Managing Latency and Bandwidth in HW/SW Co-Processing

Endric Schubert, Missing Link Electronics

Abstract

Many embedded systems demand for software and hardware co-processing, Digital Signal Processing for signal conditioning, for example. Adding an FPGA as a companion chip to a modern CPU is an established concept for hardware software co-processing. This is reflected in architectures like GENIVI, the Intel E6x5c architecture, etc. Key is an efficient communicating link between the CPU and the circuitry inside the FPGA. Today's high speed serial interconnect can deliver high bandwidth. But latency requirements must also be met. We present metrics and experimental results to guide embedded system engineers during system architecture design and implementation. Our experiments were performed on exemplary embedded platforms combining Intel Atom CPU with modern FPGA.

1 Introduction

Configurable Systems provide advantageous architecture solutions to meet cost and performance requirements in certain embedded applications. Whenever a rich software stack has to be run in combination with a variety of standard and non-standard IO, for example in so-called Cyber-Physical Systems [Lee08], a powerfull CPU must be augmented by programmable logic (FIG. 1).

Field-Programmable Gate-Arrays (FPGAs) are the foundation technology behind these Configurable Systems - they offer additional degrees of freedom for cost and performance optimizations, the flexibility to perform changes throughout the product's life cycle and can protect from device obsolescence. The ability to connect FPGAs as companion chips to microprocessors basically opens the world of building your own Application Specific Standard Processor (ASSP) to embedded systems designers. These Configurable Systems offer many advantages:

First, they provide flexibility to implement that particular special-purpose I/O connectivity that a microprocessor and/or micro-controller may not have. For example, certain automotive and/or industrial I/O such as CAN, FlexRay, MOST, SerCos, Profibus, Ethercat, etc are typically not provided by many general purpose processors but can easily be implemented in an FPGA companion chip - an aspect which is illustrated by the GENIVI Alliance's computing platform or the Intel Industrial Control Reference Design [Int09]. Using Sigma-Delta converters, FPGAs can even be used to provide integrated analog-to-digital or digital-to-analog connectivity [SRZ10] to an embedded system.

Second, the FPGA's compute fabric offers powerful means for parallel processing [Sea10] and digital signal processing, as it is, for example, required by video image stream processing in automotive driver assist systems and in machine visioning applications. Also, parallel processing in the FPGA fabric sometimes is more advantageous for implementing real-time behavior compared to software running in a Real-time Operating System (RTOS) [SS11].

This makes system implementation more efficient and allows to scale a platform's compute performance with the application's needs.

However, to meet the cost/performance target it is important to pick the right partitioning between what's processed in software in the CPU and what goes into custom co-processing inside the programmable hardware. Key to such Asymmetric Multi-Processing (AMP) architectures is an efficient link between the CPU and the FPGA, and in particular, the bandwidth and latency that this link supports. The choices for this link depend on the CPU device: PowerPC has the Processor Local Bus (PLB) or the Auxiliary Processing Unit (APU), Texas Instrument's OMAP has the Chip-to-Chip (C2C), ARM offers the Advanced eXtensible Interface (AXI), and Intel comes with Peripheral Component Interconnect Express (PCIe).

Intel Architecture (IA) comes in a wide variety of device offerings, ranging from low-power ATOM CPUs up to high-performance Core i7. A significant advantage is the broad software, operating system and development toolchain ecosystem.

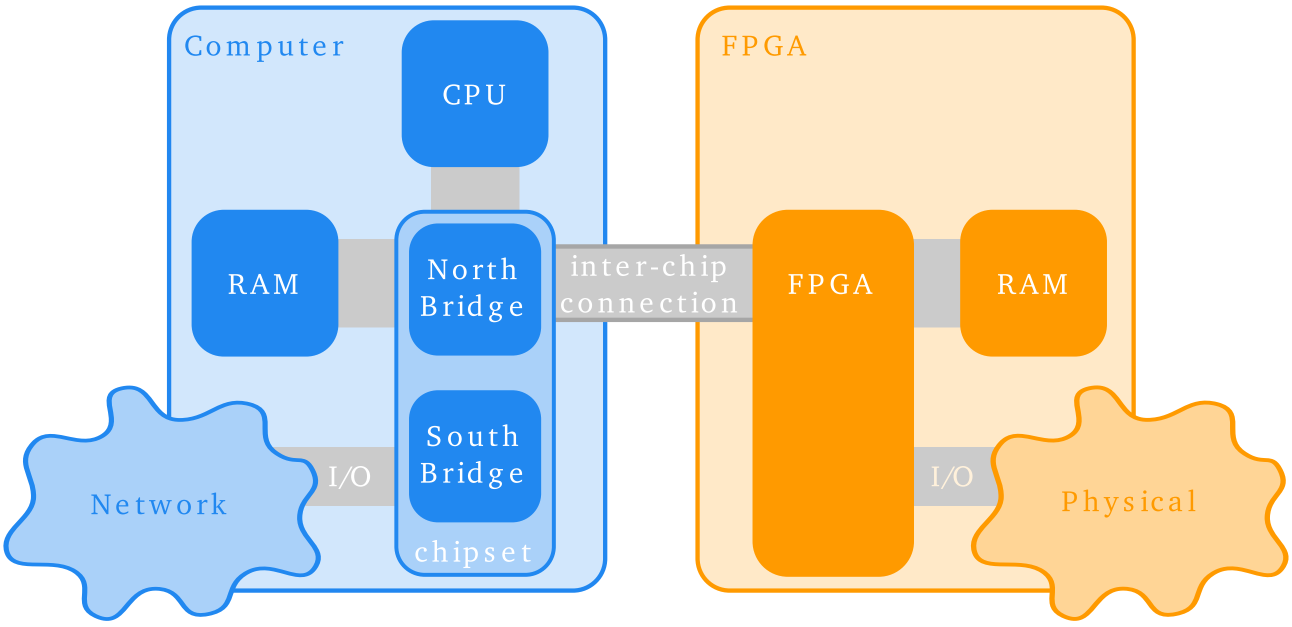


Figure 1: Configurable Systems Architecture

PCIe was introduced in 2004 by the Intel Corporation to replace the older PCI and AGP standards. The theoretical bandwidth of a single-lane PCIe connect in version 1.1 is 250 MB/sec. Today, it is not only ubiquitous in personal computers but it is also an important choice for compute intensive embedded systems.

While in the "old days" of PCI a special PCI chipset was needed for connectivity, most of today's FPGA devices have dedicated, built-in hard macros for PCIe connectivity. Some FPGAs even have dedicated PCIe endpoints such as the Altera Arria II GX, Arria V, and Stratix V devices or the Xilinx Virtex-5, Virtex-6, and Virtex-7 devices. As a result, the design challenges have moved from hardware printed circuit board design to finding the proper system architecture in the FPGA and in developing the software infrastructure for PCIe connectivity between the microprocessor and the FPGA companion.

This technical paper discusses results of a quantitative analysis of different AMP architectures when a high-speed serialized PCIe is used. The data points presented are a result of our joint research with the Institute of Microelectronics at the University of Ulm, Germany [Rot10, San10] and shall give guidance for the "everyday" embedded system designer when he builds his own Configurable System.

2 Technical Background

The use of companion FPGAs quickly leads to loosely-coupled or tightly-coupled AMP, notwithstanding, that the CPU device itself may exhibit a multi-core Symmetric Multi-Processing (SMP) architecture with heterogeneous instances of distributed memory blocks. Sometimes, depending on the application and the underlying hardware/software partitioning, these memory blocks can be used as local memory, solely for use by one single CPU.

Sometimes, it is required that two or more CPUs exchange data via shared memory, which leads to so-called Non-Uniform Memory Architectures (NUMA). The communication path between these CPUs and the memory blocks, and the bandwidth and latency involved, greatly influence the overall systems behavior. Memory bandwidth is the rate at which data can be read from or stored into a semiconductor memory by a CPU. It is usually expressed in units of bytes per second. Memory latency is a measure of time delay experienced between requesting a memory read, or store, and the moment the data is actually available, or stored in memory.

The growing gap between the performance of a CPU and the performance of memory has been a limiting factor on processing performance, especially in embedded systems. When using FPGA technology this gap even more limits the overall system performance, as the memory controllers typically are implemented in programmable logic and optimized for resource cost instead of performance.

Our quantitative analysis shows that these metrics, especially for remote access to memory via PCIe, may limit a Configurable System's performance.

PCIe is a bit serial, asynchronous, packet-oriented, high-speed data link with low voltage differential signals. In version v1.1 one single lane permits to send and receive data simultaneously with a bit rate of $b_{phy} = 2.5$Gbit/s. In version v2.0, the de-facto standard for embedded systems today, this bit rate is $b_{phy} = 5.0$Gbit/s. In order to increase the bandwidth, multiple PCIe lanes can be bundled in the power of two. Since every byte is recoded in 10 bits for transmit (8bit/10bit) $m_{encoding} = 8/10$, the overall symbol rate is $b_{raw} = m_{encoding} \cdot b_{phy} = 250$MB/s for PCIe v1.1 or 500MB/s for PCIe 2.0, resp. The data is transmitted via a layered protocol within so called transaction layer packets (TLP). Thus, the overhead for control data will reduce the total bandwidth compared to the theoretically possible symbol rate.

The overall bandwidth is heavily influenced by the used payload size. The PCIe specification allows a maximum payload size (MPS) of up to $l_{payload} = 4096$B. For example, with a maximum payload size of $l_{payload} = 128$B and a header size of $l_{header} = 20$B the packet efficiency is reduced to $\eta_{packet} = l_{payload}/(l_{payload} + l_{header}) = 0.86$. With a bit transfer rate of $b_{phy} = 2.5$Gbit/s and 8bit/10bit encoding ($m_{encoding}$) bandwidth is reduced to

$$b_{max} \quad = \quad b_{phy} \cdot m_{encoding} \cdot \eta_{packet} = 215\text{MB}/\text{s}.$$

Another crucial point is the latency of the transmissions because it affects the entire system substantially. In particular, this is very important for the responsiveness of real-time applications in closed loop scenarios. Due to the complex interaction of multiple components (like memory controller or bus/link bridges) the overall latency cannot be easily calculated by a formula. To calculate the round trip time / latency, we have to take into account the time needed to transfer a single package.

The time to transfer a single TLP with a payload of 128B via a PCIe 1.1 is given by:

$$t_{TLP} \quad = \quad (l_{payload} + l_{header}) \cdot \frac{m_{encoding}}{b_{phy} \cdot n_{lanes}}$$

Under the stated conditions, for a single lane, the t_{TLP} is 592ns. The latency to transfer a single TLP improves when spread across all available PCIe lanes. Bundled to 16 PCIe 1.1 lanes, the mentioned time decreases to 37ns. Additionally the delays introduced by all intermediate switches have to be taken into account. Examinations have shown that the latency of one switch usually ranges from 110ns to 150ns [Reg07].

3 Architecture Choices for HW/SW Co-Processing

Multiple choices exist for implementing Configurable Systems in FPGAs: First, the embedded CPU can be a hard CPU core highly optimized for area and performance, for example like the PowerPC 440 inside the Xilinx Virtex-5FXT devices. Or, it can be a soft CPU core, for example like the Xilinx MicroBlaze [Xil09b] or the Altera NIOS-II, both of which are implemented by programmable gates and as such can go into almost any device of the corresponding FPGA vendor. Obviously, a soft CPU core lacks performance over a hard CPU core.

Memory can either be implemented as external SRAM or external DRAM. Or, it can be implemented using device-internal On-Chip-Memory (Altera speak) / BRAM (Xilinx speak). Depending on the FPGA device vendor choices exist for the Memory Controller Unit (MCU), as well, and these choices are important for delivering a performance system: Because of the direct impact of the MCU on the overall system's performance, it sometimes is justified to build a specialized, optimized MCU in-house, or purchase one as an IP core from a 3rd party.

FIG. 2 shows the exemplary components we have used for our quantitative analysis: A mini-ITX board from Point-of-View POV/ION and the Xilinx reference platform ML507, connected via a single PCIe v1.1 lane. Several FPGA design variants have been implemented using the embedded FPGA CPU choices, memory choices and MCU choices available.

3.1 Exemplary Hardware Components

The POV/ION board features an Intel Atom 330 processor running at 1.6 GHz, an NVIDIA MCP79 chipset, 4GiB DDR2 RAM (PC2-5300) and one PCIe x16 1.1 slot. The Xilinx ML507 development kit features a Xilinx Virtex-5FX70T FPGA and PCIe v1.1 single lane [Xil07]. This FPGA device has a PowerPC 440 as a hard CPU core plus soft CPU core MicroBlaze [Xil09a, Xil09b].

Three FPGA design variants have been implemented. The first variant (PowerPC+MC) features a PowerPC 440 running at a clock frequency of 400MHz and a simple memory controller (MC) which connects DDR2 main memory to the PowerPC's builtin memory controller interface (MCI). The second variant (PowerPC+MPMC) is also based on PowerPC 440, but the DDR2 main memory is connected via the vendor-provided multi-port memory controller (MPMC). The MPMC connects up to eight independent interfaces and buses to a single memory. Both PowerPC 440 design variants have a 128 bit wide PLB bus. The third variant (MicroBlaze) features the Xilinx MicroBlaze 7.20d which is configured with a five-stage pipeline and a vendor-provided Memory Management Unit (MMU). The MicroBlaze's cache is made configurable in size up to 64 KiB. The PLB interface of the MicroBlaze is limited to a data width of 32 bit.

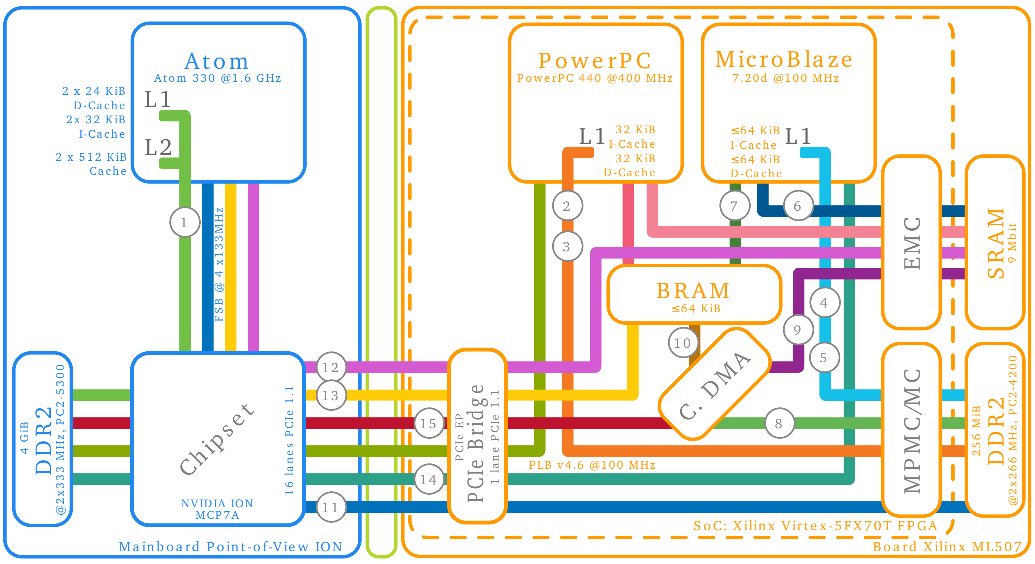


Figure 2: Configurable System Building Blocks and Connectivity

Additionally to the memory controllers for the DDR2 main memory, all three design variants feature an external memory controller for on-board SRAM and several BRAM controllers. All design variants have a PLB-to-PCIe bridge and a vendor-provided Central DMA engine. Both PLB interfaces to the CDMA controller, PLB master and PLB slave, are connected to the same PLB bus, where the PCIe bridge is connected also. Thus the CDMA controller can transfer data across the PLB-to-PCIe bridge in both directions and can be used either by the FPGA or the ATOM CPU.

3.2 Exemplary Software Components

The performance analysis setups are based on the Linux operating system for each of the three processors, ATOM, PowerPC 440, and MicroBlaze. While the processors inside the FPGA run a lightweight Linux V2.1 from Missing Link Electronics, the ATOM CPU runs a fully featured Ubuntu 8.04LTS distribution.

Since user space applications run in virtual memory mode, they cannot access other memory regions. Therefore, several kernel modules were developed to map certain memory locations to the user space. This also includes the distant memory and the CDMA engine which both are connected via PCIe. The upper part of the MicroBlaze's DDR2 main memory is cut-off in order to establish shared, fast and unrestricted data exchange between other processors. This memory region is used for DMA access because it is not restricted to Linux kernel page alignment.

Similar to the MicroBlaze's Linux, a kernel module was developed for the ATOM CPU in order to map the PCIe memory regions and allocate the DMA memory. The DMA memory address is written to the PLB-to-PCIe bridge so as to enable the CPUs inside the FPGA to write into the main memory of the ATOM CPU, directly. In the kernel memory management the maximum DMA buffer size is restricted to 2,048 pages at 4 KiB each. As interrupts are not available in user space, the successful completion of a DMA transfer gets notified by polling a control register. In order not to block the bus transfers unnecessarily the polling is only done in the last fraction of each transfer.

4 Quantitative Analysis Setup

The Open Source LMbench tool suite was used for latency and bandwidth measures. LMBench provides simple, portable benchmarks in order to compare POSIX compliant Linux systems. Every benchmark test is run 100 times for each configuration to increase accuracy and to get better statistics like variation, confidence intervals or quantiles. Jitters in repeating single measurement results are discussed in terms of interquartile range (IQR) and minima and maxima.

To measure the latency of a memory read, the tool *lat_mem_rd* of said LMbench tool suite was used. Operating on data only, instruction loads cannot be measured. The latency measurements use all caches and all main memories. To obtain the size of the cache by means of latency, the size of the working set is varied. Since the memory hierarchy prevents the detection of cache write-backs, there is no way to measure the latency of memory writes. The bandwidth is also measured with said LMbench tool *bw_mem*; it is run in different modes to obtain measurements for read, write, copy or other combined bandwidths.

To simulate high system load the tool *stress* was built and used. This tool continuously allocates memory and thus causes high load in the memory system. It is run on each CPU which performs the transfer and is configured such that one eighth of the total main memory is used only. This serves to mimic a more realistic setting.

The exemplary Configurable System was analyzed at several levels. Our experiments start from the inside of the system involving only few components. Then the tests were widened consecutively in order to cover the entire system performance. Each test analyzes certain aspects using certain distinct components. By comparing these tests, conclusions can be drawn regarding the system components involved. Thus it is possible to obtain metrics for certain components that can not be measured directly.

The *Inner-Processor-System* tests reflect the performance of a stand-alone processor system. It investigates how each processor accesses its memory, and correlates to Paths 1-4 in FIG. 2. Tests regarding cache line size and prefetching were performed first. After that, the focus was directed to the memory hierarchy performance covering latency and bandwidth, additionally measured under high memory loads.

Memory accesses were cached to increase the performance. Operating on allocated memory which was obtained from an operation system allowed us to additionally measure each cache's performance. This gives an insight into how well the different memory performance levels are covered by the memory hierarchy. These measurements reflect the way a stand-alone processor performs. For industrial control systems, for example, it is of major interest to know both how long it takes to read from the cache, and the cost of a cache miss. The overall system performance is heavily influenced by the complex memory hierarchy and its parameters like the size of a cache line and the entire cache size.

The *Inner-Board-System* tests covered NUMA aspects. We investigated the way how shared memory or a memory of an I/O controller is accessed with regards to latency and bandwidth. This correlates to Paths 5-7 in FIG. 2. Caching schemes are by-passed when accesses to the memory locations are not part of the main memory. To obtain similar results as with the native memory, the memory regions were mapped into user space. This scenario is similar to a supervisor processor in an FPGA which communicates with real-time subsystems via mailboxes. The tests were performed once again using the CDMA controller because of the special interest for data acquisition and transfers of large blocks of data.

The *Cross-Board-System* tests covered the way how distant memory is accessed when connected through a high-speed serial link, which in our case was PCIe. The test correlates to the Paths 11-14 in FIG. 2. The data flow crosses from one processing domain to the other. The requests on the synchronous parallel bus (PLB) have to be translated to requests on the asynchronous serial link (PCIe). This scenario is closely related to the communication between a software partition and a hardware partition of a heterogeneous AMP architecture. The performance of the PCIe link defines the behavior of the overall system, such that with a high latency and a low bandwidth the system behaves like a loosely-coupled system. This test aimed to check whether a closed loop system can be built including the communication delay of such a serial link. Because the PLB-to-PCIe bridge does address translations from one address space to the other, the mechanism of memory access is the same as for the local memory. Thus, the same benchmarks were used. Analog to the previously described local memory test, this test was performed once again using the CDMA controller.

5 Quantitative Analysis Results

Three different CPUs and with multiple, different memory architectures were analyzed. While the results are exemplary, they can serve as a guideline when building a Configurable System.

The most important aspect is the memory subsystem. Here the ATOM CPU clearly outperforms the FPGA-based CPUs, due to the much higher effective bus clock rate of the ATOM CPU compared to the PowerPC 440 and the MicroBlaze, despite the fact that the measured memory bandwidth of the ATOM CPU only reaches two thirds of the theoretical bandwidth. Furthermore, the memory bandwidth of the PowerPC 440, the MicroBlaze and the CDMA seem to be limited by the Processor Local Bus (PLB) and the FPGA-based memory controllers.

To mitigate such bottlenecks, usually an efficient caching strategy and cache hierarchy can be implemented. In both aspects, the ATOM CPU shows higher performance compared to the smaller embedded cores. Thus, the L1 cache of the PowerPC 440 and the MicroBlaze is, in terms of latency, somewhat comparable to the slower L2 cache of the ATOM CPU. In addition, the very effective prefetching strategy, that speculatively loads other cache lines, is only implemented by the ATOM CPU but not by the FPGA-based CPUs. The latter have been optimized for embedded applications in preference of energy efficiency and heat consumption, over computational power.

Other memory subsystem related concerns are the inner-board transfers and the remote inter-board transfers. The performance of local requests within the FPGA remain on a synchronous bus and are reasonably good. Thus, bus component implementations inside the FPGA companion are fast, efficient and configurable in bandwidth concerns. This allows to decide between resource usage, power consumption, clock rate and bandwidth. However, inter-board transfers, in our case via PCIe, imposes latency and, therefore, the overall system's performance drops, independent from which side initiates the transfers and the used hardware. This is an interesting observation and not so obvious. For example, compared to the local DDR2 RAM access, the latency of mapped memory crossing the PCIe v1.1 lane for the MicroBlaze is about 7 times as high. The situation for the ATOM CPU to access the memories of the FPGA board, compared to its native memory is even worse with a 18x latency increase.

To counteract this problem, it may not be sufficient to combine multiple PCIe lanes because this improves the bandwidth but does not reduce the latency sufficiently. Furthermore, the bandwidth performance for PCIe reads is also low due to the PCIe protocol itself. To investigate the reason for the latency and bandwidth decrease, a consumer PCIe graphics card was attached to the ATOM CPU. With this configuration, the latency to access the GDDR2 memory of the graphics card over a single PCIe lane was measured to be around 1000 ns. The ATOM CPU accessing the FPGA's memory was measured near by 2000 ns. Compared to the graphics card the latency is twice as high. Since the memories are similar, the latency increase by approximately 1000 ns could only be introduced by the PCIe controller, the PLB-to-PCIe bridge and additional PLB bus transfers inside the FPGA. Since the PCIe controller on the FPGA is a dedicated standard controller and the local PLB bus transfers are sufficiently fast, only the PLB-to-PCIe bridge can be accountable for the latency increase. Bandwidth concerns were also validated: With the reduction of the payload size for PCIe transfers introduced by negotiations, the packet efficiency decreases and the bandwidth further drops. Thus accessing PCIe memory regions in a word-wise manner, the packet efficiency drops to 17 percent and far more packets have to be transfered, each additionally causing traffic. In all tests, PCIe mechanisms to increase the bandwidth e.g. packet queuing were not utilized.

Further investigations on this bridge have shown that the PCIe protocol translation is determining the behavior. To avoid low bandwidth performance of read requests, PCIe writes should be preferred, whenever possible. Like local DMA transfers above, in some situations a caching or bundling strategy for the PCIe mapped memory could be applied within the PLB-to-PCIe bridge or the operating system.

The CDMA controller on the FPGA exhibited interesting behavior during inter-board communication: Read, write or copy operations performed by a local soft-core or hard-core CPU are driven word-wise. Thus, several independent bus requests are required in order to transfer several words in a row. The CDMA bundles these requests and performs them in parallel as a block which additionally relieves the CPU. This is reflected in high transfer rates for copy operations. Unfortunately, in our scenario the CDMA requires to access the bus twice, first to read the memory content from BRAM etc. into its own FIFO buffer and, secondly, to commit this data to the PCIe bridge registers for the inter-board transfer. Therefore this access pattern may completely block the memory subsystem by the CDMA controller.

This behavior is problematic because the PLB is then blocked. Thus, all PLB participants will not be able to communicate. Furthermore, the CDMA controller concurrently accesses the PLB for reads and writes in burst mode for a large amount of clock cycles. However, the overall PCIe performance is far from reaching the theoretical bandwidth of 2.5 Gbit/s. If the CDMA would be integrated into the PCIe bridge the efficiency of DMA transfers likely improve.

References

[Int09] Intel® Industrial Control Reference Design. Technical report, Intel®, Gleichmann Electronics Research, 2009.

[Lee08] Edward A. Lee. Cyber Physical Systems: Design Challenges. *International Symposium on Object/Component/Service-Oriented Real-Time - ISORC 2008*, 2008.

[Reg07] Jack Regula. Overcoming Latency in PCIe Systems. *EE Times*, 2007.

[Rot10] Bernd Rottler. Architecture Exploration of microSD Host Controllers on a Programmable System-on-Chip. Master's thesis, Universität Ulm, Ulm, Germany, 2010.

[San10] Leo Santak. Hardware-/Software Platform Architectures for Distributed Industrial Control Systems. Master's thesis, Universität Ulm, Ulm, Germany, 2010.

[Sea10] Endric Schubert and et. al. Building a Better Crypto Engine the Programmable Way. *XCELL Journal*, 72, 2010.

[SRZ10] Endric Schubert, Johannes Röttig, and Axel Zimmermann. Delta-Sigma converters for audio output in an infotainment FPGA. *EETimes europe AUTOMOTIVE*, June 2010.

[SS11] Endric Schubert and Glenn Steiner. Design Choices for Cyber-Physical Systems. *DAC - Knowledge Center*, 2011.

[Xil07] UG349: ML505/ML506/ML507 Reference Design User Guide. Technical report, Xilinx, Inc., March 2007. Visited on November 28th, 2011.

[Xil09a] ds100: Virtex-5 Family Overview. Technical report, Xilinx, Inc., June 2009. Visited on November 28th, 2011.

[Xil09b] UG200: MicroBlaze Processor Reference Guide. Technical report, Xilinx, Inc., June 2009. Visited on November 28th, 2011.

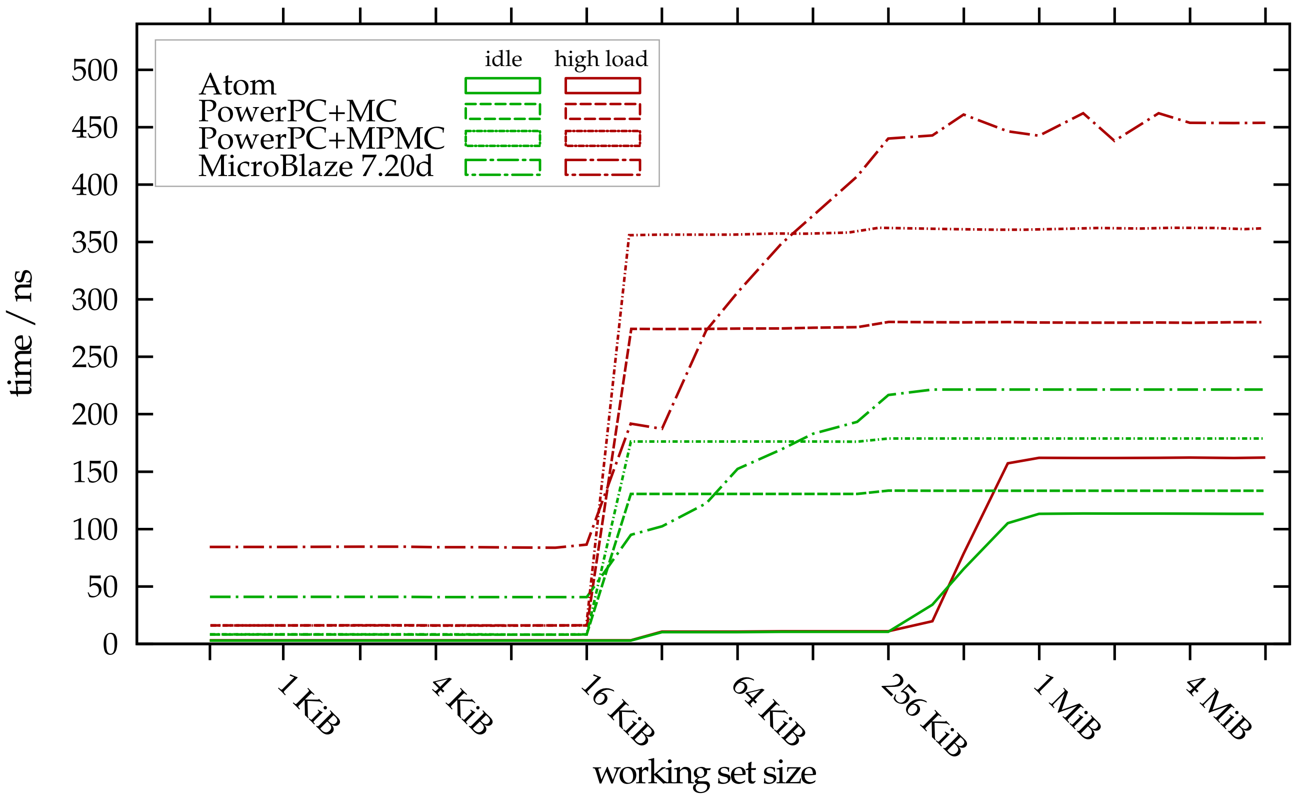


Figure 3: Latency over working set size in idle mode and busy mode, for ATOM 330, PowerPC440+MC, PowerPC+MPMC, and MicroBlaze.

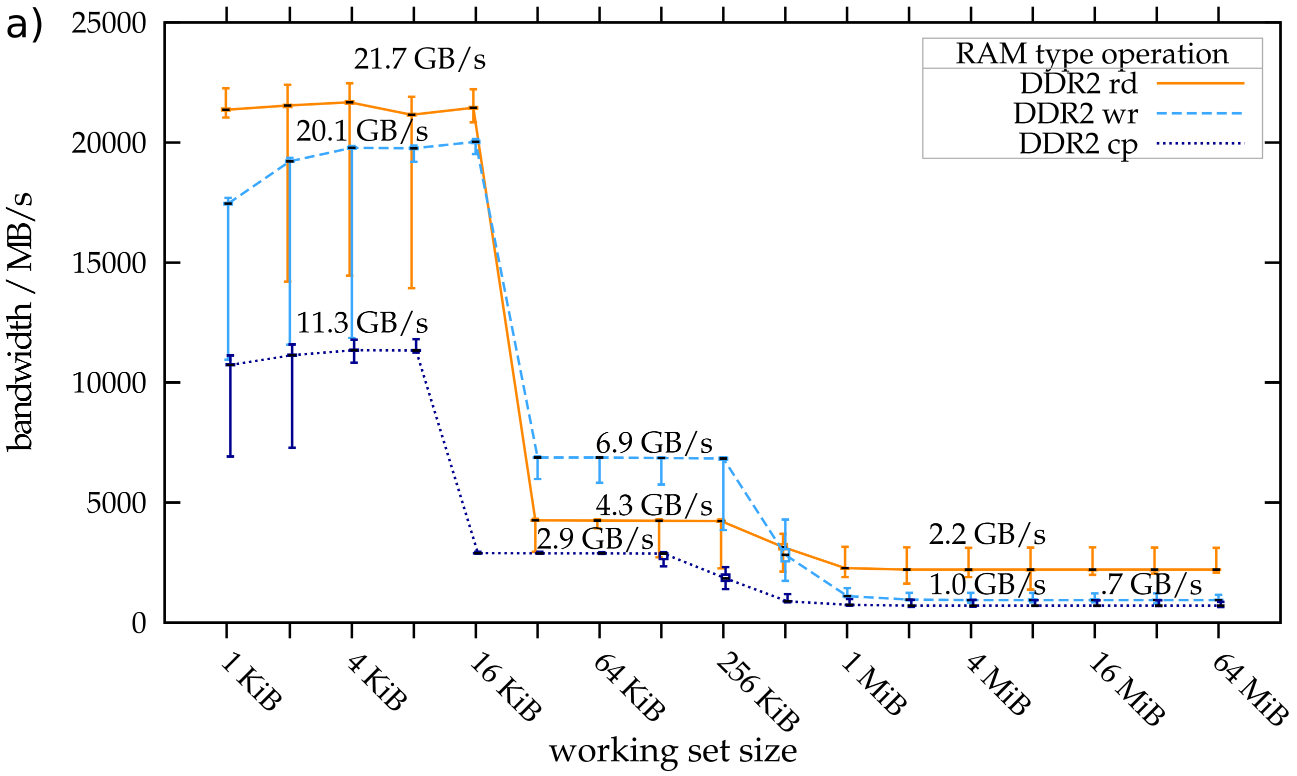


Figure 4: Bandwidth over working set size on native memory in busy mode for ATOM 330.

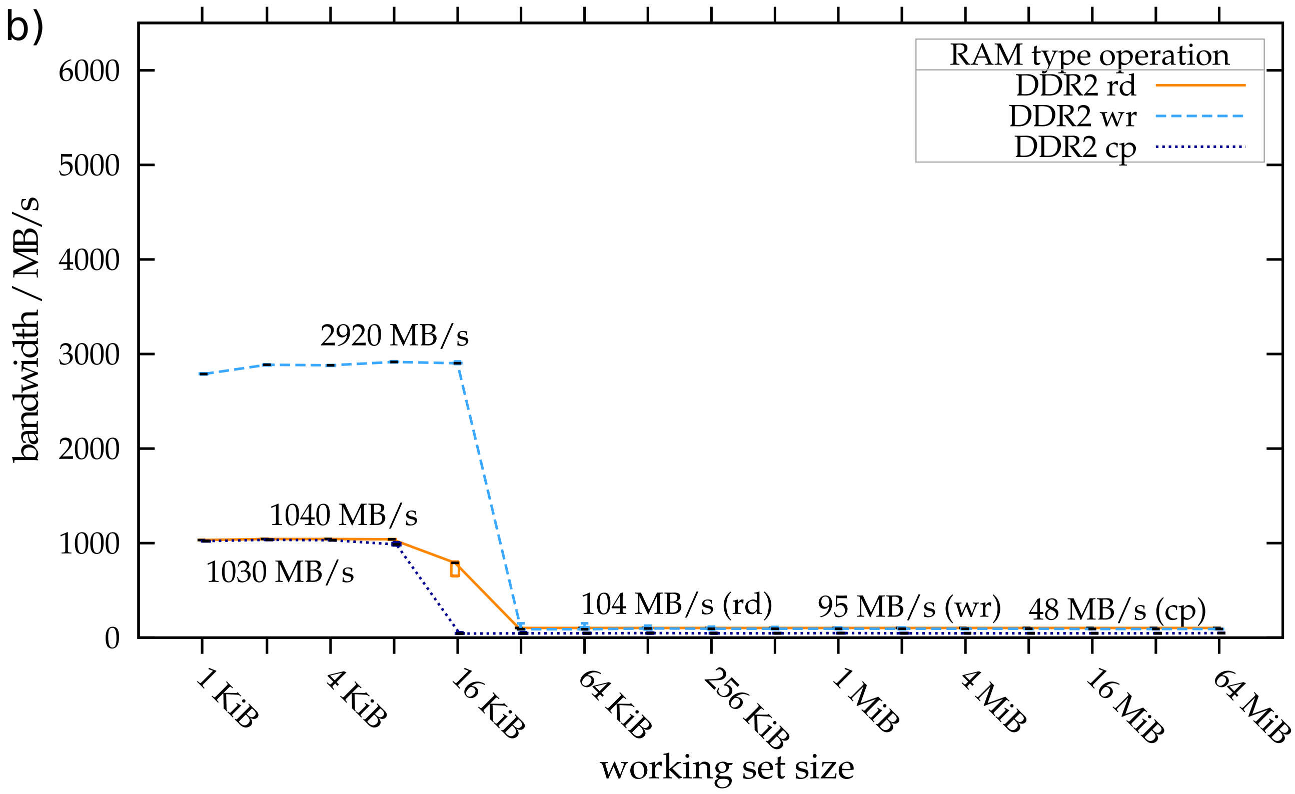


Figure 5: Bandwidth over working set size on native memory in busy mode for PowerPC+MPMC.

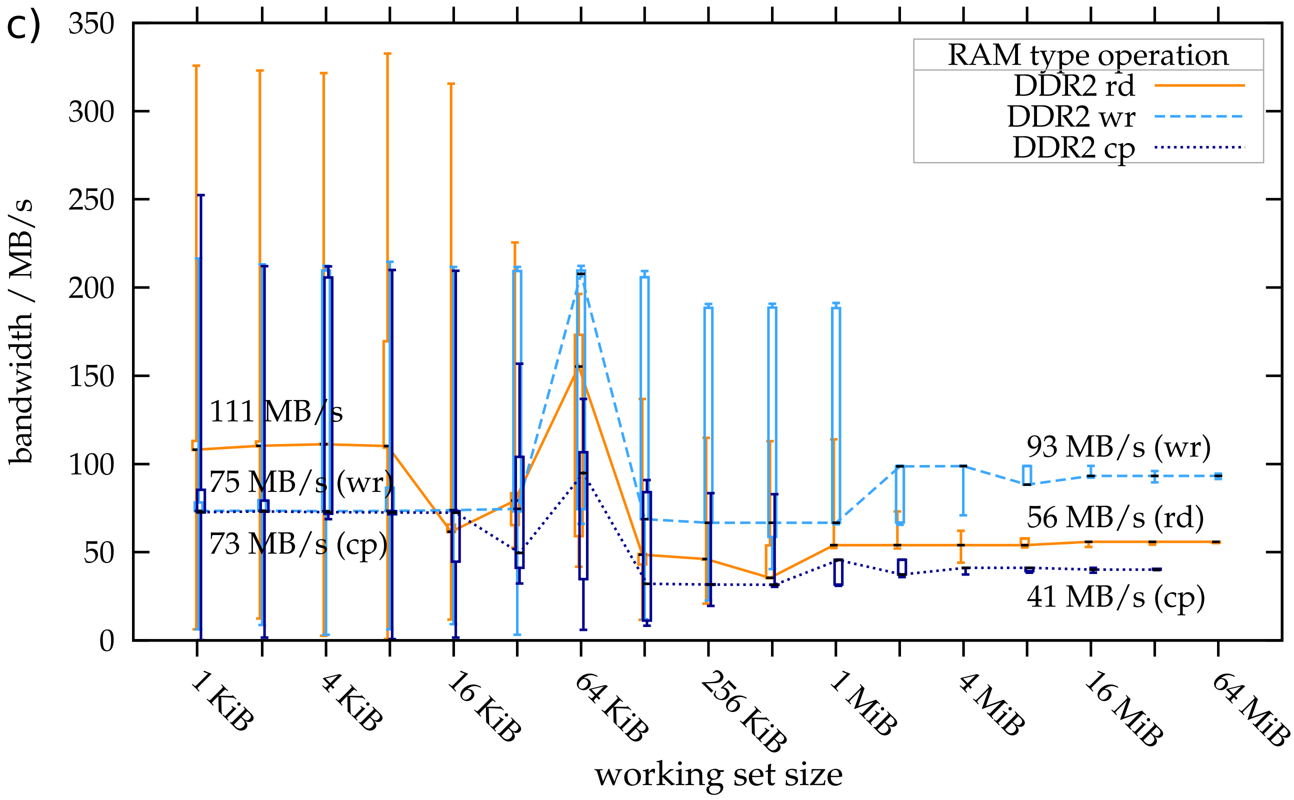


Figure 6: Bandwidth over working set size on native memory in busy mode for MicroBlaze.

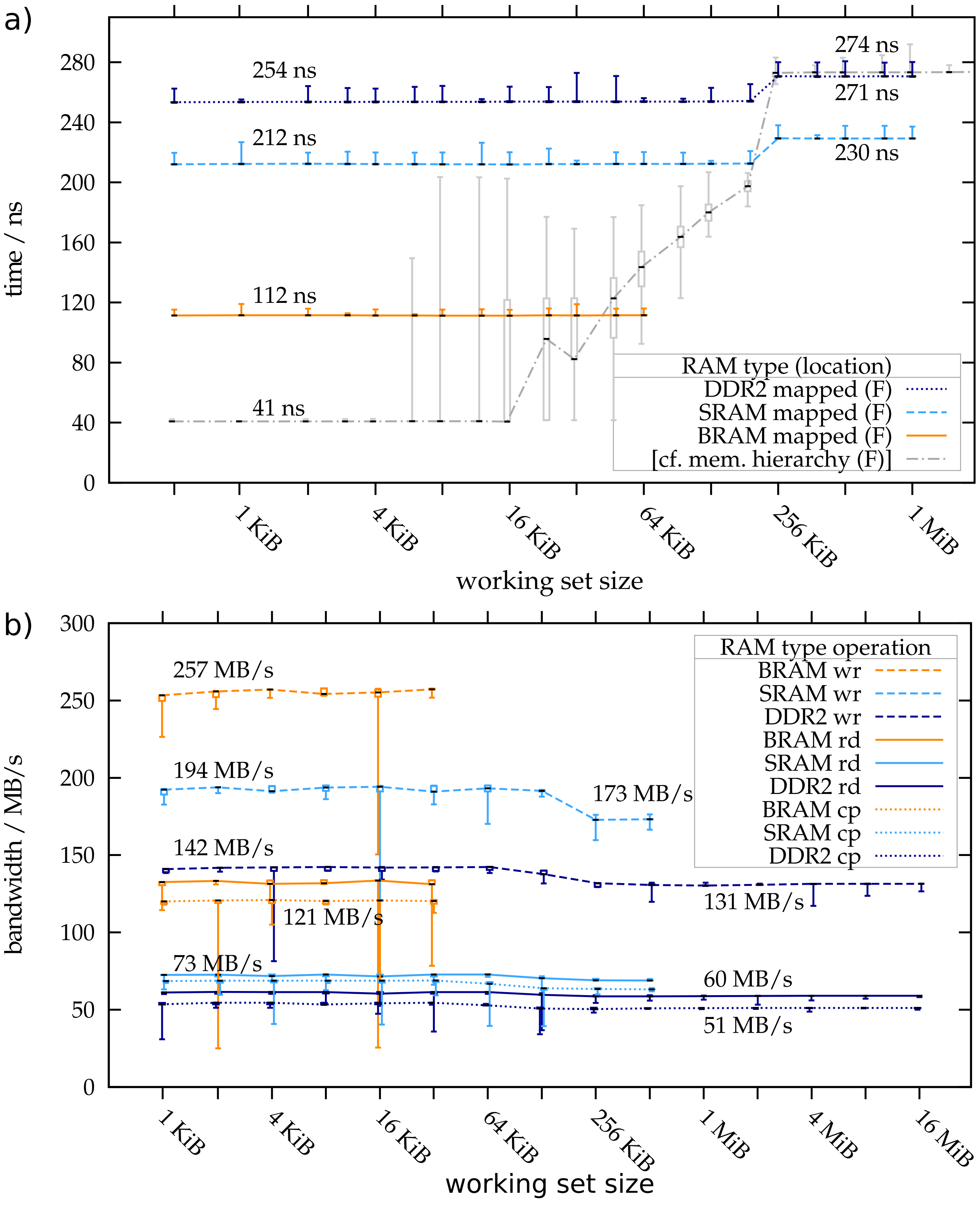


Figure 7: Latency a) and bandwidth b) over working set size for local mapped memory, idle mode, for MicroBlaze.

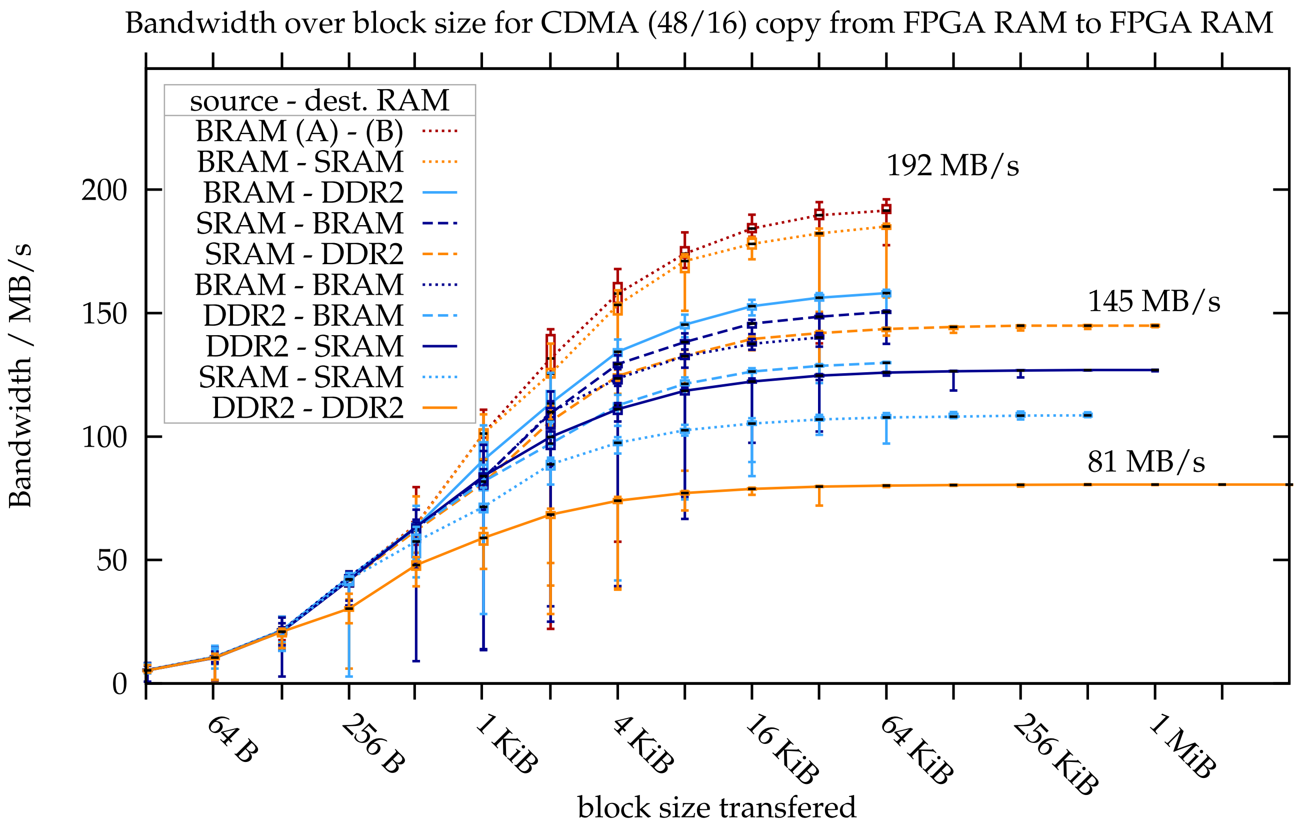


Figure 8: Bandwidth over block size for CDMA (48/16) copy from FPGA RAM to FPGA RAM.

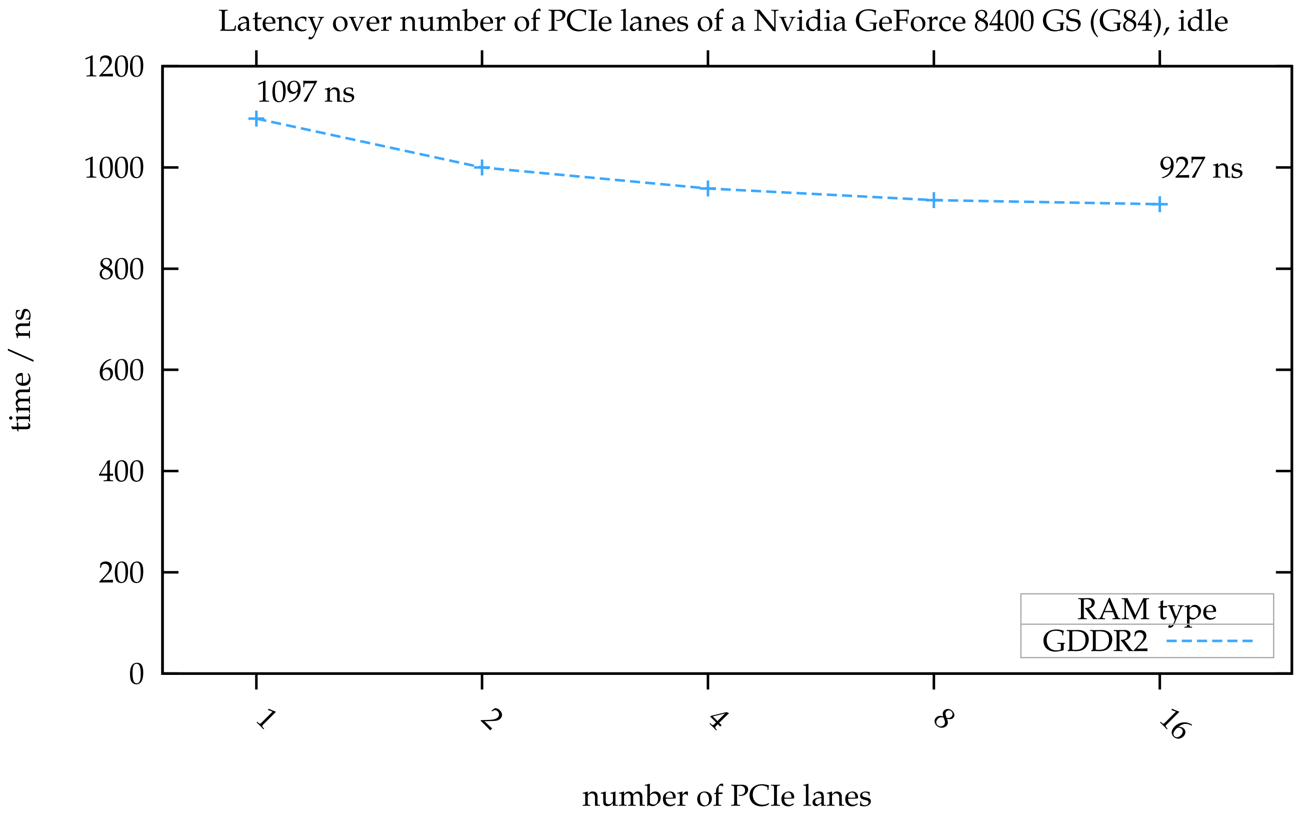
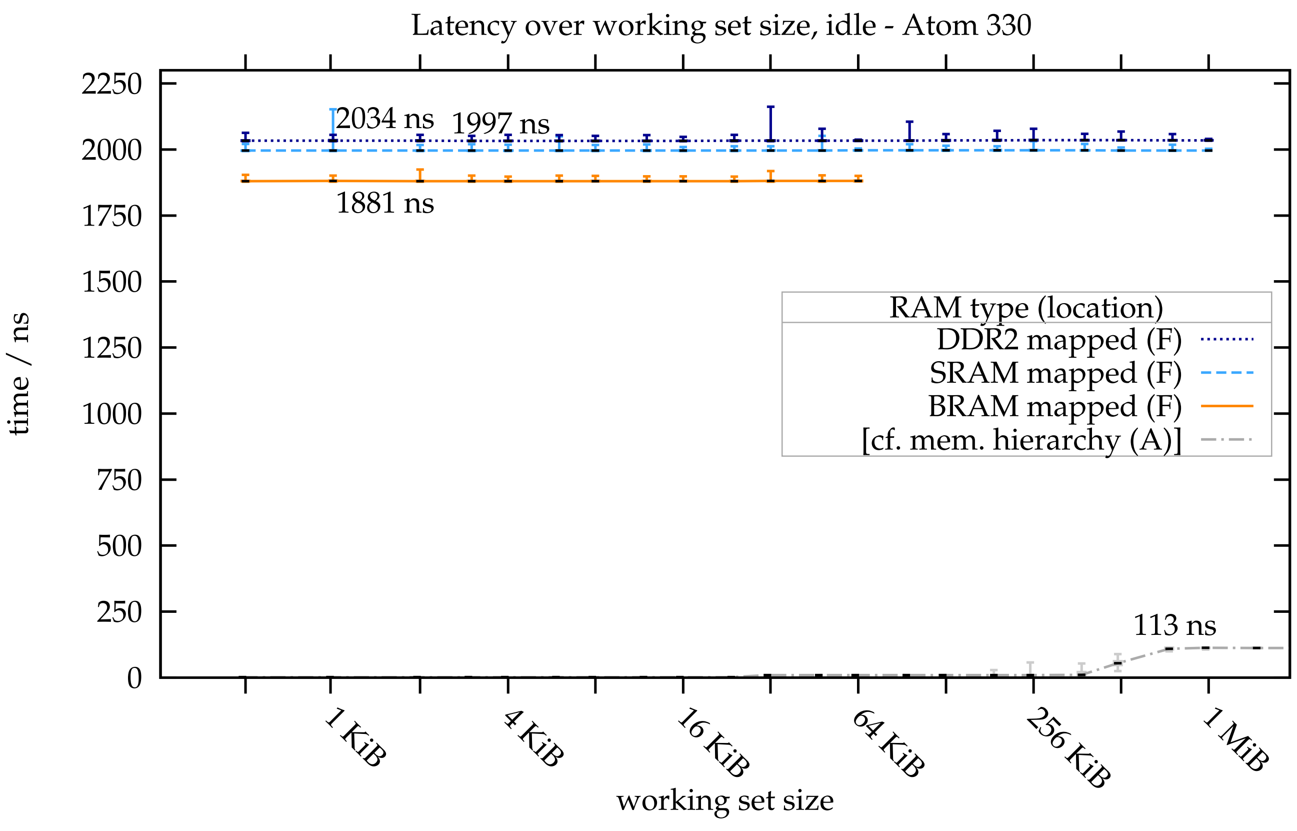


Figure 9: Latency over number of PCIe lanes for mapped VRAM to ATOM CPU, idle mode.



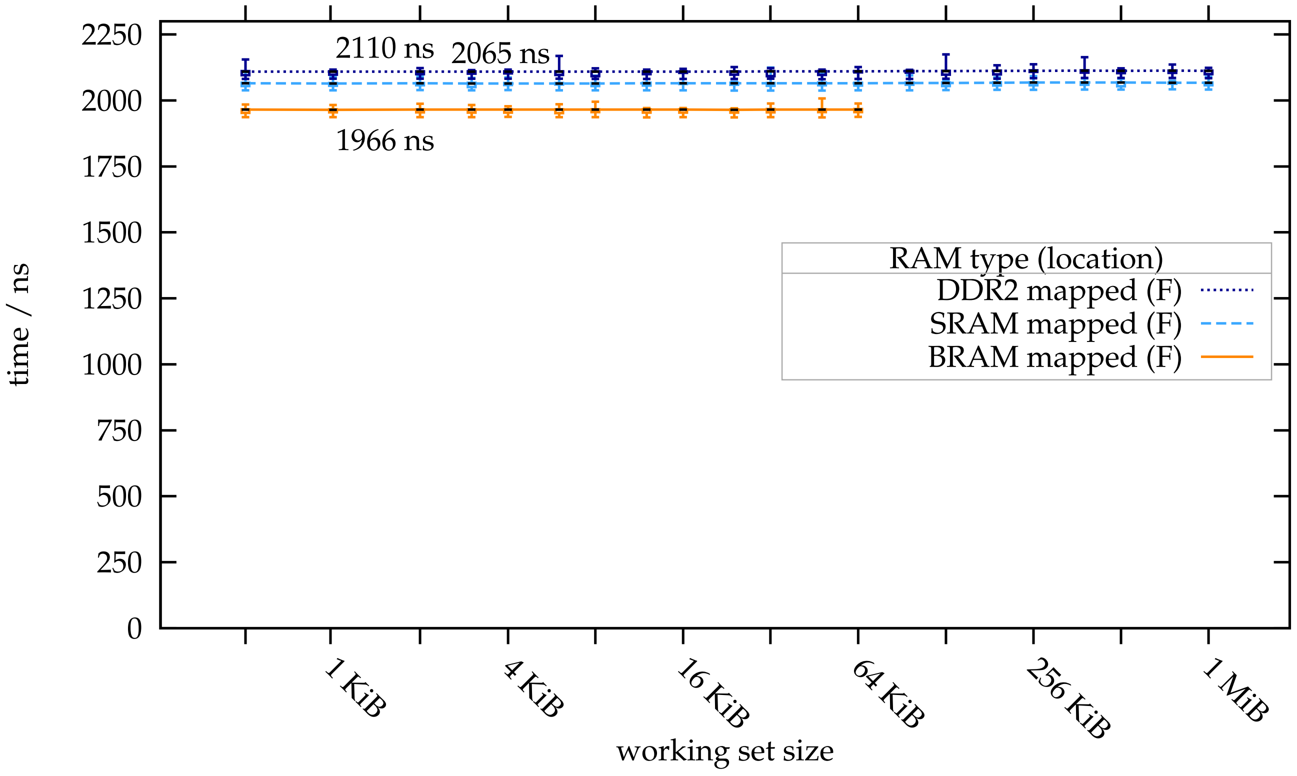


Figure 10: Latency over working set size for PCIe-mapped memory to ATOM CPU, a) idle mode and b) busy mode.

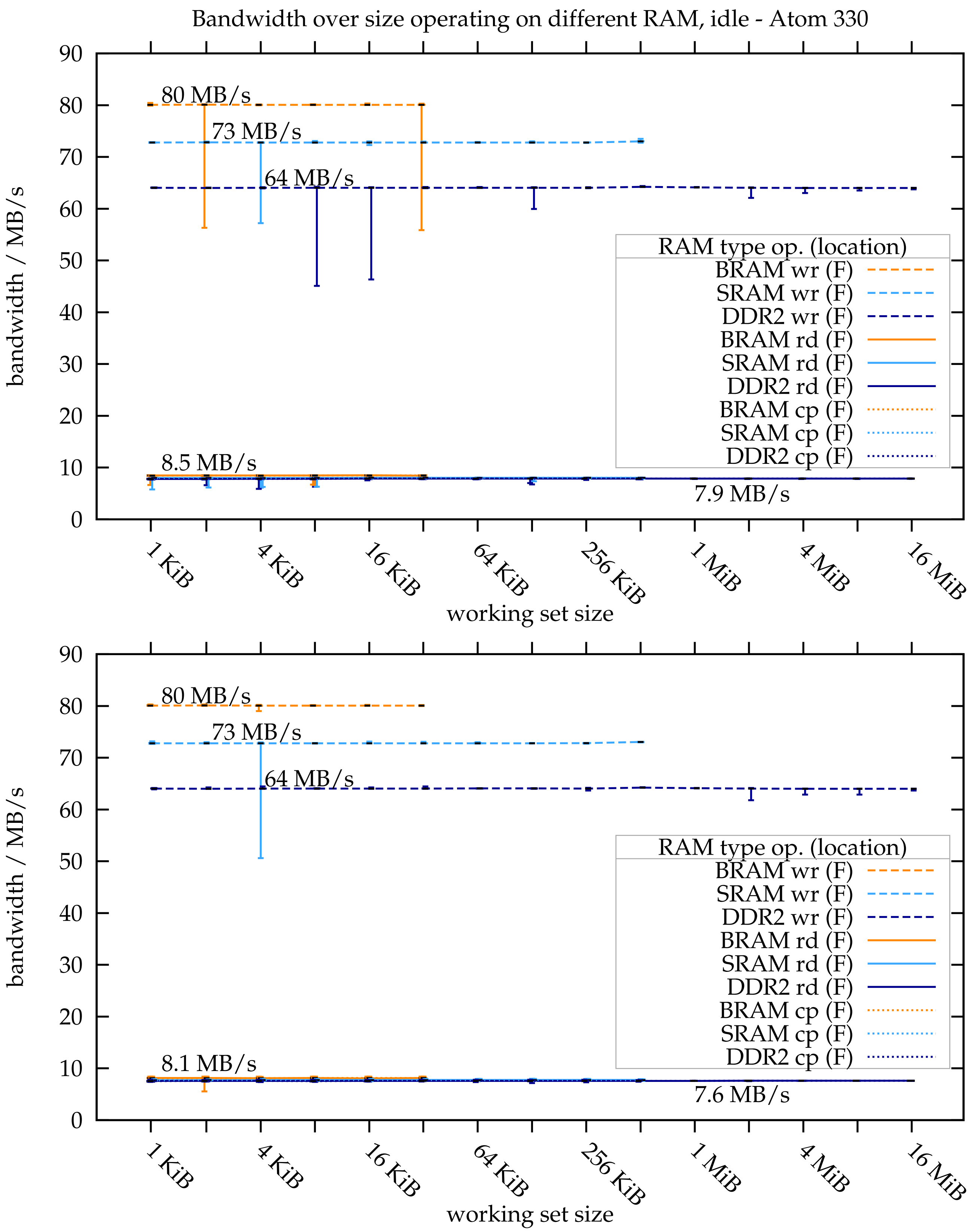


Figure 11: Bandwidth over working set size on distant mapped memory, Atom 330, a) idle mode, b) busy mode.

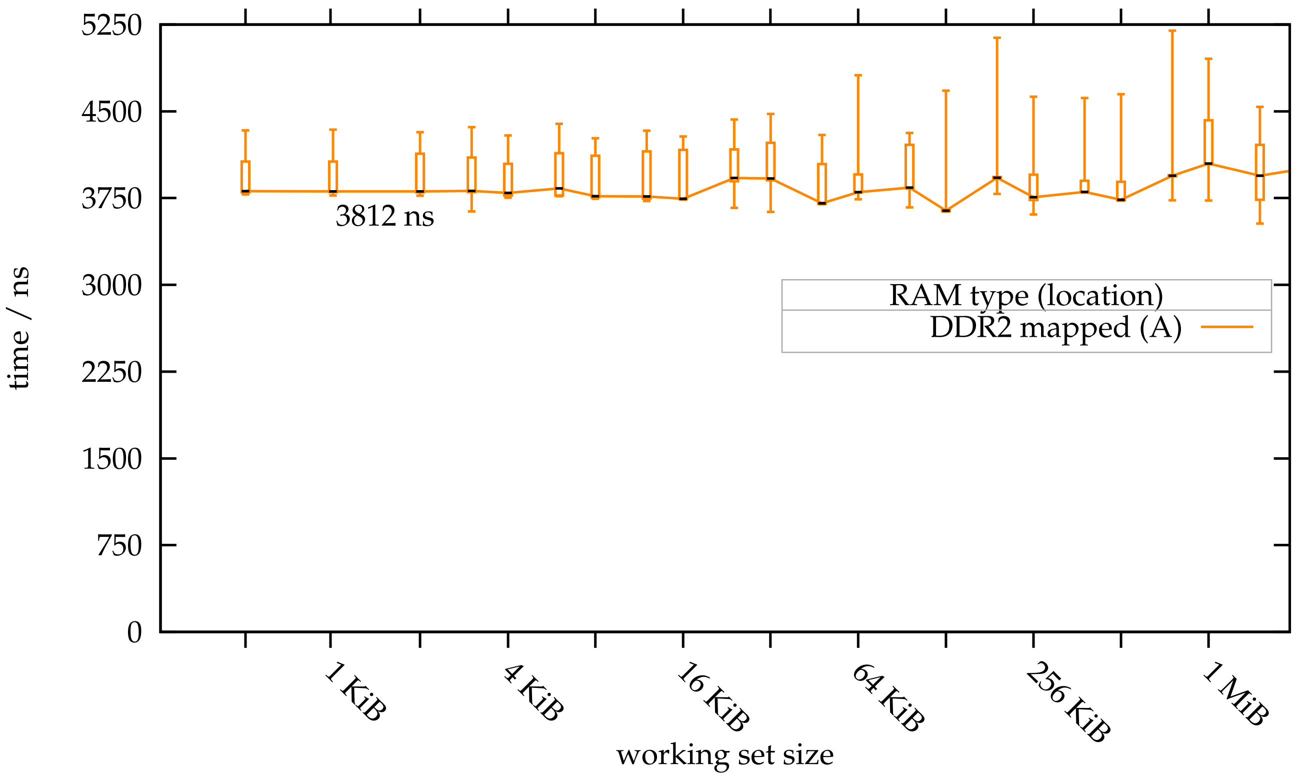


Figure 12: Latency over working set size for PCIe-mapped memory to MicroBlaze, busy mode.

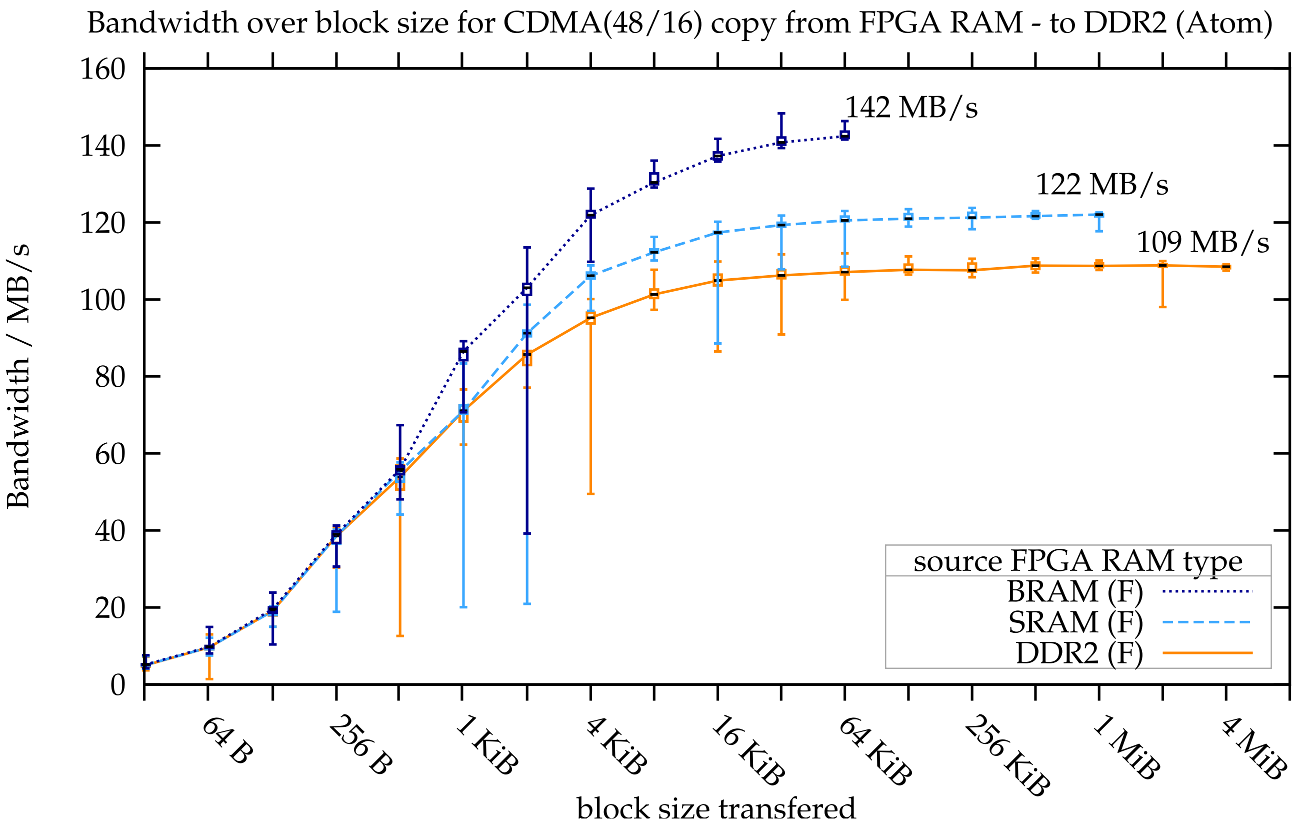


Figure 13: Bandwidth over block size for CDMA (48/16) copy from FPGA RAM to DDR2 RAM, Atom 330.