The 19th IEEE/IFIP International Symposium on Rapid System Prototyping

Integrating Abstract NoC Models within MPSoC Design

Edson I. Moreno[1], Katalin M. Popovici[2], Ney L. V. Calazans[1], Ahmed A. Jerraya[3]

[1] Faculty of Informatics, PUCRS, Porto Alegre, Brazil, {emoreno, calazans}@inf.pucrs.br
[2] TIMA Laboratory, Institute de Informatica, France, katalin.popovici@imag.fr
[3] CEA-LETI, MINATEC, France, ahmed.jerraya@cea.fr

Abstract

Current embedded applications are migrating from single processor-based systems to intensive data communication requiring multiprocessing. The performance demanded by these applications requires the use of heterogeneous multiprocessing architectures in a single chip (MPSoCs) endowed with complex communication infrastructures, such as Networks on Chip or NoCs. NoC parameter choices, such as network dimensioning, topology, routing algorithm, and buffer sizing then become essential aspects for optimizing the implementation of such complex systems. This paper presents NoC models that allow evaluating communication architectures through the variation of parameters during MPSoC design. Applicability of the concepts is demonstrated through two heterogeneous MPSoC case studies: an MJPEG decoder and an H.264 encoder.

1 Introduction

Multiprocessor SoCs (MPSoCs) are emerging as one of the technologies providing a way to face the growing design complexity of embedded systems, since they provide flexibility of programming allied to specific processor architectures adapted to selected problem classes. This leads to gains in compactness, low power consumption and performance [1][2]. MPSoCs integrate hardware (HW) such as processors, memories, interconnect and special purpose modules and software (SW) like operating systems and application code. MPSoC design is usually platform-based and dominated by SW design, to achieve cost and time efficiency [3]. The amount of functionality incorporated in an MPSoCs is continuously growing. Consequently, their complexity and size also increases. Therefore, on chip communication demands rise. Industry roadmaps and research literature point that communication will be the greater challenge in future MPSoC projects, representing up to 50% of the total energy consumption, thus becoming the system performance bottleneck [4]. The way an application is partitioned and the employed communication structures directly affect system energy and performance figures.

According to [5][6][7] traditional multipoint schemes, like single shared busses, will not be able to support the amount of communication required by future MPSoCs. Networks on chip (NoCs) emerge as an interesting approach because they help solving electrical problems in new deep-submicron technologies. NoCs can be more energy-efficient, more reliable, and more scalable than buses. Additionally, NoCs allow orthogonalizing computation and communication concerns, improving the capacity to design multi-billion transistor chips. However, adopting NoCs as communication infrastructures adds non-trivial design challenges to the MPSoC design flow, like architectural definition and communication protocol choice to achieve the best trade-off of cost, speed and power consumption [8][9][10].

This paper proposes a flow for inserting NoC design considerations during MPSoC design. This is achieved through communication architecture model parameterization and simulation with the target application. This flow makes use of three distinct abstraction levels to model application functionality together with the hardware architecture, and allows analyzing performance for varying NoC parameters. Among NoC parameters that can be varied stand the number of routers, the routing algorithm and the MPSoC IP cores mapping on the NoC. The main contribution of this paper is the definition of abstract NoC models that can be integrated into MPSoC design flows and the demonstration of the utility of such models through the use of two real world case studies, an MJPEG decoder and an H.264 encoder.

The paper is organized as follows. Section 2 presents related work in MPSoC communication modeling and evaluation. Section 3 depicts the proposed flow, and the proposed NoC models. Section 4 presents the conducted experiments, while Section 5 discusses results and their analysis. Finally, Section 6 gives concluding remarks.

978-0-7695-3180-9/08 $25.00 © 2008 IEEE
DOI 10.1109/RSP.2008.29

2 Related Work

Lahiri et al. [8] propose algorithms to map HW elements on predefined communication architectures according to a communication profile defined through performance analysis. This analysis enables to discover potential contentions on shared channels. However, the approach is assuming a memory-less communication infrastructure, which prevents the application of the method to most router-based NoCs.

Madsen et al. [11] present a method to optimize the usage of communication infrastructures modeled at system level with an RTOS, performing a static functionality analysis. Communication latency is modeled using best case figures, without considering congestion.

Coppola et al. [12] present OCCN, a framework for modeling and simulation of communication infrastructures which enables defining protocols, master and slave port behavior and statistical packages. However, OCCN does not support description of architectural details of the communication environment.

Bertozzi et al. [13] propose a three-step flow for the exploration and synthesis of NoCs: NoC topology mapping, selection and generation. The input is a core graph, describing the communication behavior of the system. The graph capture is not considered in their flow, being based on statistical analysis and simulation. This may lead to a wrong choice of the communication architecture in case a bad core graph is picked.

Xu et al. [14] present a methodology for evaluating different NoC architectures at low levels of abstraction, based on the application communication behavior. For accurate results, tools like OPNET, Design Compiler and SPICE are employed. Although the accuracy of the result is high, activities like communication analysis, performance analysis and interconnection design require time consuming modeling and simulation tasks.

Dumitrascu et al. [15] present a flow focused on obtaining communication architecture evaluation, based on the OCCN library [12] for all the communication models. Some statistical information is captured by the flow. Authors compare a distributed memory server, an AMBA bus and an Octagon NoC models. However, they do not support optimization of the communication architecture.

The work presented in this paper provides a method to evaluate different NoC configurations for a given application. The NoC models proposed in Section 3 are based on <blind> [16], a NoC infrastructure that enables automatic generation of RTL NoC descriptions for synthesis. In the proposed models, details of communication protocol are hidden from the system designer, assuming a read/write communication style. This enables gradual refinement of NoC from an abstract description down to architecture definition, e.g. topology and routing algorithm.

3 Communication Modeling

This Section starts by presenting the assumptions on the MPSoC design flow. Next, follows the discussion of the proposed generic communication model and the proposition of the abstract NoC models.

The design of MPSoCs, involving hardware and the software, requires that architecture and implementation definitions be done gradually. Figure 1 presents an MPSoC generic architecture. Figure 1(a) represents the global view of the architecture. The system is a composition of CPU subsystems (CPUSSs) which execute application threads, hardware subsystems (HWSS) which implement specific application behavior and the communication infrastructure. CPUSSs may include different components such as a processor CPU, a network interface, local memories and other peripherals (Figure 1 (b)).

In this work, application functionality is a composition of threads statically allocated to either a CPUSS or a HWSS, corresponding to the mapping process. The way threads are distributed affects communication characteristics. The communication characteristics are defined by the intra and inter subsystem data exchange. The intra subsystem data exchange occurs between threads allocated to the same subsystem, while inter subsystem communication refers to threads allocated to distinct CPUSS/HWSS.

Regarding the inter subsystem communication, the choice of efficient communication architectures is essential in the design flow. However, partitioning threads and evaluating alternative communication architectures are steps usually conducted at different moments of design. The proposed flow allows varying and evaluating communication architecture characteristics for an application mapped on a target architecture.

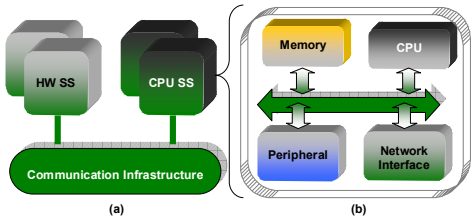

Figure 1: An MPSoC generic model. (a) Global view. (b) Architecture of CPUSS.

3.1 Communication Modeling

The flow proposed here to integrate NoC design during MPSoC design is depicted in Figure 2, and has three steps: (i) application modeling, (ii) communication architecture definition, and (iii) communication architecture refinement.

The first step consists in the functional modeling of the target application. Based on requirements specification, a high level application model is created using Simulink, from MathWorks. This model combines the ap-

plication description, partitioning and mapping. Application threads are virtually mapped to a given CPUSS/HWSS of the target MPSoC architecture. Communication between threads is described using primitives of the underlying framework. The model also contains the communication mapping information, i.e. the way threads communicate when mapped in the same or in distinct subsystems (e.g. shared memory or FIFOS). The simulation of the high level application model allows validation of the application functionality.

The second step defines an abstract model of the communication infrastructure. This model corresponds to the Virtual Architecture (VA), composed of abstract CPUSSs and HWSSs. The evaluation of communication infrastructure alternatives takes place through the adoption of dedicated links, busses or NoCs. In case a NoC architecture is chosen, the network dimension, i.e. its number of routers, has to be defined. The simulation of the VA model guarantees validation of the application mapping on the target architecture (e.g. it can guarantee deadlock-freedom) and generates quantitative data for communication evaluation. Example quantitative data are the amount of read and write requests, the number of packets sent/received, and the net amount of exchanged data. Based on these values, the designer may find communication bottlenecks, may change the network dimensioning, remap communication buffers to different storage resources (e.g. local memory, global memory) or adopt different communication strategies (e.g. dedicated hardware FIFOs). The net amount of exchanged data is an early indicator for an appropriate mapping of the MPSoC IP cores distribution over the NoC.

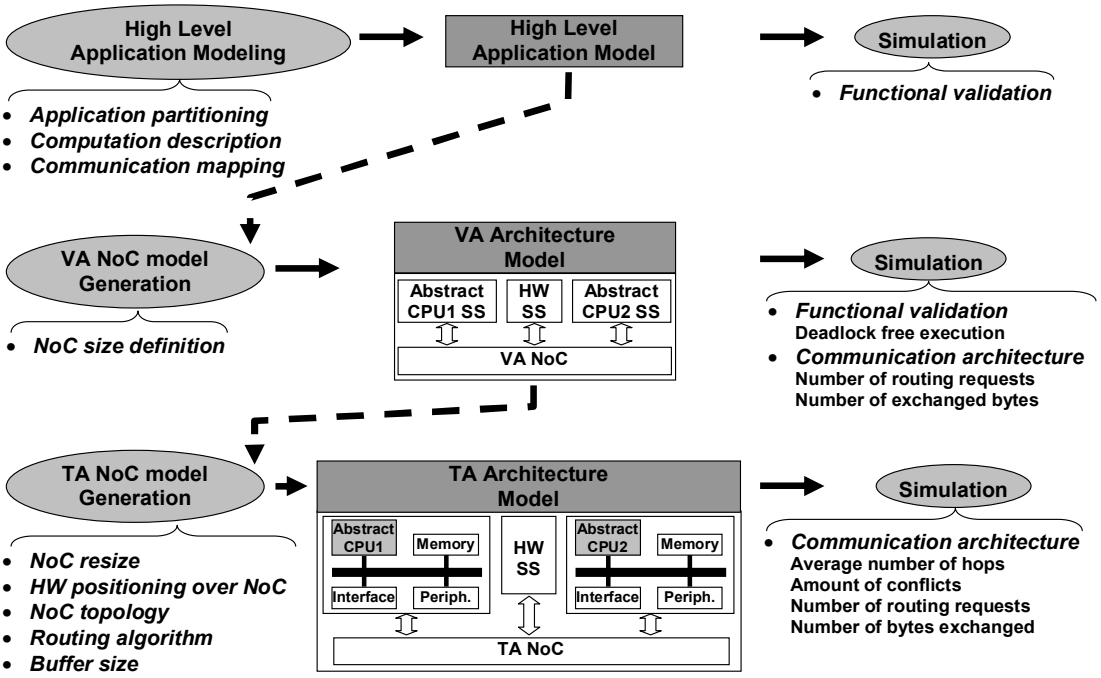

Figure 2: Flow adopted for integrating NoC modeling during MPSoC design.

The third step refines the communication infrastructure. This corresponds to the Transaction Accurate (TA) architecture model. The TA model allows detailing the local architecture of each CPUSS: abstract CPU, Local Memory, Network Interface and other peripherals. Additionally, it implements the communication protocol. Concerning the NoC, its architecture become explicit, by specifying the number of access points per subsystem, the mapping of IP Cores, the specific NoC topology, the routing algorithm and router buffer sizes. Each one of these characteristic influences the energy consumption, performance and silicon area of the final system. The simulation of the TA model allows communication architecture performance evaluation, by measuring values such as the average number of hops taken by each packet, the degree of NoC congestion, the overall number of routing requests and the total number of bytes exchanged during simulation with real data.

3.2 NoC Models

The following paragraphs describe the NoC models adopted in the VA and TA architecture. Both were implemented in SystemC.

The VA NoC Model

The VA NoC model is illustrated in Figure 3. This model allows fully parallel communication among different subsystems composing an MPSoC. The model represents an abstract NoC where information like topology, routing algorithm, arbitration or buffer size information are abstracted. Communication architecture is modeled like a crossbar, where any set of communication events may take place simultaneously.

The VA NoC model is composed of three basic elements, which are the network interface (NI), the mapping table (MT) and the router. The NI is responsible for providing send/receive operations for communicating threads, encapsulating these requests in packets, capturing and interpreting packets arriving from the NoC, and delivering them to subsystems. The MT is responsible for storing and informing the correspondence between IP Core range addresses and NoC physical addresses. For example, IP Core addresses between 0x00400000 and 0x007FFFFF may correspond to a single NoC physical address, say 0x0. The router is in charge of sending and receiving packets from source to destination.

The VA description of the MPSoC is automatically generated from the high level application description. From the simulation of the VA description together with the VA NoC, further information can be captured. Examples are the amount of data exchanged between the different subsystems, the storage elements worst case size requirement for the communication buffer, the number of operations (send/receive) originated from each access point of the NoC and the amount of read/write operations performed at storage elements. Also, it is possible to obtain first estimates of NoC area, based on the number of routers alone.

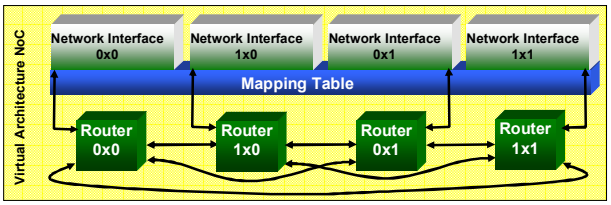

Figure 3: VA NoC model components and example.

The TA NoC Model

The TA NoC model is illustrated in Figure 4. This model adds still more architectural details such as topology, routing algorithm and router buffers size. The TA model of the MPSoC is automatically generated.

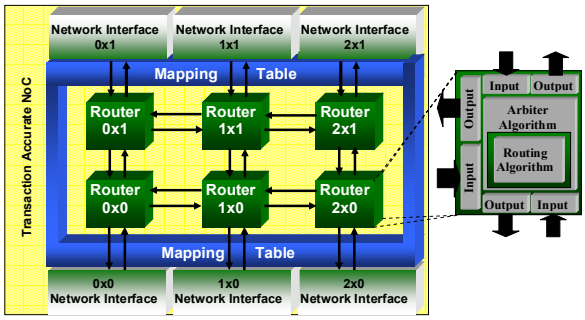

Figure 4: TA NoC Model example: mesh topology, pure XY routing algorithm, round robin arbiter algorithm.

The TA NoC model is composed of the same basic elements as the VA NoCs (NI, MT and routers) with a more detailed implementation. Topology (e.g. mesh, torus), routing algorithm (e.g. pure XY, west first), arbiter algorithm (e.g. round robin or fixed priority) and buffer size (e.g. number of flits) can be varied. The packet structure in this model is composed of destination address, size and body fields, similar to that assumed in the synthesizable NoC description.

The TA NoC allows extracting information from the system communication architecture, including: (i) number of routing requests; (ii) number of packets inserted into the NoC; (iii) amount of exchanged bytes; (iv) average number of bytes per packet; (v) the number of transmitted packets, (vi) number of failing routing requests, due to NoC congestion.

4 Experiments

This Section presents the results obtained by applying the proposed flow in the case of the Motion JPEG decoder and H.264 encoder applications mapped onto a multimedia platform. For the MJPEG decoder, 10 frames were used as input bitstream encoded using QVGA YUV 444 format. For the H.264 encoder, 5 frames encoded using QCIF YUV 420 input video format were used. The target hardware architecture is a simplified version of Diopsis [17], which includes an ARM and a DSP subsystem (two CPUSSs). The ARM subsystem includes the processor core and local memories, while the DSP subsystem includes the DSP core, data and program memories, DMA, interrupt controller and synchronization components. The HWSS nodes consist of an external distributed memory subsystem (DXM) and the peripheral on tile (POT) subsystem. The POT includes system peripherals of the ARM processor (timer, interrupt controller), and also I/O components like serial peripheral interface. The next Section presents the NoC modeling flow using 2D mesh and 2D torus topologies.

4.1 Modeling MJPEG decoder and H.264 encoder

The MJPEG decoder and the H.264 encoder were first modeled at the functional level and mapped on the Diopsis architecture. The MJPEG application was partitioned into 4 threads. Two threads were mapped to the ARM subsystem, one thread to the DSPSS and one to the POT. The MJPEG decoder model contains three inter subsystem communication units.

The H.264 encoder was partitioned into two threads, T1 and T2 which were mapped to DSP and ARM, respectively (Figure 5). The communication between the ARM and DSP subsystems require 26 communication units. The communication buffers were mapped on global memory for both applications.

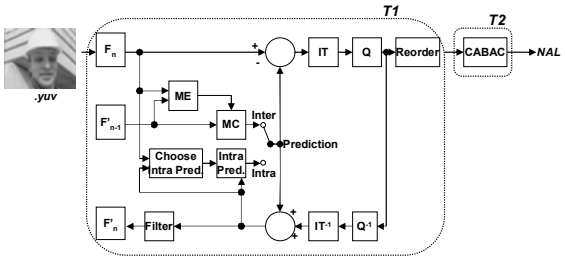

Figure 5: H.264 application modeling and partitioning.

4.2 Diopsis and the VA NoC

Figure 6 represents the Diopsis architecture with VA NoC, running the MJPEG application. In this model, 4 access points are necessary to connect the DXM, POTSS, ARMSS and DSPSS.

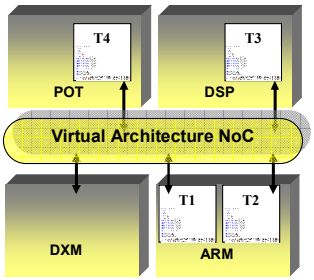

Figure 6: VA description of Diopsis architecture interconnected by VA NoC.

TABLE 1 and TABLE 2 show simulation results of the VA NoC model for the MJPEG decoder and H.264 encoder. The first two columns show IP Cores and NoC addresses relation. The third column displays the total number of read/write requests. The designer may define a better mapping of hardware or the size of packets with such values. The fourth and the fifth columns represent the amount read and write operations from/to the global memory. The sixth and the seventh columns allow evaluating the amount of communication injected into the NoC.

TABLE 1: Results captured from VA NoC during the MJPEG simulation.

	NoC address	Read/Write request	Read operation	Write operation	Packets sent	MBytes sent
DXM	0x0	0	4,798,220	4,798,156	223,173	21.28
POTSS	0x1	37,195	0	0	74,390	0.99
ARMSS	1x0	74,391	0	0	185,979	11.29
DSPSS	1x1	111,586	0	0	260,367	12.42
				Total	743,909	45.98

For the MJPEG decoder simulation, the DSP and the DXM were those who injected most packets into the NoC. Even if the DXM injected fewer packets than the DSP, it can be seen that it inserted much more bytes than any other subsystem, due to block transfers to/from memory, where a control packet can more than a single word.

For the H.264 encoder, the DXM was the element that inserted the largest amount of data into the NoC. In both cases, DXM packets originate from read requests and confirmation packets.

TABLE 2 - Results captured from VA NoC after encoding 5 frames with H.264.

	NoC address	Read/Write request	Read operation	Write operation	Packets sent	GBytes sent
DXM	0x0	0	309,483,404	309,483,404	19,175,640	2.55
POTSS	0x1	1,065,313	0	0	0	0,07
ARMSS	1x0	3,195,940	0	0	6,391,880	0.84
DSPSS	1x1	3,195,940	0	0	6,391,880	0.47
				Total	31,959,400	3.86

4.3 Diopsis and TA NoCs

Figure 7 gives the block diagram for the Diopsis platform using the TA NoC as communication infrastructure. Two types of TA NoC topologies were modeled: 2D mesh and 2D torus.

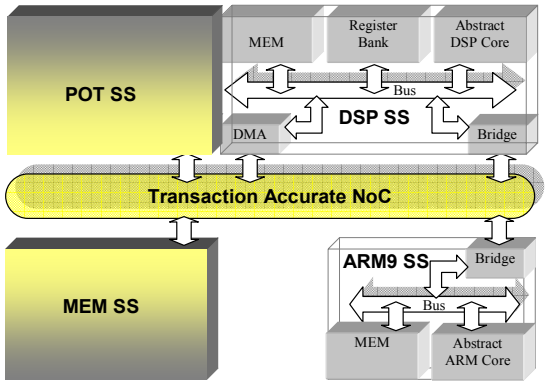

Figure 7: Block diagram of architecture for the Diopsis platform with TA NoC.

First, the TA NoC employs a 2D mesh topology, a pure XY routing algorithm and a round robin arbiter algorithm at each router and wormhole as packet switching strategy (TA Mesh). Then, the TA NoC used 2D torus topology and deadlock free of a non-minimal west-first

routing algorithm proposed by Glass and Ni [18] (TA Torus).

In the TA model, five access points to the NoC were necessary: four access points for the different subsystems, as previously presented in the VA model, and one additional for the DMA component, which becomes explicit in the TA model and has direct link to the interconnect component. With regular topologies such as the 2D torus and mesh, the smallest usable NoC has 6 routers (3x2).

TABLE 3 shows results captured from TA Mesh model simulation for the MJPEG decoder application. The first and second columns represent the correspondence between the different subsystems and the NoC access points. A routing request is performed at least once per packet per router it will cross. Depending on the application, the NoC structure, routing algorithm, and on NoC congestion state, the routing request may occur as many times as needed inside a router. For the MJPEG simulation, 42.623.519 routing requests were issued. The third column of TABLE 3 presents the percentage of routing requests at each router, while the remaining columns detail this information for each router port.

TABLE 3: Percentage of routing requests during simulation with the TA 2D Mesh NoC.

	NoC address	TOTAL	LOCAL	NORTH	SOUTH	EAST	WEST
MEM SS	0x0	24,1%	11,3%	5,7%	0 %	7,1%	0 %
POT SS	0x1	11,4%	2,9%	0 %	1,3%	7,2%	0 %
	1x0	18,3%	0 %	0 %	0 %	7,0%	11,3%
DSP (DMA)	1x1	10,1%	0 %	0 %	0 %	7,2%	2,9%
ARM SS	2x0	20,0%	7,2%	1,5%	0 %	0 %	11,3%
DSP (Bridge)	2x1	16,1%	7,4%	0 %	5,8%	0 %	2,9%

Figure 8 shows the amount of data that traverses each router in the TA Mesh for the MJPEG encoder application. The local port of each router inserts packets into the NoC, while the remaining ports transfer them inside the NoC. The value assigned to the local port of router 0x0 (MEM SS) corresponds to response packets due to read requests or confirmation packets due to write requests. Depending on the way communication is mapped and performed, these values can change. For example, block transfer operations (amount of operation that will be transferred in one packet) allows optimizing the amount of data exchanged inside the NoC by minimizing the amount of control data.

TABLE 4 shows results captured from TA Mesh model of the H.264 encoder simulation. The third column of the Table represents the exchanged amount of data and control information (e.g. operation request, confirmation response, etc). Again, the remaining columns of the Table detail the amount of data transmitted per router port.

For the TA Torus model, 30.440.287 and 3.693.770.895 routing requests were issued during the simulation of the MJPEG decoder and H.264 encoder respectively, representing in both cases a 29% reduction of routing request activity when compared to the TA Mesh. This was possible because the 2D torus topology has longest minimum paths that are only half of those in 2D meshes in hops. Also, tori networks have better path diversity than meshes. This, if exploitable by the routing algorithm, leads to less network congestion, thus potentially reducing routing requests.

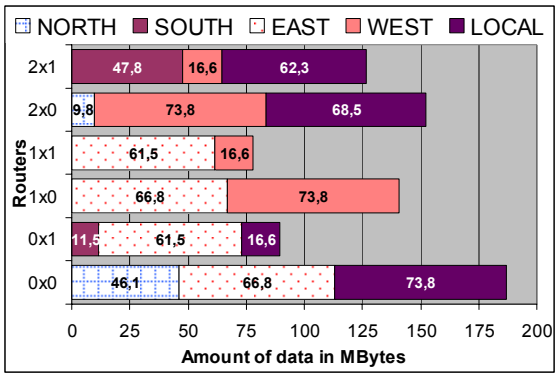

Figure 8: TA Mesh data quantification on a per router and per port basis, for MJPEG simulation. For TA Torus, Routers 1x0 and 1x1 display no traffic activity, while other ports have values identical to TA-Mesh.

TABLE 4: Amount of data transmitted in TA 2D Mesh NoC for H.264 encoder.

	NoC address	LOCAL	NORTH	SOUTH	EAST	WEST
MEM SS	0x0	9,21 GB	6,15 GB	0,00 GB	8,40 GB	0,00 GB
POT SS	0x1	1,85 GB	0,00 GB	1,85 GB	6,15 GB	0,00 GB
	1x0	0,00 GB	0,00 GB	0,00 GB	8,40 GB	9,21 GB
DSP (DMA)	1x1	0,00 GB	0,00 GB	0,00 GB	6,15 GB	1,85 GB
ARM SS	2x0	6,07 GB	1,85 GB	0,00 GB	0,00 GB	9,21 GB
DSP (Bridge)	2x1	6,15 GB	0,00 GB	4,98 GB	0,00 GB	1,85 GB
Total data inserted		23,28 GB				

5 Results and Analysis

Through the VA NoC model it is possible to estimate the NoC size in number of routers, the amount of communication present in the system and the functionality of the application over the virtual architecture. In the present work, a VA NoC composed of 4 routers and the communication between CPUSSs assuming a global memory was assumed. A deadlock free execution of the application was detected for both MJPEG decoder and H.264 encoder case studies.

The evaluation of different NoC architecture is possible at the TA model. For the TA NoC models, resize was performed and topology, and consequently routing algorithm, was varied. In both NoC topologies and applications, no congestion was detected. Consequently, buffer size remained constant. Based on simulation results, the torus NoC allows a better performance while consuming less energy, due to the decrease on the path covered by packets. On the other hand, mesh NoCs allow reduction of communication area overhead. In this case, area over-

head reduction is due mostly to the elimination of buffers in routers located at the border of the NoC.

6 Conclusions and Future Work

The flow presented here allows integrating the design of NoCs during MPSoC design for some target application running on top of a system platform. The description of the approach showed hints on what kind of information can be obtained from the proposed abstract NoC models. This information is currently being used to develop a design exploration flow for MPSoCs using NoCs as intra-chip communication architecture.

The flow proposed here was employed to design an MJPEG decoder and H.264 encoder running on the Diopsis platform, experimenting with two different NoC topologies, 2D mesh and 2D torus.

Future work includes extending the supported NoC templates at both the VA and TA abstraction levels, adding diversity to the choices of NoC templates. Another ongoing work is linking the TA level to the RTL synthesizable abstraction level and improving parameterization at the TA level, to include e.g. the possibility of buffer dimensioning at the network interface. This is an important step to decouple application transmission rates from NoC transmission rates, which increases independence of application concerns from communication architectures.

REFERENCES

[1] Jerraya, A; Tenhunen, H. and Wolf, W. Multiprocessor Systems-on-Chips. IEEE Computer, 38(7), 2005.

[2] Tan, Z. et al. Design and implementation of the Software System on MPSoC: An HDTV Decoder case study. IEEE Consumer Electronics, 52(4), 2006.

[3] Cesario, W. et al. Component-based design approach for multicore SoCs. DAC, 2002.

[4] Kadayif, I.; Kandemir, M. and Chen, G. Influence of Communication Optimizations on On-Chip Multi-Processor Energy. SoC Conference, 2003.

[5] Benini, L. and De Micheli, G. Networks on chips: a new SoC paradigm. IEEE Computer, 35(1), 2002.

[6] Sgroi, M. et al. Addressing the System-on-a-Chip Interconnect Woes Through Communication-Based Design. DAC, 2001.

[7] Lee, H. G. et al. Design Space Exploration and Prototyping for On-chip Multimedia Applications. DAC, 2006.

[8] Lahiri, K.; Raghunathan, A. and Dey, S. Design space exploration for optimizing on-chip communication architectures. IEEE Trans. on CAD, 23(6), 2004.

[9] Narasimhan, A.; Kumaravelu, O. and Sridhar, R. An investigation of the impact of network parameters on performance of network-on-chips. Midwest Symposium on Circuits and Systems, 2005.

[10] Monchiero, M. et al. Exploration of Distributed Shared Memory Architectures for NoC-based Multiprocessors. Journal of System Architectures, Elsevier, 2007.

[11] Madsen, J. et al. Network on Chip Modeling for System Level Multiprocessor Simulation. RTSS, 2003.

[12] Coppola, M. et al. OCCN: A Network on chip Modeling and Simulation Framework. DATE, 2004.

[13] Bertozzi, D. et al. NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems on Chip. IEEE Trans. on Parallel and Distributed Systems, 16(2), 2005.

[14] Xu, J. et al. A methodology for design, modeling, and analysis of network on chip. ISCAS, 2005.

[15] Dumitrascu, F. et al. Flexible MPSoC Platform with Fast Interconnect Exploration for Optimal System Performance for a Specific Application. DATE, 2006.

[16] Omitted for the purpose of blind review.

[17] Paolucci, P. et al. SHAPES: a tiled scalable software hardware architecture platform for embedded systems. CODES+ISSS, 2006.

[18] Glass, C. and Ni, L. The Turn Model for Adaptive Routing. Journal of the ACM, 41(5), 1994.