Towards Architectural Support for Bandwidth Management in Mixed-Critical Embedded Systems

Ioannis Christoforakis[*] , Maria Astrinaki and George Kornaros[†]
Technological Institute of Crete
Computer Engineering Department
Heraklion, Estauromenos, Greece
P.O. Box 71500
[ych, kornaros]@ie.teicrete.gr

ABSTRACT

Mixed-critical platforms require an on-chip interconnect and a memory controller capable of providing sufficient timing independence for critical applications. Existing real-time memory controllers, however, either do not support mixed criticality or still fail to ensure negligible allow a certain degree of interference between applications.

On the other hand, Networks-on-Chip manage the traffic injection rate mainly by employing complex techniques; either back-pressure based flow-control mechanisms or rate-control of traffic load (i.e. traffic shaping). This work proposes such a Traffic Shaper Module that supports both monitoring and traffic control at the on-chip network interface or the memory controller.

The advantage of this Traffic Shaper Module is that at system level it provides guaranteed memory bandwidth to the critical applications by limiting traffic of non-critical tasks.

The system is developed in the Xilinx ZYNQ7000 System-on-Chip while the measurements were captured on a Zedboard development board. By enabling the Traffic Shaper in our architecture we achieved fine-grain bandwidth control with negligible overhead, while providing bandwidth of only 0.5-5 percent less than the theoretical specified bandwidth.

CCS Concepts

•**Computer systems organization** → *Embedded and cyber-physical systems;* Embedded systems; Embedded hardware;

[*]Associated Researcher Intelligent Systems and Computer Architecture(ISCA) Lab, TEI Of Crete and PhD candidate of Universita Politecnica Delle Marche of Ancona

[†]Dr G. Kornaros is currently an Assistant Professor with the Technological Educational Institute of Crete, Heraklion and Vice Director of Intelligent Systems and Computer Architecture(ISCA) Lab.

Keywords

Security; Mixed Criticality; Traffic Shaping; Network-On-Chip; Memory Bandwidth Control

1. INTRODUCTION

Nowadays, the modern Systems-on-Chips (SoCs) present a complex growth. Power, area and cost constraints [13], due to resource sharing, occur since the amount of applications mapped to a single system have been increasing. Applications characterized by real-time constraints exist alongside with applications with no constraints thus creating mixed time-criticality systems. Large numbers of applications run simultaneously in different combinations or use-cases and are dynamic, therefore they can be dynamically stopped or started. In order to guarantee satisfaction in application requirements, the system has the need to be verified for each of the use-cases. The verification complexity grows exponentially with the number of applications if they can be combined arbitrarily.

In the case of the active use-case change, in order to adapt to the new set of applications and requirements, the hardware components may have to be reconfigured. Even during reconfiguration, the running applications must demonstrate correct behavior. For this reason, the transitions between use-cases result in additional analysis complexity. Finding a common analysis model has been proved difficult when mixed time-critical applications and different models of computation are combined. Even if such a model exists, a single change in an application requires all use-cases, in which the application is active, to be re-verified in general.

A recent trend in real-time systems is to integrate tasks and components of different criticality levels on the same hardware platform. The objective of such mixed-criticality systems [1], [2] is to save space, weight, or energy by reducing the number of computation platforms, and at the same time to provide safety guarantees for the critical components of the system. Timing predictability is an important property when designing such systems, especially for the safety-critical components. Worst-case execution time (WCET) analysis [3], [4] becomes significantly easier if the hardware is more predictable. Many researchers have explored the possibility of designing predictable processors [5, 6, 7] and predictable memory hierarchies [9].

To provide guarantees for time-critical applications it is mandatory to precisely control the usage of system resources. This work describes a methodology for bandwidth man-

agement on Network-on-Chip (NoC) interfaces and system memory traffic, by using custom circuitry with minimal overhead. In particular, we demonstrate how to achieve this with the use of a specialized hardware Traffic Shaper Module (TSM); through attaching a TSM instance at any initiator Network Interface (NI) we control the number of accesses based on configured bandwidth and regulate the flow of traffic. We present a prototype system based on the ZYNQ7000 Processing System-on-Chip [12]. More specifically, we make the following contributions:

- introduce a low overhead *Traffic Shaper Module* at register-transfer level (RTL) that allows precise fine-grain traffic control in bytes/clock cycle; the block may provide the possibility for both monitoring, as well as for guaranteeing maximum bandwidth at an initiator on-chip network interface
- propose a technique that can be used to enforce bitrates different than what a physical interface is capable of
- prove that, by enabling the Traffic Shaper Module in our architecture, we achieve a performance overhead of just 0.5-5%, in contrast to the theoretical bandwidth

The rest of this paper is organized as follows. In Section II this work is positioned with respect to related works. Section III presents the proposed architecture and the developed traffic control methodologies, while experiments are shown in Section IV, and finally Section V concludes the paper.

2. RELATED WORK

There is a range of previous works for creating systems that support quality of service with mixed-criticality techniques both in register-transfer level and in the operating system and kernel level. For instance, several techniques for predictable real-time DRAM controllers have been proposed in previous works, although not all are suitable for mixed-criticality systems, since non-critical task performance cannot be sacrificed too much for critical task predictability. In a real-time system, bounded latency and interference must be considered, in addition to overall throughput and fairness. Kim et al [14] exhibited a DRAM controller that is able to separate the critical from the non-critical memory access groups (MAGs). They define algorithms for computing safe and tight upper bounds of worst-case latencies, resulting in predictable memory accesses for critical MAGs. In [15] developers designed a reconfigurable SDRAM controller. This controller offers both predictable and composable service, making it a suitable SDRAM resource for use in virtual platforms. Composability is enabled by the use of composable memory patterns combined with TDM arbitration, and they show that the worst-case performance degradation is negligible. The work in [16] deals with the problem of mixed time criticality workloads in the context of an SDRAM controller. The main idea is to allow the command scheduler to exploit locality that presents itself within the time window that naturally exists between opening and closing a row. The controller can exploit a fraction of the locality available in the request stream, without increasing the worst-case schedule length. Compared to a close-page policy (keeping a DRAM bank in idle state), the average execution time of the benchmark applications is reduced by 7.9% using the conservative open-page, while still satisfying the constraints of the Field Resource Tracker (FRT) application (guarantee enough worst-case performance to satisfy requirements).

Another technique is the one presented in [17]. This paper presents FlexPRET, a fine-grained multi-threaded processor designed to exhibit architectural techniques useful for mixed criticality systems (e.g. scratchpad memories). The designers claim that their system provides hardware-based isolation to hard real-time threads while allowing soft real-time threads to efficiently utilize processor resources. There are also timing instructions, extending the RISC-V ISA, that enable cycles to be reallocated to other threads when not needed to satisfy a temporal constraint. They even provide a concrete soft-core FPGA implementation, which is evaluated for resource usage.

A further option that can also be combined with our proposed mechanisms from our perspective is the implementation of algorithms for Mixed Criticality Optimization as in [18], where the optimization of mapping and partitioning is performed at the same time, and not separately.

3. TRAFFIC CONTROL METHODOLOGY

Initially, we introduce the design of a modern heterogeneous SoC to support a mixture of critical and non-critical computing entities with the requirement to provide application isolation, in terms of usage of system resources, e.g., memory bandwidth.

3.1 System Architecture

We target embedded devices that can host applications with different requirements in terms of bandwidth either of system memory or of other on-chip components. We assume a Network-on-Chip viewed as an interconnect intellectual property module with conventional read and write semantics; the network interface offers read/write and block transfers. Our architecture is based on ZYNQ System-on-Chip of Xilinx. We utilize both the Processing System that consists of the dual ARM Cortex-A9 CPU [12], as well as the Programmable Logic (PL) area, wherein we integrate the Traffic Shaper Module (TSM). More precisely, our architecture involves the following components and attributes.

- It utilizes the Processing System (PS) Direct Memory Access (DMA) controller that is used for burst transactions.
- The CPU routes accesses to the memory through the GP port to the PS DDR3 memory. A central interconnect is located within the PS that comprises multiple switches in order to connect to the system resources by using the AXI point-to-point channels for communicating addresses, data, and response transactions between master and slave ports [12]. This ARM AMBA 3.0 interconnect implements a full array of the interconnect communications capabilities and overlays for QoS, debug, and test monitoring. The interconnect manages multiple outstanding transactions and is architected for low-latency paths for the ARM CPUs and for the PL master controllers. The accesses from the PS DMA cross only one GP Port and HP port (HP0), as shown in Fig. 1.
- The device integrates a Microblaze soft-processor [10]

that is used to monitor resources that are used by the main processor (A9 CPU) and to collect information from the AXI performance monitor. Even though monitoring and controlling of system resources, i.e. NoC and memory bandwidth must be implemented with customized hardware to offer fine-grain control and response times, a software-oriented approach (using MicroBlaze) offers a coarser-grain solution but more flexible.

- The device uses a custom Address Remap Block, to remap addresses that arrive through the PS and redirects them to the DDR.

- Custom circuitry is integrated at RTL level, which provides the ability of monitoring, control and supplying guaranteed bandwidth for critical applications.

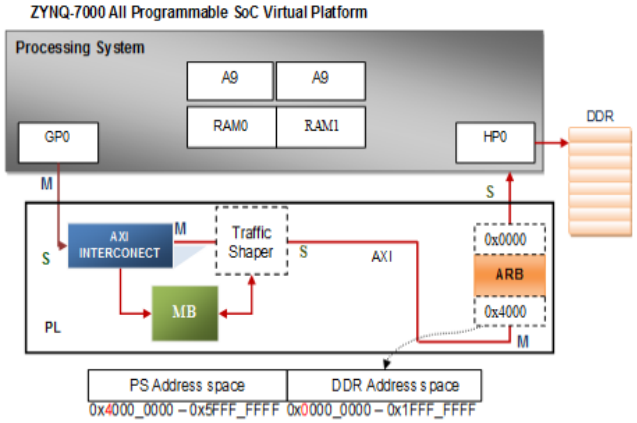

Figure 1: Architecture layout supporting precise bandwidth shaping through the use of the hardware Traffic Shaper and embedded microcontrolled manager in a dedicated MicroBlaze soft-processor (MB); memory traffic flows through the GP port to the DDR3 memory while TSM monitors and controls at byte granularity

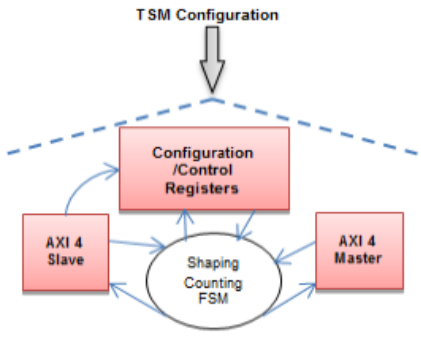

Figure 2: Traffic Shaper Module (TSM) internals as implemented in ZYNQ-7000; TSM configuration can occur on the fly and takes effect only after the current time window

3.2 Traffic Shaper Module (TSM)

Shaping is a QoS (Quality-of-Service) technique that we can use to enforce pre-specified bitrates, different than what the physical interface is capable of. TSM is a hardware block that implements control of incoming data traffic which actually consists of *read* and *write* transactions, following the AXI4 protocol [22]. More precisely, the developed TSM controls the number of accesses based on configured/programmed bandwidth and regulates the flow of traffic, in order to apply restrictions on the consumed bandwidth. Figure 1 shows the prototype system that integrates the TSM. The proposed architecture contains the registers that are listed in Table I. The designer's options are, a) to attach it on a Master Interface and control it via the main CPU, b) to connect the TSM control port and establish the overall management of the TSM to another independent core, which thus offloads the burden to dynamically monitor and evaluate QoS per connection or per process; the main CPU (ARM Cortex-A9) communicates with that core only to provide configuration parameters. In this work we chose the second method.

TSM utilizes a separate AXI-lite interface for configuration. The TSM main components are shown in Figure 2. The TSM configuration is done via the MicroBlaze. The programming of registers determines the maximum data transfer bytes to transfer in a time window. The *AxLen* signals, *AxSize* and *AxValid* are responsible for indicating a new transaction and whether this is a single word or a burst. The DMA inside the PS supports AXI burst lengths of 1 to 16 transfers, for all burst types [3]. However, the AXI4 protocol extends the burst length support for the INCR burst type to support 1 to 256 transfers. The burst length for AXI3 is defined as, Burst_Length = AxLEN[3:0] + 1, while the burst length for AXI4 is defined as, Burst_Length = AxLEN[7:0] + 1, to accommodate the extended burst length of the INCR burst type in AXI4.

Table 1: Description of Traffic Shaper Module basic components

Register	Description
maxcc	Stores the Maximum time limit (cycles)
maxbw	Stores the maximum transfer bytes limit, maximum bytes to transfer in *maxcc* time interval
acc	Accumulating counter adding the new transaction bytes, should be less than maxbw limit
totalacc	Sums all the new accâĂŹs and keep the values after the reset
burst	Stores each new AXI4 transaction, burst or single word
cccnt	Clock cycle counter; in each clock cycle cccnt increases by 1 measuring total clock cycles

4. EVALUATION

4.1 Methodology

Several different ways and methods exist, so as to monitor the activity of a processor or a module. As we presented previously this can be done by designing a controller that supports mixed-criticality service. Software techniques also exist, which support control at task level or at OS level [19].

Our methodology supports both control and monitoring on the Network-on-Chip (NoC) Interface, without requiring the re-design or tampering with the NoC routers or the memory controller [20]. The Traffic Shaper Module provides guaranteed bandwidth to the critical applications by limiting consumed bandwidth of the non-critical tasks.

We used a tightly system-coupled configuration to run our applications. Tightly system-coupled software means routines (modules or full applications) that work on only one type of system since these are dependent on each other and on the system configuration. For example, the driver of an embedded cyber-physical system device requires extensive programming changes to work in another environment, but is usually optimized to offer predictability, minimal latency and even fault-tolerance. We ran two stand-alone (baremetal) applications in A9 CPU and MicroBlaze. The MicroBlaze is responsible to monitor and control the configuration of the TSM registers. The A9 CPU can be configured to be in control of this block (start, restart, counters initialization and monitoring), but the main goal is to dedicate the processor only to the application running on it. In our case the accesses cross only one GP port (GP0) towards the slave PS interconnect (as shown in Fig. 1). If the DMA engine is utilized to perform bulk data transfers, then the DMA sequence of operations is as follows:

- DMA Configuration
- Setup DMA Interrupt
- DMA Initialization
- DMA Start Transfer
- DMA Check handler done
- DMA Stop and reset

If, we want to exploit the maximum theoretical bandwidth that manufacturers specify [12], then the NoC specificities and the memory controller properties should be carefully exploited. For instance, in the ZYNQ architecture any transaction must use specific ports and route along the interconnection scheme to avoid conflicts. The maximum supported bus clock is 533 MHz in the DDR3 module for all speed grades, reaching a theoretical maximum bus bandwidth of 1333 Mb/s. All DMA transactions use AXI interfaces to move data between the on-chip memory, the DDR memory and the slave peripherals in the PL.

Initially, in evaluating the accuracy of bandwidth monitoring we used a software only approach. We used a baremetal application executing on the MicroBlaze which, with the aid of a hardware AXI performance monitor, captures the activity of the AXI interconnect. The A9 CPU drives the PS DMA to send traffic over this interconnect. As Figures 3 and 4 show, when fine-grain resolution is adopted the software monitoring delivers poor results, while when the hardware TSM is activated the mean error is negligible.

There are many options and routes to transfer data in an advanced SoC such as ZYNQ. Two typical DMA transaction examples include: 1) memory to memory (On-chip memory to DDR memory) 2) memory to/from a PL peripheral (DDR memory to PL peripheral). The incoming traffic arrives only from one GP Port crossing the AXI HP0 port and the S2 OCM port [12].

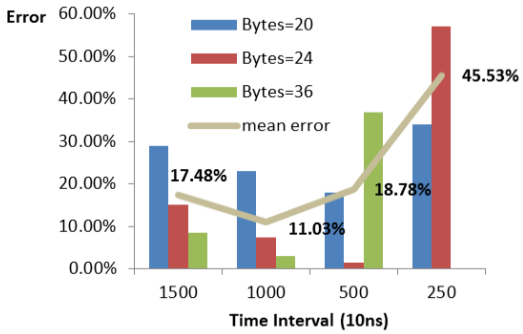

Figure 3: Comparison of accuracy when scaling the time window using varying data unit size

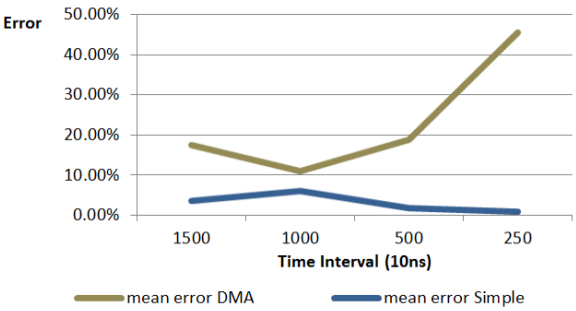

Figure 4: For four different time intervals (1500, 1000, 500, 250 cycles) we measure the error rate that raises between the theoretical and the measured traffic bandwidth without activation of the TSM. For each interval we count three different data size transfers (20, 24, 36 bytes). For example, as we observe in the graph, for 1500 cycles the average difference arises to 17.48% of bandwidth that was measured compared to the theoretical values that were calculated. By enabling the TSM the error rate in the worst case reaches 6.07%

We developed two applications to move data using the first configuration. In one scenario, one alleged critical application runs on the A9 CPU in a baremetal fashion and the MicroBlaze enables the TSM and generates traffic. As Figure 5 shows, in the first scenario the MicroBlaze writes to the memory by using the DMA engine, thus reaching 23.5MB/sec. We also notice that the traffic from the A9 CPU towards the DDR is not affected, and this is because the TSM provides the maximum throughput to the critical application. In the second scenario in Figure 6 we clearly observe that the process on the A9 CPU that makes the Reads and Writes is influenced. The traffic generator running in the Microblaze is activated at specific times and significantly affects the bandwidth received by the A9 CPU.

As long as the Traffic Shaper Module is in a 'disable' mode, the DMA can reach up to a limit depending on AxLen (Burst size and Length). Moreover, as we have observed the GP Port performs poorly in throughput, as it also appears in Figure 5. When the TSM is enabled the shaper can be configured to regulate the traffic until the consumed bandwidth is less than or equal to the maximum bandwidth that

can be reached when the shaper is disabled.

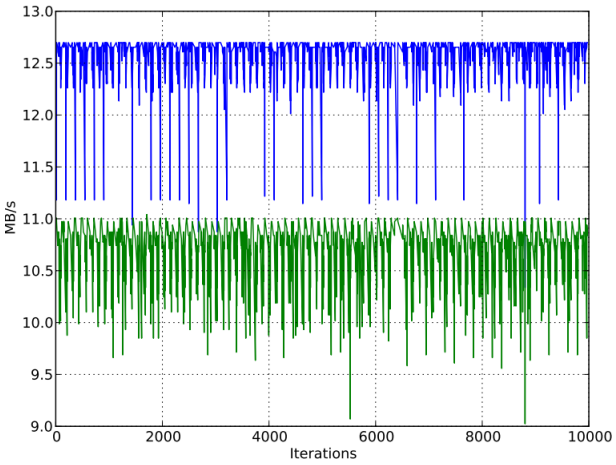

Figure 5: Nearly perfectly unaffected bandwidth served in control of the TSM

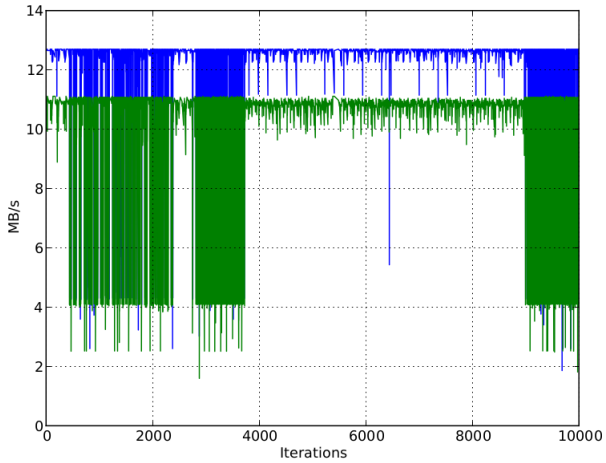

Figure 6: System impact when the TSM is deactivated; random DMAs cause irregular and uncontrolled memory bandwidth consumption

4.2 Use Case for Healthcare Application Example

Healthcare devices that integrate Bluetooth version 4.0 [21] may be found in three different locations on the patient's body, which transmit to a gateway device similar to the ZYNQ-based one that we presented previously. In this particular scenario we consider such a device to run a critical application that can process streaming data that reach 25Mbit/sec or 3 MB/sec each [21]. Consequently, we could reach up to N devices*3ÎIJÎŠ/sec theoretical aggregate throughput. The gateway device utilizes only a single Bluetooth interface to communicate with each external healthcare device.

In our use-case scenario that is depicted in Figure 7, the incoming data traffic can reach up to the maximum of 3MB/sec and the gateway forwards this traffic with at least equal speed. Consequently the maximum aggregate bandwidth is 6MB/sec, i.e 3MB/sec incoming traffic and 3 MB/sec outgoing traffic.

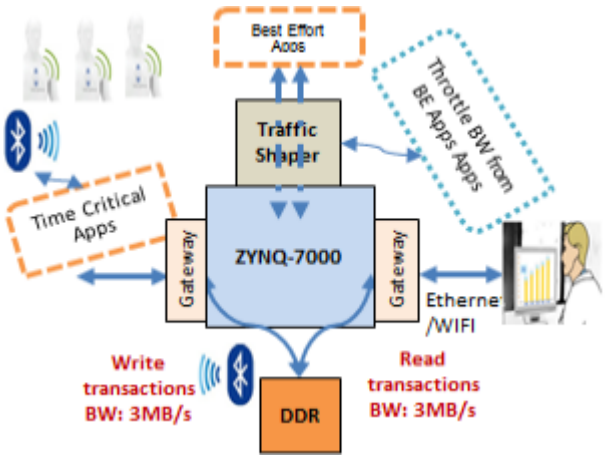

Figure 7: Healthcare use-case for a gateway of bluetooth enabled sensors

In addition, best-effort applications can be executing in parallel that also demand additional memory throughput. As we have analyzed in the previous section, when the Cortex A9 CPU communicates through the AXI_GP0/1 ports to the system memory we can achieve 10.797 MB/sec maximum throughput. The Bluetooth communicates via the ZYNQ gateway in order to feed the information to the user via a local Ethernet or WIFI connection.

When enabling the Traffic Shaper Module, we can throttle the throughput of the best effort applications so as to provide guaranteed BW for the critical function of conveying the devicesâĂŹ data, which may contain important messages for the patients that must receive guaranteed delivery to the user. The TSM can be configured so that it can support the desired service level even to the maximum throughput (9 MB/sec).

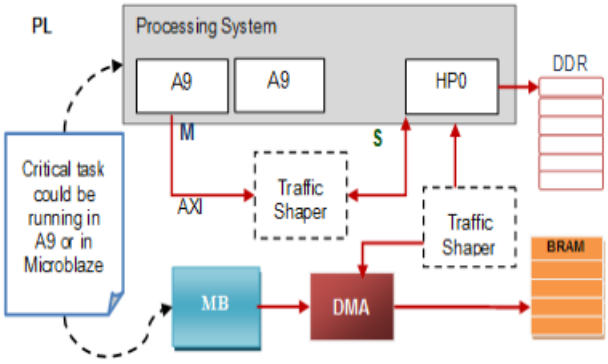

Figure 8: Healthcare device internal organization; a SoC with controlled resource usage

4.3 Simulation of use case scenario using a random traffic generator

The above scenario was implemented by creating a Random Traffic Generator (RTG), as shown in Figure 8, which can generate up to 3MB/sec read/write traffic in total. The

RTG runs on the Microblaze whereas a non-critical application with its own demands in throughput runs in parallel on the A9 CPU with none impact to its performance.

In addition, notice that even in the case where the Microblaze has the need to consume the maximum bandwidth (3MB/Sec), the TSM ensures that the application's demands in terms of throughput, which runs in parallel on the A9 CPU, is not affected by the needed bandwidth that is consumed by the Microblaze.

5. CONCLUSIONS

A multicore embedded system is demonstrated in this paper, where the bandwidth is controlled by establishing a hardware Traffic Shaper Module. In such systems, in order to ensure system predictability and applications' properties such as criticality, we have to control the usage of systems resources. We achieve this by using a Traffic Shaper Module; its goal is to accurately control the number of accesses to provide the desired configured bandwidth. We presented a methodology for bandwidth management suitable for NoC interface and memory controller with the usage of custom circuitry and a minimal software monitoring application. The system was prototyped using the Xilinx ZYNQ7000 System-on-Chip and the measurements extracted from a zedboard development board. By enabling the Traffic Shaper in our architecture we achieved very fine-grain control with negligible overhead, while providing bandwidth of only 0.5-5 percent less than the theoretical bandwidth specified.

Acknowledgments

This research has been co-financed through the European FP7 DREAMS project under the Grant Agreement No. 610640.

6. REFERENCES

[1] S. Vestal, *Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance*, in 28th IEEE Int'l Real-Time Syst. Symp., (RTSS), pp. 239âĂŞ243, Dec. 2007

[2] A. Burns and R. Davis, *âĂIJMixed criticality systems: A review*, Dept. of Computer Science, University of York, Tech. Rep, Fourth Edition, 2014

[3] R. Wilhelm et al., *The Worst-Case Execution-Time Problem Overview of Methods and Survey of Tools*, ACM Trans. on Embed. Comput. Syst., vol. 7, pp. 36:1-36:53, May 2008

[4] C. Ferdinand and R. Wilhelm, *Efficient and precise cache behavior prediction for real-time systems*, Real-Time Systems, vol. 17, no. 2, pp. 131-181, 1999

[5] S. A. Edwards and E. A. Lee, *The case for the precision timed (PRET) machine*, in Proc. of the 44th annual Conf. on Design Automation, pp. 264 - 265, Jun 2007

[6] M. Schoeberl, *A Java processor architecture for embedded real-time systems*, Journal of Syst. Archit., vol. 54, no. 1-2, pp. 265 -286, 2008

[7] I. Liu, J. Reineke, D. Broman, M. Zimmer, and E. Lee, *A PRET microarchitecture implementation with repeatable timing and competitive performance*, in Intl. Conf. on Computer Design (ICCD), pp. 87-93, Sep. 2012

[8] Y. Kim, D. Broman, J. Cai, and A. Shrivastaval, *WCET-Aware Dynamic Code Management on Scratchpads for Software-Managed Multicores*, in Proc. of the 20th IEEE Real-Time and Embedded Technology and Application Symp. (RTAS), 2014

[9] M. Schoeberl, *A time predictable instruction cache for a java processor*, On the Move to Meaningful Internet Systems 2004: OTM 2004 Workshops, Springer, pp. 371-382, 2004

[10] *MicroBlaze Processor Reference Guide*, Xilinx UG984, April 2, 2014

[11] S. Baruah, V. Bonifaci, G. D'Angelo and L. Haohan, A. Marchetti-Spaccamela, N. Megow, and L. Stougie, *Scheduling Real-Time Mixed-Criticality Jobs*, IEEE Transactions on Computers, vol 61, no 8, 2012

[12] *Zynq-7000 All Programmable SoC Technical Reference Manual UG585 v1.10*, Xilinx, February 23, 2015

[13] P. Kollig et al., *Heterogeneous Multi-Core Platform for Consumer Multimedia Applications*, In Proc. DATE, 2009.

[14] H. Kim et al., *A Predictable and Command-Level Priority-Based DRAM Controller for Mixed-Criticality Systems*, Real-Time and Embedded Technology and Applications Symposium (RTAS), 2015

[15] S. Goossens, J. Kuijsten, B. Akesson and K. Goosens, *A reconfigurable real-time SDRAM controller for mixed time-criticality systems*, Hardware/Software Codesign and System Synthesis (CODES+ISSS), International Conference, 2013

[16] S. Goossens, B. Akesson and K. Goosens, *Conservative open-page policy for mixed time-criticality memory controllers*, Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013

[17] M. Zimmer, D. Broman, C. Shaver and E.A. Lee, *FlexPRET: A Processor Platform for Mixed-Criticality Systems*, Proceedings of the 20th IEEE Real-Time and Embedded Technology and Application Symposium (RTAS), Berlin, Germany, April 15-17, 2014

[18] T.-S. Domitjian and P. Pop, *Design Optimization of Mixed-Criticality Real-Time Applications on Cost-Constrained Partitioned Architectures*, International Conference on Embedded and Real-Time Computing Systems and Applications, 2013

[19] G. Ciocarlie, H. Schubert and R. Wahlin, *A Data Centric Approach for Modular Assurance*, Workshop on Real-time, Embedded and Enterprise-Scale Time-Critical Systems, 23 Mar 2011

[20] G. Kornaros, I. Papaefstathiou, A. Nikologiannis and N. Zervos, *A fully-programmable memory management system optimizing queue handling at multi gigabit rates*, Proceedings of the 40th annual Design Automation Conference, pp. 54–59, 2003

[21] "Specification Documents", Bluetooth SIG, *www.bluetooth.org*

[22] *AXI Reference Guide UG761 V13.1*, Xilinx, March 7, 2015