Evaluation of Interconnect Fabrics for an Embedded MPSoC in 28 nm FD-SOI

Gregor Sievers*, Johannes Ax*, Nils Kucza*, Martin Flaßkamp*, Thorsten Jungeblut*,
Wayne Kelly†, Mario Porrmann*, Ulrich Rückert*
*Cognitronics and Sensor Systems Group, CITEC, Bielefeld University, Bielefeld, Germany
†Science and Engineering Faculty, Queensland University of Technology, Brisbane, Australia
Email: gsievers@cit-ec.uni-bielefeld.de w.kelly@qut.edu.au

***Abstract*—Embedded many-core architectures contain dozens to hundreds of CPU cores that are connected via a highly scalable NoC interconnect. Our Multiprocessor-System-on-Chip CoreVA-MPSoC combines the advantages of tightly coupled bus-based communication with the scalability of NoC approaches by adding a CPU cluster as an additional level of hierarchy. In this work, we analyze different cluster interconnect implementations with 8 to 32 CPUs and compare them in terms of resource requirements and performance to hierarchical NoCs approaches. Using 28 nm FD-SOI technology the area requirement for 32 CPUs and AXI crossbar is 5.59 mm^2 including 23.61% for the interconnect at a clock frequency of 830 MHz. In comparison, a hierarchical MPSoC with 4 CPU cluster and 8 CPUs in each cluster requires only 4.83 mm^2 including 11.61% for the interconnect. To evaluate the performance, we use a compiler for streaming applications to map programs to the different MPSoC configurations. We use this approach for a design-space exploration to find the most efficient architecture and partitioning for an application.**

I. Introduction

The decreasing feature size of microelectronic circuits allows for the integration of more and more processing cores on a single chip. Therefore, the high number of processing cores poses high demands on the underlying communication infrastructure. For providing efficient communication between the CPUs and to increase scalability, a dedicated NoC infrastructure is inevitable. Nevertheless, the area and power overhead of a NoC is high compared to the small processing cores. The CoreVA-MPSoC used in this work is a highly scalable multiprocessor system based on a hierarchical communication infrastructure (cf. Fig. 1) and a configurable VLIW processor. The CPU cores in a cluster are tightly coupled via a bus interconnect that can be compliant to either the ARM AXI or the OpenCores Wishbone standard. Both bus standards support a shared bus or a crossbar topology. There is no common shared memory, but the CPUs can access each other's local memory in a Non-Uniform Memory Access (NUMA) fashion. The global interconnect of the CoreVA-MPSoC is a configurable Network-on-Chip, which allows for the implementation of different network topologies. A typical configuration, used throughout this paper, consists of a 2D-mesh with a processor cluster consisting of multiple CPU cores connected to each network node of the NoC. The CoreVA-MPSoC targets streaming applications specialized for embedded mobile devices, which require a high resource efficiency.

This paper shows an exploration of different bus configurations to aim for the most resource-efficient intra-cluster communication in the CoreVA-MPSoC. The main contributions of this work are the analysis of the scalability of tightly coupled processor clusters in hierarchical MPSoCs. We compare different interconnect bus standards and topologies (shared bus, crossbar, and NoC) in a 28 nm FD-SOI technology for 8 to 32 CPU cores. We determine the optimal number of register stages to obtain a target frequency of 830 MHz which is the maximum frequency of our CPU cores. To compare execution performance, a self-built compiler is used to map different streaming-based benchmarks to the analyzed 16 CPU MPSoC configurations.

II. Related Work

Energy efficient, hierarchical MPSoCs have been widely adopted in research and industry. However, there is not much research into the partitioning of cluster interconnects in combination with a NoC-based MPSoC. The STM STHORM [1] connects up to 16 CPUs and multi-banked L1 data memory via a logarithmic interconnect. Four of these CPU clusters are connected via a NoC. STHORM can be programmed via OpenCL or a proprietary Native Programming Model. The Kalray MPPA-256 [2] is a commercial, hierarchical 288-core MPSoC targeting embedded applications. Each CPU cluster contains 16 processing CPUs, a system CPU and shared memory. Nevertheless, the impact of the number of cores per cluster and different cluster interconnects are not analyzed in these papers. Adapteva's Epiphany E64G401 [3] is a 64 CPU multiprocessor with 2 MB memory. The maximum operation frequency is 800 MHz. A 3-layer 2D-mesh NoC is used as interconnect fabric. The Epiphany does not introduce a cluster-level hierarchy but solely relies on NoC communication. Our hierarchical approach reduces the NoC overhead by allowing tightly-coupled communication within a CPU cluster. Angiolini *et al.* [4] compare an AMBA AHB shared bus, a partial 5-layer

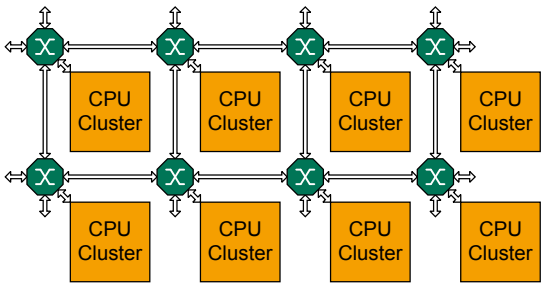

Fig. 1: Hierarchical CoreVA-MPSoC with 4x2 mesh NoC.

AMBA AHB crossbar, and an xpipes NoC as interconnect fabrics for a MPSoC connecting 30 IP cores. The shared bus does not provide adequate bandwidth for the considered applications. The 5-layer crossbar outperforms the NoC in terms of area and power in a 130 nm process. Angiolini *et al.* consider only NoC nodes (cluster) with 1 master (CPU or traffic generator) and slave (memory) per NoC node. We compare different partitionings with different numbers of CPUs per NoC node. Kumar *et al.* [5] perform a design-space exploration for a multiprocessor with 4, 8, and 16 Power4-like CPUs. They consider a shared bus, a crossbar, and a hierarchical interconnect consisting of two shared buses. It is shown that the architecture of the interconnect highly influences overall system performance. For example, the area savings due to reduced shared-bus bandwidth can be used for larger caches which results in an improved system performance. Kumar *et al.* target server applications and do not consider a NoC.

III. The CoreVA-MPSoC Architecture

The CPU used in our MPSoC is named CoreVA [6] and features a configurable 32 bit VLIW architecture. It has separate instruction and data memories and six pipeline stages. The number of VLIW issue slots, arithmetic-logic-units (ALUs), multiply-accumulate (MAC), and load-store-units (LD/ST) can be adjusted at design time. The CPU integrates a bus slave interface to enable access to the memories from the bus and for initialization/control. To avoid CPU stalls due to bus congestion, a FIFO is used to decouple CPU bus writes from the bus master interface. Both master and slave interfaces are generic to enable the evaluation of different bus standards and topologies (see below). Our C-compiler tool-chain for a single CPU is based on LLVM and supports VLIW and SIMD vectorization.

A. Cluster level

In a CPU cluster several CoreVA CPUs are tightly coupled using an interconnect fabric. The cluster implements a NUMA architecture, where each CPU can access the data memories of all other CPUs within a cluster (cf. Fig. 2). In this work, two different interconnect standards are considered. Wishbone (WB) is an open-source interconnect-standard maintained by the community project OpenCores [7] and is used in a broad range of academic and open-source projects. WB represents a classic

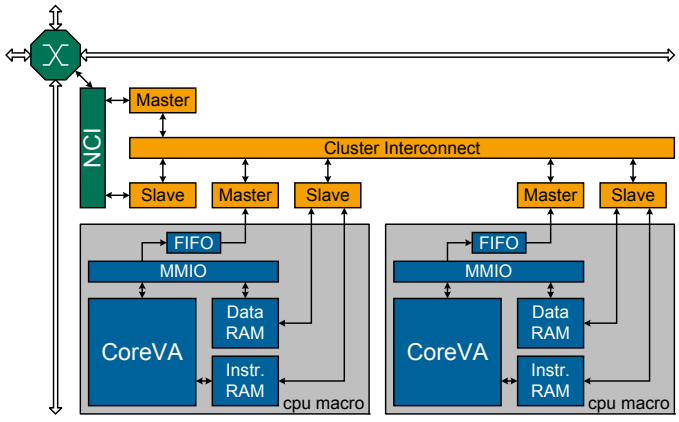

Fig. 2: CPU cluster with NoC-Cluster-Interface (NCI).

bus standard that can be found in (multi-) processor systems for the last two decades. We implemented the pipelined variant of the standard with asynchronous cycle termination. AMBA AXI4 is the latest interconnect specified by ARM Inc. [8]. AXI is targeting high performance embedded multiprocessor systems and is widely used in both industry and academia. AXI defines separate channels for address- and data transfers. In addition, read and write channels are separated and allow for parallel read and write requests even for a shared bus. This results in five channels in total (read and write address, read and write data, and write response). Both interconnect implementations are not registered by default. Register stages can be added to both master and slave ports of the interconnect to increase the maximum clock frequency and to simplify place and route (P&R) timing closure. A WB master can issue a write operation every cycle whereas AXI allows a write operation every second cycle only. However, a write operation of another AXI master can be interleaved. Outstanding read requests are not supported as our VLIW architecture does not implement out-of-order execution. The minimum read latency is 4 cycles for both AXI and WB (without register stages). The WB shared bus requires 1 arbiter in total, the WB crossbar 1 arbiter per slave, and the AXI shared bus 5 arbiters (1 per channel). The AXI crossbar interconnect requires 2 arbiters per slave (read and write address channel). The data channels do not require extra arbitration, because our interconnect does not support outstanding transactions. We use a round robin arbitration scheme. The data bus width of both interconnects is configurable at design time and is fixed to 32 bit in this work.

B. Network on Chip

For realizing MPSoCs with dozens or hundreds of CPU cores, a second interconnection hierarchy level, a Network on Chip (NoC), is introduced to the CoreVA-MPSoC. The NoC considered in this work features packet switching and wormhole routing. Each packet is segmented into small flits, each containing a 23 bit header for control information (for a 4x4 NoC) and 64 bit payload data. The maximum payload size of a packet is configurable at design time and set to 4 kB in this work. The NoC is built up of switch boxes, each having a configurable number of ports. This flexibility enables the implementation of most common network topologies. In this work, a 2D-mesh topology is used (cf. Fig. 1). Each switch box has a latency of two clock cycles. To increase the average throughput, virtual channels can be introduced to the NoC, which implies dedicated input buffers in each output port. One port of each switch box is connected to a cluster via a network cluster interface (NCI, cf. Fig. 2). The NCI acts as a DMA controller within the CPU cluster. It stores incoming NoC flits directly in the data memory of the target CPU. Vice versa, outgoing packets stored in the data memory are separated into flits and transferred to the switch box. For very large scaled CoreVA-MPSoCs the NoC can be extended by a Globally-Asynchronous Locally-Synchronous (GALS)-based approach by using mesochronous links [9]. Mesochronous links between the switch boxes divide the system into frequency domains. Each domain consists of a cluster and a single switch box (cf. Fig. 2). In this work, only small NoCs are considered, so virtual channels and GALS are not required and disabled. An accurate simulator of the MPSoC has been developed to aid the software development and debugging.

IV. Implementation Results

In this section we compare synthesis results for MPSoC configurations comprising 8, 16, and 32 CoreVA CPUs (cf. Fig. 3). We used a highly automated standard-cell design-flow based on Cadence Encounter Digital Implementation System. We vary the interconnect type (WB or AXI) and topology (shared bus or full crossbar). In addition, we consider NoC configurations with 1, 4, and 8 CPUs per cluster. Basic block is a CoreVA CPU hard macro with 2 VLIW slots, 16 kB data- and 16 kB instruction-memory. The maximum frequency of this hard macro is 830 MHz in a 28 nm FD-SOI standard cell technology[1]. Area requirements are 0.133 mm^2 and the estimated power consumption is 16.38 mW. In this work, the whole MPSoC has a single clock domain and all syntheses are performed with a target frequency of 830 MHz.

First, we determined the required number of master and slave register stages for the interconnect. For each configuration, we started without any register stage and increased the number of registers until the target frequency was achieved. Registers can be placed in between masters and the interconnect and/or in between the interconnect and the slaves. All configurations with 8 CPUs and the 16 CPU WB shared bus require 1 master register stage to meet the timing. The 32 CPU AXI shared bus configuration requires 2 master- and 1 slave register stages. All other considered configuration require 1 master- and 1 slave register stage.

The 8 CPU cluster with WB shared bus has a total area of 1.09 mm^2, the interconnect requires only 2.33% (0.025 mm^2) of this area. WB crossbar, AXI shared bus, and AXI crossbar consume 0.059 mm^2, 0.053 mm^2, and 0.094 mm^2 respectively. A 4x2 mesh NoC with 1 CPU per cluster (4x2x1) has an area of 1.53 mm^2 with 0.464 mm^2 (30.28%) for the NoC. This shows that a pure NoC has a large area overhead compared to bus-based interconnects. The NoC with two 4-CPU clusters (2x1x4) requires 1.22 mm^2 with 0.070 mm^2 for the two AXI crossbar clusters and 0.090 mm^2 for the NoC. Area for the 16 CPU clusters varies from 2.18 mm^2 (WB bus) to 2.46 mm^2 (AXI crossbar). The WB crossbar is 3.9 times larger than the WB shared bus whereas the AXI crossbar requires 2.3 times more area than the AXI shared bus. Both considered NoC configurations have approximately the same size compared to the full AXI crossbar (2x2x4: 2.45 mm^2, 2x1x8: 2.41 mm^2). Considering 32 CPU cores, the shared bus implementations of WB and AXI scale quite well and require only 2.32% and 8.02% of the overall area. The crossbars interconnects consume 14.81% (WB) and 23.61% (AXI) of the overall area. A 4x2x4 MPSoC consumes 13.05% of the area whereas the 2x2x8 MPSoC requires 11,61% of the overall area for the interconnect.

The synthesis tool provides power estimation based on default input switching activities. For our CoreVA CPU hard macro, switching activities of 10% result in a good power estimation compared to our simulation-based annotations. In the following we present first power estimations for the 16 CPU cluster. The WB shared bus cluster consumes 275 mW in total including 5% for the interconnect. The WB crossbar dissipates 308 mW (15% for the interconnect), the AXI shared bus 314 mW (16%), and the AXI crossbar 346 mW (24%). The 2x2x4 NoC consumes 349 mW and the 2x1x8 NoC 336 mW. Future work includes more detailed power analysis based on gatelevel simulations.

[1]STMicroelectronics, 10 metal layer, Worst Case Corner: 1.0 V, 125°C

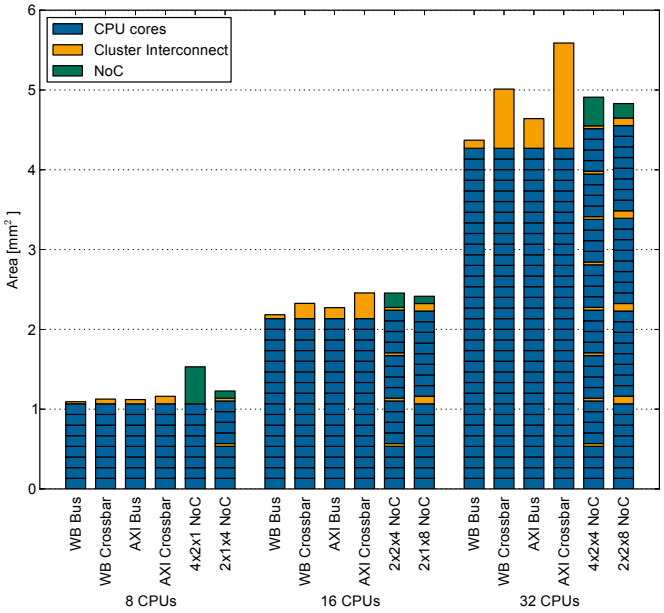

Fig. 3: Area requirements of different MPSoC configurations, 2-issue VLIW CoreVA CPU, and 32 kB memory per CPU.

Fig. 4 shows layouts of a 16 CPU Cluster with a WB crossbar interconnect and a 2x2x4 NoC. Both layouts contain 16 2-issue VLIW cores and 512 kB memory in total. The clock frequency is slightly reduced compared to logic synthesis estimation (825 MHz cluster, 812 MHz NoC). The area of the 16 CPU cluster is 21.0% (0.49 mm^2) higher compared to the synthesis estimation. The 2x2x4 NoC-based MPSoC requires 3.02 mm^2 which is a 22.7% increase compared to synthesis results. The deviation in the area requirements can mainly be explained by an exceptional routing overhead for the interconnect (esp. in the center of the layout) that was estimated too optimistically by the synthesis tool. This overhead could be decreased by using a CPU hard macro that does not use all routing layers.

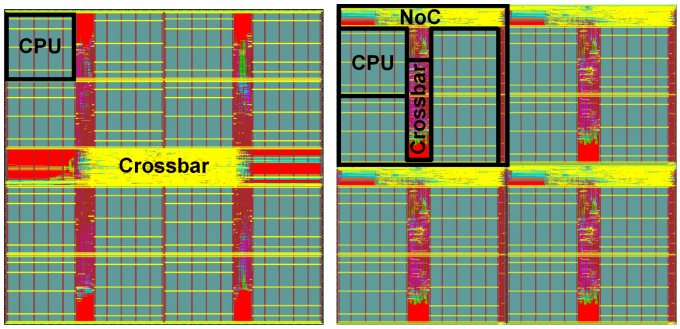

Fig. 4: Layout of a 16 CPU cluster with a WB crossbar (left, 2.82 mm^2) and a 2x2x4 NoC (right, 3.02 mm^2).

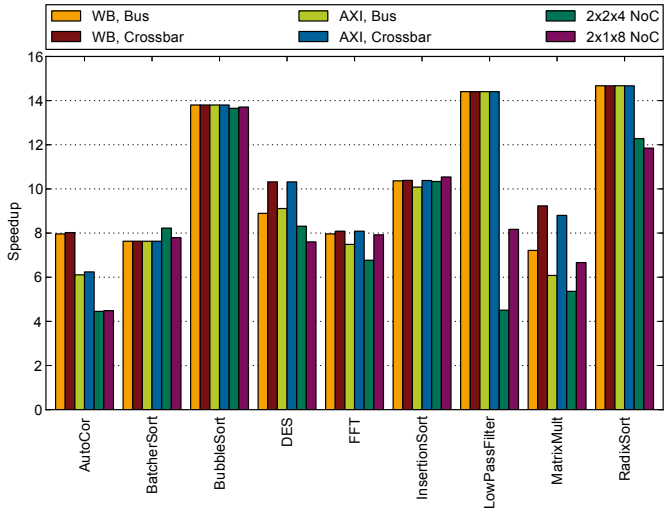

Fig. 5: Speedup of MPSoC configurations with 16 CPU cores in relation to a single CPU.

V. Benchmark Results

Programming a large number of CPU cores is a complex task requiring compiler support. In [10] we presented a compiler for the StreamIt language [11] targeting the CoreVA-MPSoC architecture. A StreamIt program consists of a collection of filters, in which a filter takes as input a data stream, processes the data and produces an output data stream. Each filter is entirely independent and communicates only via input and output channels. Filters can be executed in parallel and allow the compiler to map them to different CPUs using a simulated annealing optimization algorithm. To manage the communication channels between two filters on different CPUs, our StreamIt compiler uses a unified communication library. A communication channel handles buffer management and features a mutex-based synchronization scheme. Each channel consists of two or more buffers to hide latencies by filling one buffer while another is being read (multi buffering). The data buffers of a cluster-internal channel are allocated in the memory of the receiving processor to avoid bus read latencies. For a NoC channel, data buffers are allocated at the sending and at the receiving cluster. Because of this, the memory footprint of a NoC channel is doubled compared to a cluster channel.

Fig. 5 shows the throughput speedup for some StreamIt benchmarks [11] using 16 CPUs and different MPSoC configurations compared to a single CPU. The applications BubbleSort, RadixSort, and LowPassFilter scale well, with a speedup factor of about 14 for all cluster configurations. The shared bus is a communication bottleneck for DES, MatrixMult, and FFT. The advantages of WB compared to AXI (e.g., 4.9% for MatrixMult) can be explained by the reduced write performance of AXI (1 write requires 2 cycles). If a single master is accessing the bus extensively, the CPU FIFO cannot hide this drawback of AXI. However, AXI promises better results in MPSoCs that, e.g., contain a DMA controller and for other programming models like OpenCL. The NoC performance is comparable to a WB and AXI crossbar for most applications. Some applications show a performance decrease for the NoC configurations. The NoC performance for AutoCor, Low Pass Filter, MatrixMult, and RadixSort is low compared to the single cluster configuration (e.g., 68.7% performance decrease for MatrixMult and 2x2x4). For BatcherSort, the NoC-based MPSoC shows a speedup compared to the crossbar interconnects because the NCI acts as an DMA controller. This shows that the most efficient MPSoC configuration highly depends on the considered benchmark.

VI. Conclusion

In this work we evaluated different interconnect fabrics for our embedded multiprocessor system CoreVA-MPSoC. The scalability of bus-based interconnects is analyzed for 8 to 32 CPU cores, WB and AXI bus standard, shared bus and crossbar topologies, and a 2-D mesh NoC. The AXI crossbar interconnect and NoC have the same area requirements for 16 CPUs. For 32 CPUs, the area of the AXI crossbar configuration is 15.7% larger compared to our hierarchical NoC-based 2x2x8 MPSoC. These results show that in terms of area requirements the reasonable maximum size for a full crossbar interconnect is 16 CPUs for the considered 28 nm FD-SOI technology. The execution of different streaming applications for 16 CPUs shows an advantage of AXI crossbar over AXI shared bus (4.35% on average) and NoC (28.04% for 2x2x4). Future work will analyze larger MPSoC configurations and different NoC topologies. In addition we will examine partial crossbars and different memory topologies within a cluster.

Acknowledgments

This research was supported by the ATN – DAAD Joint Research Co-operation Scheme: Tightly Coupled Software Tools and Adaptable Hardware for Resource Efficient Multiprocessor Architectures, the DFG CoE 277: Cognitive Interaction Technology (CITEC), and the German Federal Ministry of Education and Research (BMBF) within the Leading-Edge Cluster "Intelligent Technical Systems OstWestfalenLippe" (it's OWL), managed by the Project Management Agency Karlsruhe.

References

[1] L. Benini *et al.*, "P2012: Building an ecosystem for a scalable, modular and high-efficiency embedded computing accelerator," in *DATE*. IEEE, 2012, pp. 983–987.

[2] B. D. de Dinechin *et al.*, "A Distributed Run-Time Environment for the Kalray MPPA-256 Integrated Manycore Processor," in *Procedia Computer Science*. Elsevier, 2013, pp. 1654–1663.

[3] "E64G401 Epiphany 64-Core Microprocessor," Adapteva, Inc., Tech. Rep., 2014. [Online]. Available: http://www.adapteva.com/epiphanyiv

[4] F. Angiolini *et al.*, "Contrasting a NoC and a traditional interconnect fabric with layout awareness," in *DATE*. IEEE, 2006, pp. 124–129.

[5] R. Kumar *et al.*, "Interconnections in Multi-Core Architectures: Understanding Mechanisms, Overheads and Scaling," in *ISCA*. IEEE, 2005, pp. 408–419.

[6] S. Lütkemeier *et al.*, "A 65 nm 32 b Subthreshold Processor With 9T Multi-Vt SRAM and Adaptive Supply Voltage Control," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 8–19, 2013.

[7] "OpenCores Project." [Online]. Available: http://opencores.org/

[8] "AMBA AXI and ACE Protocol Specification," 2013. [Online]. Available: http://www.arm.com/products/system-ip/amba/

[9] T. Jungeblut *et al.*, "A TCMS-based architecture for GALS NoCs," in *ISCAS*. IEEE, 2012, pp. 2721–2724.

[10] W. Kelly *et al.*, "A Communication Model and Partitioning Algorithm for Streaming Applications for an Embedded MPSoC," in *Int. Symp. on System on Chip (SoC)*. IEEE, 2014.

[11] W. Thies *et al.*, "StreamIt: A Language for Streaming Applications," in *Int. Conf. on Compiler Construction*. Springer, 2002, pp. 179–196.