B. Juurlink, W. Karl (Hrsg.)
Proceedings 28th PARS Workshop

A Quantitative Analysis of Processor Memory Bandwidth of an FPGA-MPSoC

Robert Drehmel[1], Matthias Göbel[2], Ben Juurlink[3]

Abstract: System designers have to choose between a variety of different memories available on modern FPGA-MPSoCs. Our intention is to shed light on the achievable bandwidth when accessing them under diverse circumstances and to hint at their suitability for general-purpose applications. We conducted a systematic quantitative analysis of the memory bandwidth of two processing units using a sophisticated standalone bandwidth measurement tool. The results show a maximum cacheable memory bandwidth of 7.11 GiB/s for reads and 11.78 GiB/s for writes for the general-purpose processing unit, and 2.56 GiB/s for reads and 1.83 GiB/s writes for the special-purpose (real-time) processing unit. In contrast, the achieved non-cacheable read bandwidth lies between 60 MiB/s and 207 MiB/s, with an outlier of 2.67 GiB/s. We conclude that for most applications, relying on DRAM and hardware cache coherency management is the best choice in terms of benefit-cost ratio.

Keywords: Memory Bandwidth; MPSoC; Interconnects

1 Introduction

As MPSoCs evolve into more complex devices containing increasingly heterogeneous processing units, a growing variety of specialized memories becomes available to developers. Although logically connected to a single system bus, the processors and memories in modern MPSoCs are physically connected using an intricate net of interconnects with distinctive performance characteristics. As interconnects are often seen as the limiting factor of SoC performance[1], being aware of their influence on bandwidth can play an important role in the process of designing a system. The achievable bandwidth when accessing different memories depends on various factors, e.g. the processor's performance, the pathway of interconnects between the processor and the memory, and the cacheability associated with the memory region. To make informed decisions about which memory to use for a given task, developers need to understand their capabilities and limitations and need to be able to compare their performance systematically.

[1] Technische Universität Berlin, Embedded Systems Architecture, Einsteinufer 17 (6. OG), 10587 Berlin, Germany
drehmel@campus.tu-berlin.de
[2] Technische Universität Berlin, Embedded Systems Architecture, Einsteinufer 17 (6. OG), 10587 Berlin, Germany
m.goebel@tu-berlin.de
[3] Technische Universität Berlin, Embedded Systems Architecture, Einsteinufer 17 (6. OG), 10587 Berlin, Germany
b.juurlink@tu-berlin.de

The main contribution of this paper is a systematic and comprehensive evaluation of the processor memory bandwidth achievable using the different processing units of the Xilinx Zynq Ultrascale+ MPSoC line.

This paper is organized as follows. Section 2 discusses related work, section 3 gives a brief overview of the platform used, section 4 outlines the design and implementation of the bandwidth measurement tool, section 5 details the experimental setup and presents the results of the evaluation. Finally, section 6 draws conclusions based on the results.

Note that in the following, when using the term *memory bandwidth*, we mean the observed bandwidth (or throughput) when accessing a certain memory, including and emphasizing the multiplicity of components involved in the process, like caches, store buffers, interconnects, memory controllers, and the memory itself.

2 Related Work

Göbel et al.[2] conducted thorough system bus bandwidth analyses of chips from the previous generation of FPGA-MPSoCs: the Intel Cyclone-V, the Xilinx Zynq-7020, and the Zynq-7045. Choi et al.[3] evaluated specialized systems with an FPGA and processor combination, i.e. Intel Xeon E5-2680v2/Stratix V on an Intel HARP board and Intel Xeon E5-2620v3/Xilinx Virtex 7 on an Alpha Data board.

Closely related to our work, Bansal et al.[4] recently evaluated the memory subsystem of the Zynq Ultrascale+, but with a focus on giving advice for the design of real-time applications. They measured bandwidth only for a single processing unit running Linux to three different memories for a specific amount of time (5 seconds) and using unspecified instructions generated by the compiler. In contrast, we measured sequential accesses for varying transfer sizes to four different memories using a sophisticated standalone application, for two processing units, and for two types of instructions.

3 Xilinx Zynq Ultrascale+

A chip of the Zynq Ultrascale+ series provides an Application Processing Unit (APU) and a Real-Time Processing Unit (RPU)[5]. The APU consists of an ARM Cortex-A53 processor[6] with four cores, each with 64 KiB L1 cache (separate 32 KiB for instructions and data) and a shared 1 MiB L2 cache. The RPU consists of two single-core ARM Cortex-R5 processors[7], each with 32 KiB L1 cache (combined for instruction and data). One of the two main regions of the chip, the *Processing System* (PS), contains common MPSoC components; the other one, the *Programmable Logic* (PL), holds the integrated FPGA fabric. We used the Zynq Ultrascale+ ZU9EG chip for our tests. It supplies the system designer with memories of four main categories, namely

- external Dynamic RAM (DRAM) through an memory controller integrated in the PS,

- 256 KiB On-Chip Memory (OCM),
- 256 KiB Tightly-Coupled Memory (TCM), 128 KiB coupled to each Cortex-R5 of the RPU, again divided into two separate 64 KiB blocks (ATCM and BTCM),
- 32.1 MiB Block RAM (BRAM) in 912 blocks in the PL.

Some chips of the series include so-called UltraRAM on-chip memory in the PL. Unfortunately, this is not the case with the ZU9EG, therefore we were not able to include this kind of memory in our tests.

4 Benchmark Tool

The core functionality of our benchmark tool is to measure the time it takes the processor to complete the execution of a bandwidth test function in a specific execution context. The context of the execution of the test function – and the arguments passed to it – are highly parameterizable.

The tool allows the user to set parameters to

- select various forms of cacheability for inner and outer domains (ARM-specific),
- select the ISA to use for the memory access (base or Adv. SIMD)[8][9],
- select the access type (read or write),
- select the width of the memory transfer (in powers of two),
- enable or disable the data cache,
- specify the number of rounds of reading and writing before the measurements (to fill the cache with read-allocate and write-allocate, respectively),
- select the shareability domain (ARM-specific), and
- enable or disable APU coherency (Zynq Ultrascale+-specific).

The user can define a stack of functions to automatically gather results for a set of parameters. Each function included in the stack iterates through all possible values of a single parameter and calls the next function in the stack after setting a new value. The last function in the stack is the test run function that evaluates all the currently set parameters, sets up the context (e.g., hardware configuration and memory attributes) accordingly and runs the test. The results for each test run are saved along with the parameters used for the test. After all tests completed, the user can query the result database for further processing, for example to perform advanced analyses to find statistical anomalies or to generate plots in LaTeX documents.

To retain full control of the hardware configuration, we developed our benchmark tool as a bare-metal application using Xilinx's *standalone* library. The tool currently supports Cortex-A53 and Cortex-R5 processors through a hardware abstraction layer that provides functionality like management of hardware cycle counters, caches, Memory Protection Units (MPUs), and Memory Management Units (MMUs).

The hardware abstraction layer also provides a set of hand-optimized read and write benchmark test functions. For a given processor, if $2^{W_{\text{bus}}}$ bytes is the width of the master

Processor	Mnemonic	Access width (bytes)	ISA	Type
APU	`LDP`	$2 \cdot 8 = 16$	ARMv8-A	load
APU	`STP`	$2 \cdot 8 = 16$	ARMv8-A	store
APU	`LD1`	$8 \cdot 8 = 64$	ARMv8-A Adv. SIMD	load
APU	`ST1`	$8 \cdot 8 = 64$	ARMv8-A Adv. SIMD	store
RPU	`LDM`	$8 \cdot 4 = 32$	ARMv7-R	load
RPU	`STM`	$8 \cdot 4 = 32$	ARMv7-R	store
RPU	`VLDM`	$8 \cdot 8 = 64$	ARMv7-R Adv. SIMD	load
RPU	`VSTM`	$8 \cdot 8 = 64$	ARMv7-R Adv. SIMD	store

Tab. 1: Instructions used in the bandwidth test functions

interface to the system bus, and $2^{W_{\text{insn}}}$ bytes is the largest number of bytes transferable with a single instruction, for each n in $[W_{\text{bus}} : W_{\text{insn}}]$ an optimized function is provided that transfers 2^n bytes in one loop iteration. Assuming $W_{\text{bus}} \leq W_{\text{insn}}$, for a test width of W_{test} bytes, the function that is optimized to transfer W_{f} bytes is selected, where

$$W_{\text{f}} = \begin{cases} W_{\text{bus}} & : W_{\text{test}} \leq W_{\text{bus}} \\ W_{\text{test}} & : W_{\text{bus}} < W_{\text{test}} < W_{\text{insn}} \\ W_{\text{insn}} & : W_{\text{test}} \geq W_{\text{insn}} \end{cases}$$

Each function is provided in all four possible combinations for read/write access types and base ISA/Adv. SIMD instructions. Table 1 shows the instructions used in the bandwidth test functions and their corresponding access widths.

To configure the caching behavior for the Cortex-A53, the tool sets the memory attributes in the page table entries corresponding to the tested memory region to *normal memory, inner/outer non-cacheable* and *normal memory, inner/outer write-back* to test non-cacheable and cacheable accesses, respectively. We found that disabling the data cache (by clearing the `C` bit in the `SCTRLR_ELx` register) can have a different effect (i.e. reduced bandwidth) than marking a region non-cacheable in its page table entries. We attribute this to the fact that clearing the abovementioned bit disables the data cache and the unified caches, and has non-intuitive effects such as preventing the caching of page table memory.

The boot code of the standalone library for the Cortex-R5 installs a default MPU configuration that includes a region that spans the first 2 GiB of the address space. On the Zynq Ultrascale+, the ATCM is mapped to `0x0` for each Cortex-R5. The code and data segments, the heap, and the stack of the bandwidth tool are held in a region starting at `0x10000000`. We use a region starting at `0x20000000` to test the DRAM. So instead of a single memory region that spans the first 2 GiB of the address space, beginning at the start of the address space, the tool configures three consecutive memory regions that each span 256 MiB. This is necessary to change the cacheability attributes for the TCM and the DRAM without interfering with the cacheabilty of the region the bandwidth tool runs in.

5 Evaluation

In the following, we first detail the experimental setup and subsequently present the quantitative results gathered.

5.1 Experimental setup

The benchmark tests use the following fixed parameters: enabled data cache, four rounds of prefilling the data cache with reads (read-allocate), zero rounds of prefilling the data cache with writes (write-allocate), outer shareability domain, and enabled APU coherency. We are basing our decision to leave the APU coherency enabled on the premise that most system designers will keep it enabled – it is the default setting and is useful for all except a minority of special applications.

We ran tests for every combination of the parameters *processor* (APU/RPU), *caching* (write-back/non-cacheable), *memory* (DRAM/OCM/ATCM/BRAM), *instruction type/ISA* (base ISA/Adv. SIMD), and *access type* (read/write), and *transfer size* ($N + 1$ different transfer sizes, 2^0 to 2^N bytes, where 2^N bytes is the size of the memory region tested). Every test was run 1000 times and averaged accordingly.

We tested memory accesses to 16 MiB DRAM, 256 KiB OCM, 64 KiB ATCM, and 1 MiB BRAM. The ZCU102 board used for the evaluation has 4 GiB of DDR4 memory (2133 MHz) installed. Although the four 64 KiB TCMs available on the chip can be mapped into a consecutive 256 KiB region, this is only possible when both Cortex-R5 processors are running in *lock-step* mode (coupled execution of both processors). A more common mode of operation is the antithetic *split* mode (separate execution of both processors), in which each processor can only access its own (non-consecutively-mapped) ATCM and BTCM. These and other particularities of the TCM as implemented in the Zynq Ultrascale+ are described in detail in [5]. We decided to choose a single 64 KiB ATCM as the TCM test region to cover a broader range of potential applications. The 1 MiB of BRAM (128 bit data width and 64 KiB depth) consist of 256 RAMB36 BRAM blocks (a utilization of roughly 28%) generated by a Xilinx *Block Memory Generator* [10], connected to a Xilinx *AXI BRAM Controller* [11] that is in turn connected to the PS using the HPM0 port.

5.2 Results

Non-cacheable APU memory access. First, we discuss the bandwidth results for non-cacheable APU memory accesses as displayed in figure 1. Results for reads and writes both show the maximum sustainable bandwidth. Non-cacheable read bandwidth only increases with the size of memory transferred (a slope similar to figure 5b). This is a trivial consequence of a decreasing percentage of time spent processing the prologue and epilogue instructions of the test function. Reading using the base ISA provides roughly twice as much

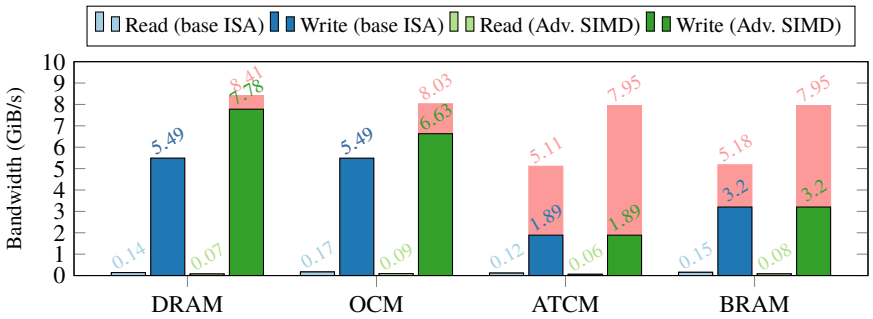
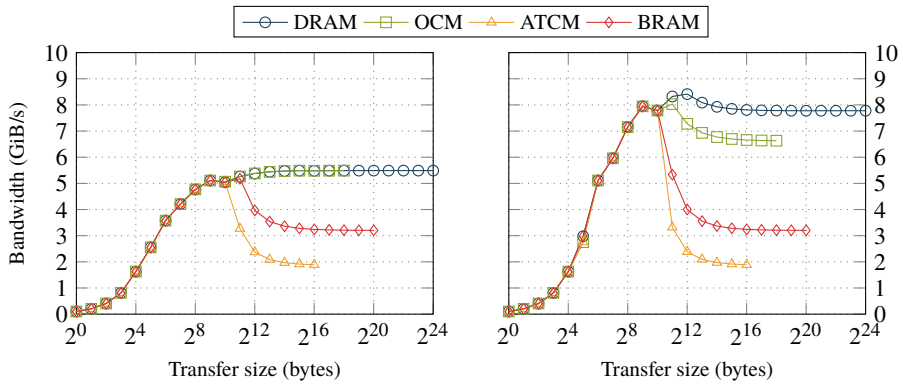


Fig. 1: Bandwidths of non-cacheable APU memory accesses (for the largest tested transfer size). denotes unsustainable or short-term write bandwidth.



(a) Base ISA (b) Adv. SIMD

Fig. 2: Bandwidths of non-cacheable APU write memory accesses

bandwidth as using the Advanced SIMD instructions. Still, none of the non-cacheable APU read bandwidths exceeds 172 MiB/s. Results for write accesses also show the maximum short-term bandwidths. While read instructions need to move data back to the processor core and therefore need to wait for the read request to finish, write requests are simply handed off to another stage. This stage then processes the queued write requests autonomously. If that stage does not have the capacity to store more outstanding write requests, the hand-off from the processor is stalled. This might happen because one or more components in the path to the memory are too slow to keep up or congestion occurs. These throttling effects are visible in figure 2. Using the base ISA write instructions (figure 2a), the throttling effect appears at transfer sizes of 2^{11} bytes and 2^{12} bytes for writes to ATCM and BRAM, respectively, while for DRAM and OCM the limiting factor is the processor core. On the other hand, the Adv. SIMD write instructions (figure 2b) show a throttling effect for all memories: starting

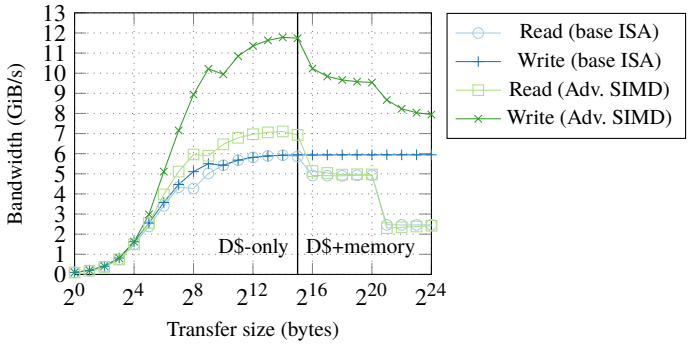


Fig. 3: Bandwidths of cacheable APU memory accesses to DRAM

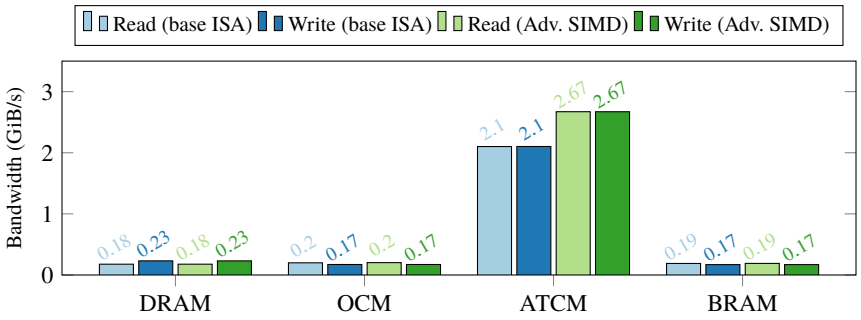
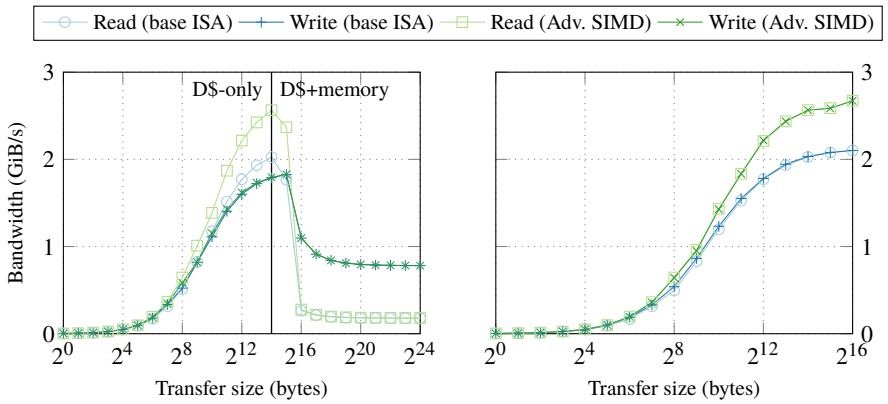


Fig. 4: Bandwidths of non-cacheable RPU memory accesses (for the largest tested transfer size)

at transfer sizes of 2^{11} bytes for ATCM and BRAM, 2^{12} bytes for OCM and 2^{13} bytes for DRAM.

Cacheable APU memory access. Figure 3 shows the cacheable APU memory bandwidths for the DRAM. As the bandwidths of cacheable APU memory accesses up to the transfer size of 2^{15} bytes are practically the same for all memories due to little to no cache misses, we show only the DRAM plot. There is a first significant drop at a transfer size of 2^{16} bytes, where the L1 data cache with its size of 32 KiB can only hold part of the data. Another drop is found at 2^{21} bytes, where the L2 cache size of 1 MiB is exhausted. The read bandwidth peaks at 7.11 GiB/s and the write bandwidth peaks at 11.78 GiB/s, both using Adv. SIMD instructions. It is notable that the Advanced SIMD instructions for writing outperform the base ISA instructions almost by a factor of two at transfer sizes around 2^{14} bytes.

Non-cacheable RPU memory access. Figure 4 shows the maximum bandwidths achievable with non-cacheable RPU memory accesses. The throttling effects exposed by the results



(a) DRAM (b) ATCM

Fig. 5: Bandwidths of cacheable RPU memory accesses

for the non-cacheable APU write memory accesses do not appear with non-cacheable RPU write memory accesses. The reason is not because the Cortex-R5 is not fast enough to force congestion. In fact, it congests an internal component; when the processor encounters a write instruction destined to access a non-cacheable memory region, it submits a write request to its *store buffer* which directly generates an AXI write request [12] on the AXI master interface [7]. The processor then has to wait for the AXI request to complete before it can submit the next write request to the store buffer. Hovering between 181 MiB/s (DRAM) and 207 MiB/s (OCM), read bandwidth to all memories except the ATCM is roughly comparable to non-cacheable APU read bandwidth, apart from the non-existence of a performance gap between base ISA and Adv. SIMD instructions that is present in the non-cacheable APU memory access results. The ATCM unveils its strength in this benchmark with read and write bandwidths of 2.1 GiB/s for ISA instructions and 2.67 GiB/s for Adv. SIMD instructions.

Cacheable RPU memory access. Figure 5a shows the bandwidths of cacheable RPU memory accesses with different transfer sizes to DRAM. Like for the APU, up to a certain transfer size, the bandwidth of cacheable memory accesses is determined by the L1 cache. For the RPU, the transfer size where cache misses and ensuing cache line fills start to noticeably impact the bandwidth is 2^{15} bytes: a minor drop occurs from 2^{14} to 2^{15} bytes, because the 32 KiB L1 cache of the Cortex-R5 is a combined instruction and data cache and therefore instructions use up some of the cache capacity. Starting at transfer sizes of 2^{16} bytes, frequent L1 cache misses drag the bandwidth down immensely as there is no L2 cache. Figure 5b shows a dissimilar behavior for the ATCM as the Cortex-R5 processor always treats accesses to its TCMs as non-cacheable accesses. Accordingly, the bandwidths for transfer sizes of 2^{16} bytes shown in figure 5b match those of the ATCM in figure 4. The

peak cacheable read bandwidth is 2.56 GiB/s (96% of the ATCM read bandwidth) and the peak cacheable write bandwidth is 1.83 GiB/s (69% of the ATCM write bandwidth).

6 Conclusions

We used our benchmark tool to systematically measure the memory bandwidth of the different processing units of an FPGA-MPSoC.

Our results show surprisingly underwhelming non-cachable read memory bandwidths, generally ranging from 60 MiB/s to 207 MiB/s, across the board for both processing units and all non-TCM memories. This extends to non-cachable write memory bandwidths on the RPU, ranging from 174 MiB/s to 237 MiB/s for non-TCM memories. Exceptions are the read and write ATCM bandwidths of up to 2.67 GiB/s, but only when accessing the ATCM from the RPU. Write memory bandwidths can, in theory, be much higher – and are in practice for the APU – but depend heavily on processor-internal request queueing. The APU achieves maximum non-cachable short-term write bandwidths ranging from 5.11 GiB/s to 8.41 GiB/s and is able to sustain a maximum bandwidth of 7.78 GiB/s to DRAM using Advanced SIMD instructions. Cacheable memory accesses from the APU show high maximum bandwidths, reaching 7.11 GiB/s for reads (Adv. SIMD) and 11.78 GiB/s for writes (Adv. SIMD). In comparison, the RPU reaches a maximum cacheable memory bandwidth of 2.56 GiB/s (Adv. SIMD) for reads and 1.83 GiB/s (Adv. SIMD) for writes.

We conclude that the use of dedicated on-chip memory, except tightly-coupled memory, has no additional benefits in terms of bandwidth over external DRAM memory. To achieve high memory bandwidth and therefore high computational performance on the Zynq Ultrascale+ platform, leveraging the use of system-wide hardware cache coherency is therefore recommended. As hardware cache coherency management has the potential – depending on the application – to reduce software complexity, we see a dependence solely on DRAM and hardware cache coherency as the best choice for most applications. On the other hand, using our findings on unsustainable non-cacheable write memory bandwidths, an application could optimize parallelism by writing chunks of memory in specific sizes to fill the write request queue and doing other processing while the write requests are being completed. The caches behave as expected, so the usual recommendations naturally apply, such as exploiting data locality by taking knowledge of specific cache implementation parameters like cache line size into account for the implementation of an application. Advanced SIMD instructions provide at least the same – in some cases vastly superior – bandwidth compared to the base ISA instructions – non-cacheable APU reads being the exception.

To gain a better understanding of interconnect behavior, further work could include methodical stress testing that focusses on inducing interconnect congestion using multiple bus masters, i.e. multiple processor cores or multiple processing units that simultaneous issue memory access requests. Evaluation of UltraRAM memory could also prove to produce interesting results.

Acknowledgements

This research was partially funded by the *German Academic Scholarship Foundation (Studienstiftung des deutschen Volkes)*.

References

[1] Luca Benini and Giovanni De Micheli. "Networks on chips: A new SoC paradigm". In: *Computer* 35 (1 Jan. 2002), pp. 70–78. DOI: 10.1109/2.976921.

[2] Matthias Göbel et al. "A Quantitative Analysis of the Memory Architecture of FPGA-SoCs". In: *Applied Reconfigurable Computing. 13th International Symposium, ARC 2017*. Lecture Notes in Computer Science 10216. Springer, 2017, pp. 241–252. DOI: 10.1007/978-3-319-56258-2_21.

[3] Young-kyu Choi et al. "A Quantitative Analysis on Microarchitectures of Modern CPU-FPGA Platforms". In: *2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2016. DOI: 10.1145/2897937.2897972.

[4] Ayoosh Bansal et al. "Evaluating the Memory Subsystem of a Configurable Heterogeneous MPSoC". In: *Proceedings of OSPERT 2018. the 14th Annual Workshop on Operating Systems Platforms for Embedded Real-Time Applications*. 2018, pp. 55–60.

[5] *Zynq UltraScale+ Device. Technical Reference Manual*. Version v1.9. UG1085. Xilinx. Jan. 17, 2019.

[6] *ARM Cortex-A53 MPCore Processor. Technical Reference Manual*. Version Issue J (r0p4). ARM DDI 0500J (ID071918). ARM. June 13, 2018.

[7] *Cortex-R5. Technical Reference Manual*. Version Issue D (r1p2). ARM DDI 0460D (ID092411). ARM. Sept. 15, 2011.

[8] *ARM Architecture Reference Manual. ARMv7-A and ARMv7-R edition*. Version Issue C.c. ARM DDI 0406C.c (ID051414). ARM. May 20, 2014.

[9] *ARM Architecture Reference Manual. ARMv8, for ARMv8-A architecture profile*. Version Issue D.a. ARM DDI 048D.a (ID103018). ARM. Oct. 31, 2018.

[10] *Block Memory Generator v8.4. LogiCORE IP Product Guide*. PG058. Xilinx. Oct. 4, 2017.

[11] *AXI Block RAM (BRAM) Controller v4.1. LogiCORE IP Product Guide*. PG078. Xilinx. Dec. 5, 2018.

[12] *AMBA AXI and ACE Protocol Specification. AXI3, AXI4, AXI5, ACE and ACE5*. Version Issue F.b. ARM IHI 0022F.b (ID122117). ARM. Dec. 21, 2017.