deflective). Figure 1 a) depicts a simple example of a 3x3 NOC. The shown mesh topology is widely used [2][7][9] due to its regularity but the considerations in this paper can comparably be understood for other topologies as well.

The question of how to operate a NOC with a large number of resources challenges designers because established mechanisms and approaches for single-core solutions might not be transferred to NOCs without major changes and advancements. Such an operation has to include aspects of load balancing, power issues, quality of service requirements, and many more administrative tasks [3]. To do so, control messages have to be sent within the network to signalize for instance congestion of data messages, an arising thermal hot spot, or the unavailability of a function. Appropriate actions have to be taken as a reaction to these messages which result in another set of messages in the network. A possible reaction could be the adjustment of the clock frequency of a resource or the re-routing of a data stream.

Most current implementations apply a Central Coordination Node (CCN) with an operating system that has global awareness of the overall network [5] as shown in figure 1 a). Thereby, method invocation on remote resources is performed by the use of principles from distributed object systems. That is, a local function call is wrapped with its parameters into a message that is sent over the network to the remote resource. There the function call is unwrapped, executed, and sent back with the according answer of that called method. Nollet et al. [11] presented such an approach with a centralized operating system that can manage task mapping and flow control. Thereto, the resources are polled to achieve information on packet statistics. Based on these figures appropriate actions like reducing the packet injection rate or changing the packet routes can be taken. A similar approach was also developed by Hecht et al. [12] where an extended Linux system is used. Here, simple and static

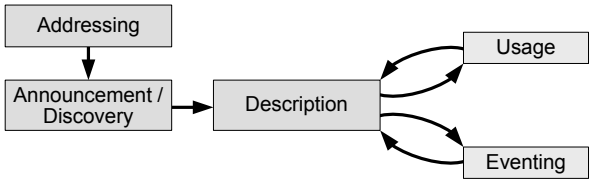

Figure 2. General stages in the life-cycle of a service in a Service-Oriented Architecture (SOA)

dimension-ordered routing is used but the operating system allows the dynamic reconfiguration of resources. That is, the execution of a software task running on a processor can be accelerated by reconfiguring another resource (meaning hardware) for that specific task. A very different approach was discussed by Benini et al. [3] where the so called system software is modular and distributed over the NOC. Figure 1 b) depicts such an example where the system behavior is handled independently in the various interfaces. Such a modular approach supports scalability because hot spots of control messages are avoided. However, optimizations and system control cannot be performed globally but only in a rather local scope.

B. Service-Oriented Architectures

Service-Oriented Architectures (SOAs) provide a standardized view (i.e. an interface) for services based on software components as well as hardware devices. Thus, allowing the seamless interaction of different implementations and diverse platforms [10]. Therefore, SOAs represent the next step in the endeavor of component based development and reusability. All services in an SOA go through different stages in their life-cycle which is illustrated in figure 2. Entry point of the life-cycle is the *addressing*, so that each service can be uniquely identified in the network. Following, a service can either *announce* its presence so that it can be used by others or it *discovers* other services for its own purposes. A so called service *description* is exchanged in both cases describing the offered functionality and its usage. Finally, if a desired service was found it can be used, controlled, or manipulated (*usage*). Additionally, a service can also subscribe to another one to get informed about state changes (*eventing*) like for instance busy or idle state. The actual implementation of an SOA can rely on a central repository (called service registry) which keeps track of all services and its states. However, a peer-to-peer oriented implementation can also be chosen where all messaging has to be performed and handled by the affected components themselves.

Several implementations of SOAs have gained broad acceptance aiming at different goals. JINI – often referred to as Java Intelligent Network Infrastructure – was developed by Sun Microsystems for spontaneous networking of services and devices based on Java technology [13]. JINI requires a central service registry where the code (called proxy), needed to use the remote services, can be downloaded. Another example is Universal Plug and Play (UPnP) which is a simple and easy-to-use SOA for rather small networks of devices [14]. The communication is IP-based and the implementation is independent of the platform as well as the programming language. Furthermore, Web service protocols [15] cover most issues addressed by SOAs and can be composed to achieve domain specific goals. Finally, Devices Profile for Web Services (DPWS) adapts a subset of Web service protocols to enable resource constraint devices to participate in Web service environments [16].

III. DISCUSSION

Both introduced approaches of NOCs and SOAs have already proven their effectiveness in their domains, but the questions have not yet been answered whether and to what extent service-oriented approaches can be adopted for the

operation of large on-chip networks. Even though the denotations differ, the fundamental ideas of both concepts are very similar and motivate a further investigation:

- Encapsulation/Abstraction: Implementation details are hidden from the user and access is only granted by the use of a determined interface.
- Reusability/Composability: Applications are divided into small tasks/services with the intention of promoting reuse and assembling composite functions.
- Portability: The need for reuse to raise productivity motivates portability across different platforms and application domains which is enabled by encapsulation.

In the following, the consistency of both concepts and the counterparts of items are discussed. The equivalent of a web service in SOAs is the remote procedure call of NOCs. For example it is of no importance for a calling resource where the data-stream was decrypted as long as the result is the plain text of the original data. On the other hand, resources handling off-chip I/O are fixed to a certain position and represent a device service. Furthermore, the *addressing* as the first step in the life-cycle of a service has to be performed in NOCs as well. Currently, a multitude of addressing schemes exist ranging from absolute over relative addressing to dynamic logical addresses used in reconfigurable architectures on FPGAs. Against this background, it is impossible to discuss all implications of addressing here but it is obvious that the resources and services need to be uniquely identified in the network. The stages of *announcement*, *discovery*, and *description* are also necessary because software changes can enable new services at any time. However, reconfiguration or malfunction of resources during production or run-time can result in different service availabilities as well. Thus, self-healing is made possible and reliability is increased. Finally, as the *usage* needs no further explanation, the *eventing* in NOCs is mainly used to signalize service availability to other interested resources (called subscribers). For instance, a resource executing a single decoder in hardware cannot offer its service when it is already busy or unavailable. Though, the *eventing* could also be extended for the management and coordination of global issues like thermal distribution and load balancing. Such messages could simply be polled, sent in regular time intervals, or only when a determined limit is exceeded.

Even though the similarities are manifold, the main difference is the large acceptable protocol overhead of SOAs in contrast to distinct limitations of NOCs given by physical conditions (e.g. power consumption, limited memory). Hence, to directly adopt an existing implementation of an SOA is impractical and unnecessary because exchangeable communication stacks are not needed within a given on-chip network. However, a small subset of the overall functionality already allows to effectively using the mechanisms of SOAs to share services with other resources or to execute remote services because they cannot be done locally (e.g. I/O) or simply quicker elsewhere (e.g. hardware decoder).

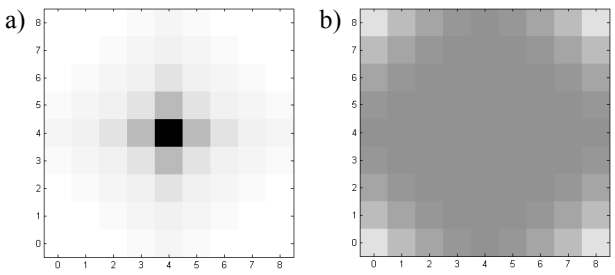

Figure 3. Level of activity of routers in a 9x9 mesh network with a) a centralized and b) distributed approach using adaptive routing

A. Centralized approach

A centralized approach assumes that all on-chip activities are initiated and controlled by a CCN which is probably a general purpose processor running an appropriate operating system. Such a system perspective allows global optimization and control of tasks, communication, and physical parameters like energy and temperature. Thereto, a central service registry is used to administrate all available services of the on-chip network and to monitor its states. That is, each resource has to *announce* its services to the registry during the setup time which leads to a massive appearance of congestions around the CCN. However, this behavior only occurs once during setup and can further be mitigated by mechanisms to randomly delay the initial *announcement* of various resources. In addition to the *description* of the service and how to use it, the *announcement* needs to include further data about its location and its power dissipation to facilitate decisions of global system control.

The simulation results of a 9x9 mesh network with an adaptive routing scheme and a centralized approach are shown in figure 3 a). 50.000 control messages were injected with a random distribution of the receiving destinations. The darker areas represent routers with high activity due to control messages sent by the CCN. It can clearly be observed that a local hot spot is created around the CCN representing a possible communication bottleneck. Hence, such a centralized approach becomes impractical with increasing network size because the number of messages to control the network as well as the distance to communicate them also increases. This will further occupy the network and result in additional latencies due to the large distance and message congestions around the CCN.

B. Distributed approach

Three different possibilities exist to implement a distributed approach of an on-chip SOA which conforms to the so called ad-hoc networks. Nonetheless, they have a lot of characteristics in common with the most important one that all activities are initiated by the resources themselves. The distributed nature of the system control promises to circumvent the hot spot around a single resource (like the CCN). Though, this comes at the price of forfeiting global awareness for the optimization of computation and communication. In addition, real-time requirements come into effect when for instance a resource wants to initiate a remote service that has slightly before been started by another resource. The critical time-frame in this example is the period between the start of the service and the reception of the *eventing* message that the service has become unavailable.

The first direct approach is to use local service registries in each resource. This means that broadcast messages have to be used for the *announcement* and

discovery to notify and update all resources and distributed repositories, respectively. An enormous emergence of messages is the result, particularly during setup. Once the system is settled, resources need to subscribe to remote services to become notified about state changes (*eventing*). Generally, a service will be used by several different resources so that events also have to be sent to all of them using multicast messages. Thus, the overall number of messages in the network is higher compared to a centralized approach and depends on the average number of subscribers to a service. Figure 3 b) depicts the results for a distributed approach with local registries where an average of five subscribers for a service is assumed. For the same conditions as described in the previous section, it can be seen that no hot spot arises and that the messages are spread over the whole network. Thereby, the maximum activity of the most occupied routers is reduced by more than factor two. However, the total amount of messages is increased five times due to the need of multicast messages for the subscribers which also raises the power consumption. Another drawback under power as well as area considerations is the large overhead of required memory because available services are stored redundantly in the local registries of the resources.

A second approach that avoids the need of local memory for the registry is to start a *discovery* every time a remote service is to be used. Thereto, broadcast messages have to be sent not just during setup but every time an activity is initiated. As a result of such an implementation, the total number of messages is drastically increased and so is the power dissipation. The raise is related to the number of resources in the network, i.e. about 81 times in the chosen scenario. Finally, the third attempt is a mixture of a distributed and a centralized approach. Thereto, control and initiation of activities remain in the local resources but a central registry is used jointly. Such an implementation does not require multi- or broadcast messages and avoids the need for local memory. The network behavior is very similar to the results shown in figure 3 a) but with twice the number of messages for *discovery*. This is because the beginning of an activity demands two messages (request and reply to the central registry) compared to a single command message from a CCN. Additionally, system optimization can only be performed on a local scale.

C. Outlook

The centralized approach seems more promising due to power savings and the possible global control. However, the hot spot of messages around the CCN severely endangers system functionality when further scaled. So, to use a mixture of both attempts might be a future solution.

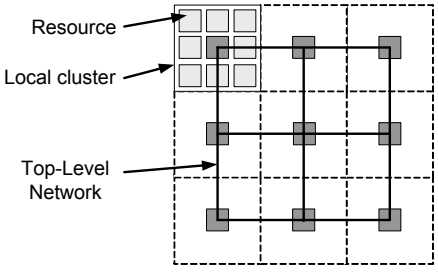

Figure 4. Possible scenario of a hierarchical network to join the concepts of centralized and distributed SOAs

This can be achieved in a hierarchical network as shown in figure 4. Each cluster of nine resources in the shown example employs a CCN with a central registry whereas the inter-cluster communication is based on the distributed attempts. For instance, a service *discovery* outside the local cluster can be done using request-reply methodology or the complete local registries are exchanged between the clusters. Future investigations will have to expose the optimal cluster size to achieve the best system performance. Thereto, all components contributing to power and delay of the system will have to be considered to achieve a final conclusion of the discussed approaches.

IV. CONCLUSION

The application of service-oriented architectures for the operation of large on-chip networks was discussed and several alternatives for the implementation were presented. The results for the centralized as well as for the distributed approach exhibit different properties and drawbacks so that the preference for one of them is application specific. Such a final evaluation depends on several parameters like network size or injection rate of control messages. As a rule of thumb, centralized systems should be favored for rather small networks and distributed systems for large ones. Lastly, a hierarchical system was discussed that could bring together the advantages of both approaches while masking their drawbacks.

REFERENCES

[1] Semiconductor Industry Association (SIA), "International Technology Roadmap for Semiconductors," 2003.

[2] W. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," Proc. of DAC, pp. 684-689, 2001.

[3] L. Benini and G. de Micheli, "Networks on chips: A new SoC paradigm," IEEE Computer, Vol. 35, pp. 70-78, 2002.

[4] V. Venkatraman et al., "NoCIC: A Spice-based Interconnect Planning Tool Emphasizing Aggressive On-Chip Interconnect Circuit Methods," Proc. of SLIP, pp. 69-75, 2004.

[5] N. Kavaldjiev, G. Smit, and P. Jansen, "Two Architectures for On-chip Virtual Channel Router," Proc. of PROGRESS, pp. 96-102, 2004.

[6] A. Mello, L. Tedesco, N. Calazans, and F. Moraes, "Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC," Proc. of SBCCI, pp. 178-183, 2005.

[7] R. Mullins, A. West, and S. Moore, "The design and implementation of a low-latency on-chip network," Proc. of ASP-DAC, pp. 164-169, 2006.

[8] S. Lee et al., "An 800MHz star-connected on-chip network for application to systems on a chip," Technical Digest of ISSCC, pp. 468-469, 2003.

[9] A. Jantsch, "NoCs: A new contract between hardware and software," Proc. of DSD, pp. 10-16, 2003.

[10] T. Erl, "Service-Oriented Architecture: Concepts, Technology, and Design," Prentice Hall PTR, 2005.

[11] V. Nollet, T. Marescaux, and D. Verkest, "Operating-System Controlled Network on Chip," Proc. of DAC, pp. 256-259, 2004.

[12] R. Hecht, S. Kubisch, H. Michelsen, E. Zeeb, D. Timmermann, "A Distributed Object System Approach for Dynamic Reconfiguration," Proc. of RAW, 2006

[13] Sun Microsystems, Inc., "JINI Architecture Specification Version 1.2," December 2001.

[14] UPnP Forum, "UPnP Device Architecture 1.0," June 2000.

[15] W3C, "Web Services Architecture," February 2004.

[16] H. Bohn, A. Bobek, and F. Golatowski, "SIRENA – Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework in different domains," Proc of ICN, p. 43, 2006