Application-specific Memory Performance of a Heterogeneous Reconfigurable Architecture

Sean Whitty, Henning Sahlbach, Brady Hurlburt, Rolf Ernst
Institute of Computer and Network Engineering
Technische Universität Braunschweig
38106 Braunschweig, Germany
{whitty | sahlbach | hurlburt | ernst}@ida.ing.tu-bs.de

Wolfram Putzke-Röming
Deutsche Thomson OHG, Germany
30625 Hannover, Germany
wolfram.putzke-roeming@thomson.net

***Abstract*— Heterogeneous reconfigurable processing architectures are often limited by the speed at which they can access data in external memory. Such architectures are designed for flexibility to support a broad range of target applications, including advanced algorithms with significant processing and data requirements. Clearly, strong performance of applications in this category is an extremely relevant metric for demonstrating the full performance potential of heterogeneous computing platforms. One such example, a film grain noise reduction application for high-definition video, which is composed of multiple image processing tasks, requires enormous data rates due to its large input image size and real-time processing constraints. This application is especially representative of highly parallel, heterogeneous, data-intensive programs that can properly exploit the advantages offered by computing platforms with multiple heterogeneous reconfigurable processing elements. To accomplish this task and meet the above requirements, a bandwidth-optimized external memory controller has been designed for use with a heterogeneous reconfigurable architecture and its NoC interconnect.**

With the help of the application described above, this paper evaluates the proposed architecture in two forms: (1) with a basic memory controller IP and (2) with the advanced memory controller design. The results illustrate the full potential of the computing platform as well as the power of heterogeneous reconfigurable computing combined with high-speed access to large external memories.

I. Introduction

Many high-performance computing platforms share a common bottleneck: the relatively slow speed at which they can load and store large amounts of data to and from external memory sources. Heterogeneous reconfigurable architectures such as the Multi-purpose Dynamically Reconfigurable Platform for Intensive Heterogeneous Processing (MORPHEUS) platform are no exception. These heterogeneous architectures are designed with flexibility as a key goal, which implies a broad range of target applications. The MORPHEUS heterogeneous platform was developed with such flexibility in mind in order to support simple tasks with minimal memory needs, as well as applications consisting of more advanced algorithms with significant processing and data requirements. This flexibility supports an efficient use of hardware resources, which is a critical property of embedded systems due to the importance of size, cost, and power consumption limitations.

To satisfy the requirements of both simple control-oriented as well as data-intensive applications, the MORPHEUS platform offers a scalable memory architecture. For best performance, a custom solution was developed in the form of an advanced quality-of-service (QoS) access optimizing memory controller [1], designed to maximize bandwidth while still delivering acceptable latencies. For applications with more moderate memory requirements, the scalability of the MORPHEUS computing architecture can be utilized to easily integrate a more standard memory solution, which offers significantly reduced performance in return for reduced area, complexity, and overall packaging cost.

In this paper, we quantify the performance offered by two specific versions of the platform with the above mentioned memory controllers, rather than focus on the memory controller architectures themselves, which were presented in detail in [1], [2]. To perform this comparison, a real-time image processing application with massive data requirements was selected as a real-world case study to demonstrate the architecture's ability to satisfy realistic memory needs. In showing the massive performance increases offered by the integration of an advanced memory controller, we demonstrate the positive influence of a high-speed memory solution on a heterogeneous reconfigurable architecture combined with high-performance applications.

This paper is organized as follows. Section II introduces the heterogeneous reconfigurable computing platform. Next, Section III outlines the scalable memory subsystem and proposes two memory controller solutions. Section IV illustrates the integration of these solutions into the MORPHEUS architecture. In Section V, a target application domain and its requirements are presented, along with an example application. Then, a thorough performance evaluation and corresponding analysis are presented in VI, followed by a final conclusion.

II. Heterogeneous Reconfigurable Architecture

The MORPHEUS project is a European collaboration (IST027342) that provides a flexible heterogeneous platform for HW/SW co-design via a unique architecture, composed of reconfigurable computing units of varying granularity [3], [4]. This concept allows high computation density common to coarse-grained reconfigurable architectures, optimal hardware structure like that found in many System-on-Chips (SoC), and the flexibility and programmability of General Purpose Processors (GPP), while at the same time attempting to minimize the disadvantages of each platform. Another goal is to provide an integrated toolset to easily map and implement target applications, allowing shorter development times than those typical for Field Programmable Gate Arrays (FPGA).

The architecture is based on an ARM9 processor, responsible for data, control, and configuration transfers between all system resources, and three Heterogeneous Reconfigurable Engines (HRE), each targeting different types of computation:

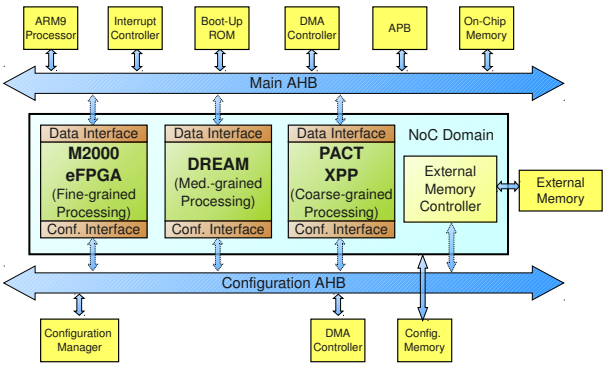

Figure 1. MORPHEUS architecture

- The coarse-grained PACT XPP provides high computation density for stream-based algorithms with deterministic control and dataflow [5].
- The medium-grained DREAM targets computation intensive applications that can iteratively run on small local data memories [6].
- The fine-grained M2000 embedded FPGA is suited for control-dominated tasks and variable data path widths [3].

Each HRE is seen by the system as a co-processor and has access to fast on-chip memories for buffering of local data. Off-chip memory access can be provided by multiple solutions, such as an off-the-shelf IP or a state-of-the art, multi-channel access optimizing DDR-SDRAM controller. In addition, a Network-on-Chip (NoC) interconnect was implemented in order to enhance the inter-HRE communication capabilities. The NoC accelerates data delivery between HREs via its spidergon design, [7] and enables processing of data streams by removing the parallel data transmission bottleneck common to centralized shared communication resources.

Finally, the MORPHEUS platform was designed to be highly flexible and extendable in several aspects. A flexible number of HREs and other peripherals can easily be added to/removed from the NoC, which supports a flexible (even) number of nodes. The architecture can therefore easily be tailored to the needs of a wide range of applications with considerably different requirements. For further information, the architecture and toolset are described extensively in [3].

For demonstration purposes, a silicon prototype was produced to allow application mapping and testing with real hardware. With the help of this prototype, it was shown in [8] that the platform flexibility, along with support for run-time reconfiguration, is critical to the successful mapping of such an application and its efficient reuse of costly chip resources. However, memory performance was found to be suboptimal. Thankfully, the extendable structure of MORPHEUS makes it an ideal platform for evaluation of the benefits to advanced applications provided by high-performance memory solutions within the reconfigurable computing paradigm.

III. Scalability of Memory Performance

Heterogenous architectures such as MORPHEUS can create the most challenging scenarious for memory controllers. Therefore, a key feature of the platform is the scalability of the memory subsystem. Due to area, power, and cost limitations required by embedded SoCs, it is important not to overdimension a design when requirements can be met by a reduced resource subset. At the same time, it is also important to be able to support advanced applications with increased resource requirements, should the need arise. In contrast to other reconfigurable architecture approaches, MORPHEUS addresses a wide range of applications with very different memory performance requirements.

Three scalability principles can be utilized with the MORPHEUS architecture. The first is the integration of a low-cost, basic memory controller connected to a single NoC node. This option is described in III-A and is suitable for applications with modest memory requirements.

The second option, described in III-B, integrates an advanced QoS access optimizing memory controller, which is scalable from 1 to 8 client read/write ports that can be connected to multiple NoC nodes. This solution can meet significantly higher memory demands.

A third and final scalability option for extremely high memory performance demands (not explored in this paper) is the integration of multiple CMCs, which is supported by the NoC interconnect through its flexible number of nodes.

A. Basic Memory Controller

For the MORPHEUS demonstration prototype, the ARM PrimeCell MultiPort Memory Controller (MPMC) [2] was integrated to provide access to external memory. Its advantages include basic QoS features and multiple client ports. In addition, the off-the-shelf IP has been utilized in multiple successful SoC designs by the project partner responsible for the physical layout and production of the silicon prototype.

However, for such a high-performance platform, the MPMC has major disadvantages that will become even more significant for extended platform variations with more processing power. The data bus has a maximum width of 32 bits for access to SDRAM and SRAM memories. For DDR-SDRAM, the maximum data bus width shrinks to 16 bits. More significantly, the MPMC does not support access optimization strategies, and can only internally buffer one outstanding read and write request, which creates a significant performance loss over theoretical maximum data rates.

Despite these disadvantages, the lightweight MPMC (26 KGates) is sufficient for many applications that do not require frequent high-speed access to external memory. In addition, the "silicon-proven" quality of the IP allows designers to integrate the peripheral into a target system very quickly, which also led to its selection for the MORPHEUS demonstration prototype.

B. Advanced QoS Access Optimizing Memory Controller

As previously stated, high-end applications designed for MORPHEUS require large amounts of memory and significant memory throughput to fully demonstrate its potential as a high-performance reconfigurable architecture. Without these architectural properties, each of the high-performance processing units will exhibit significant idle periods.

To prevent this data starvation and to eliminate external memory bottlenecks, a bandwidth-optimized DDR-SDRAM memory controller (CMC) (75 KGates), which was designed for high-performance FPGA and ASIC platforms, was extended with a high-speed NoC interface (26 KGates) for use with the MORPHEUS platform. Despite its focus on maximizing data rates, the controller also uses traffic shaping and supports multiple service levels to reduce latency when

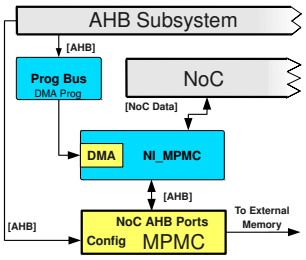

Figure 2. MPMC implementation architecture

necessary. Optimizations such as bank interleaving; which exploits the internal structure of DRAMs by accessing a second bank while another is busy; request bundling, which groups read and write requests together to minimize switch cycles; and variable-length buffers, which store multiple outstanding requests, can be used to significantly increase throughput and have been included in the CMC.

Despite the CMC's focus on throughput and achievement of up to 75% of the theoretical maximum DDR data rate at a given clock speed, access latencies also remain low compared to other solutions. A description of the controller architecture and access-optimizing strategies, its NoC interface, and preliminary throughput and latency results lies outside the scope of this paper and is available in [1] and [9].

IV. Memory Controller Integration

As described above, the extendable nature of the MORPHEUS platform supports multiple external memory controller solutions. This paper considers two such solutions. Information regarding the utilization of both memory controllers is summarized in Sections IV-A and IV-B.

A. MPMC Integration

Fig. 2 shows the silicon prototype's NoC connection to the memory controller. It uses the MPMC in combination with its own network interface, the `NI_MPMC`, which serves two purposes. First, it contains a direct memory access (DMA) unit, which may be programmed by the ARM9 via the AMBA bus subsystem to independently execute an NoC transfer. Secondly, it contains the bridges to handle the several protocol translations that must take place for a complete transaction with the MPMC. The NoC utilizes the protocol based on the high-performance STNoC described in [7]. It uses an advanced data protocol (`NoC Data`) for data transfers, and a more streamlined configuration protocol to transfer configuration information. But while the NoC uses ports based on the `NoC Data` protocol to send and receive packets to peripherals such as the memory controller, the MPMC communicates using only the AMBA Advanced High-performance Bus (AHB) protocol. Therefore, when writing data from the NoC, the `NI_MPMC` translates incoming requests from `NoC Data` signals to AHB signals before relaying the request onto the MPMC. When returning data from a read request from the NoC, the `NI_MPMC` must again translate before putting the data on its output ports. This introduces undesired latency overhead to each memory request sent to the MPMC.

The NoC uses 64-bit data buses and a 64-bit word size, and its flexible `NoC Data` protocol is capable of transferring packets ranging in size from 1-byte to 64-byte packets. The MPMC has a 32-bit data bus, and scatter-gather techniques

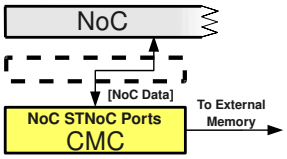

Figure 3. CMC implementation architecture

may be used to reassemble 64-bit data words. The MPMC is also capable of handling AHB bursts [2], which allows it to take advantage of multiple STNoC packet sizes.

B. Advanced Memory Controller Integration

The high-performance QoS CMC was designed to communicate directly with the NoC native `NoC Data` and configuration protocols via an advanced interface described in [1]. No additional translation between protocols is needed. Therefore, as shown in Fig. 3, the `NI_MPMC` can be completely removed. This also removes the DMA within the `NI_MPMC`, which means that transfers must only be setup by the DMA of the peripheral requesting data. Although not shown in Fig. 3, the `Prog Bus` from Fig. 2 still exists to program all DMA transfers. However, since the CMC performs direct transfers with the NoC, `Prog Bus` is not involved in CMC memory requests, as there is no DMA to program. The CMC is therefore connected directly to the NoC via a configurable number of nodes that have direct access to a CMC client port (up to 8). This direct connection removes the significant latency overhead required by expensive translations when using the MPMC.

V. Test Case Application Domain

Several applications were selected to be mapped to the heterogeneous computing platform. These include image processing applications used in intelligent surveillance systems, the PHY-layer for the IEEE 802.16e/j wireless standards, and real-time post-processing applications for digital film.

The application selected in this paper to evaluate the memory performance of the heterogeneous architecture is the Film Grain Noise Reduction (FGNR) application, which belongs to the real-time high-resolution image processing application domain. This application represents a real-world case study that requires enormous data rates due to its large input image size and real-time processing constraints.

A. Application Requirements

Real-time processing of high-resolution film or images requires high-performance processing architectures, as mentioned in Section I. The most common processing tasks are encoding, decoding, and post processing of digital film data, tasks which traditionally have not been executed in real time. With the increasing popularity of video distribution and broadcast methods such as video on demand, however, real-time encoders have become a necessity. Professional post-processing systems are also beginning to demand real-time computation because it allows immediate evaluation of results.

This real-time constraint implies that the algorithms must process continuous data streams to meet deadlines. The computation model of the MORPHEUS architecture explicitly supports stream-based processing. However, without adequate throughput to and from external memory, such applications

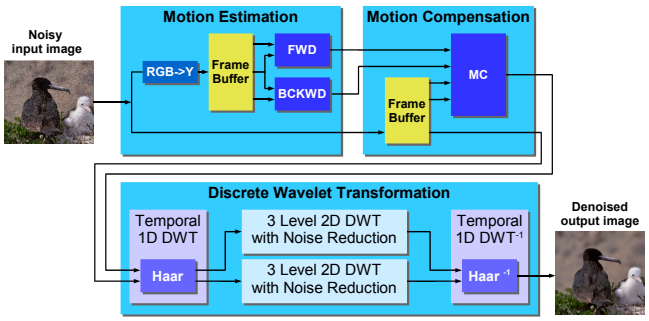

Figure 4. Advanced noise reduction algorithm

become data-starved and cannot meet real-time requirements, despite stream processing support.

Of all the applications implemented on the heterogeneous reconfigurable platform, the FGNR application exhibits the most demanding memory requirements, processing several data streams at a total of 28 GiBit/s (19.5 GiBit/s read, 9.5 GiBit/s write at 2K[1] resolution). These requirements show that the FGNR application will receive the greatest advantages from a multi-channel bandwidth-optimized memory controller.

B. Film Grain Noise Reduction Application

Film grain noise reduction is a valuable technology for the digital cinema market, allowing reduction or removal of the undesired noise that is often introduced during analog to digital source conversion. In this context, a lossless algorithm, which implies exhaustive search comparisons, is a necessity due to the stringent quality requirements of digital cinema data. Therefore, algorithms such as 3DRS [10] that are less bandwidth intensive than full-search algorithms are not applicable. The underlying theory of the algorithm used for the application is outlined in [11], and an FPGA-based implementation of the application is presented in [12].

Fig. 4 depicts the algorithm decomposed into three distinct modules, each with different processing characteristics and massive data requirements: a bidirectional Motion Estimation (ME) unit, a Motion Compensation (MC) unit, and a 2.5 dimensional Discrete Wavelet Transformation (DWT). The processing requirements of each module are described in [8].

The computational requirements can be met by the three MORPHEUS HREs due to their support for run-time reconfiguration. In order to process a frame entirely, the final mapping of the application requires reconfigurations of the DREAM and XPP, the two heavily-utilized HREs. However, with reconfiguration times as low as 2 clock cycles for the frequently reconfigured DREAM and 1000 cycles for the less frequently reconfigured XPP, this step becomes trivial. Therefore, from the application's point of view, the architecture can be seen as a reconfigurable stream-processing platform that can effectively process data as fast as it arrives.

The final mapping to the heterogeneous reconfigurable architecture, including a description of necessary HRE reconfigurations, is shown in detail in [8].

VI. Performance Evaluation

This section describes the experiments used to evaluate the heterogeneous reconfigurable platform using the described memory controller solutions.

[1]2K implies 2048x1556 pixels/frame, 32 bits/pixel, and 24 frames/s

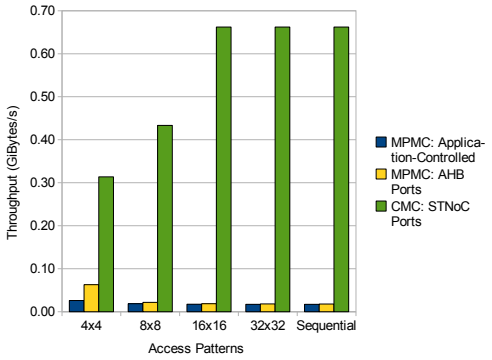

Figure 5. Write throughput per block

A. Experimental Setup

The following experiments are designed to quantify memory throughput and access latency available to the MORPHEUS HREs using both of the described memory controller solutions. Memory requests were generated during simulation of the complete MORPHEUS architecture using the CMC and MPMC in a Modelsim 6.4C environment. The simulations utilized memory traces extracted from the FGNR application, which was fully implemented on the MORPHEUS platform. The traces exhibit two caterogies of memory access patterns. The first, produced by the MC module, uses a block-by-block method pattern to read and write image data; the second access pattern consists of reads and writes to consecutive addresses until the entire image is processed, as seen in the ME module. The block-by-block method divides the image into a given block size, then reads each of these blocks individually, row by row. This represents a streaming data access pattern, especially at larger block sizes where large, consecutive data chunks are accessed. Both memory controllers are idle during the comparatively lengthy DMA reprogramming time between neighboring blocks.

Throughput measurements were taken both for the read and write of the entire image, which includes the DMA reprogramming time, and for the read and write of a single block, which only considers the active period of the memory controller. All experiments use a 32×32-pixel image. This size was selected to allow the entire image to be stored in on-chip memory when necessary. Larger images could easily be supported by removing this requirement or increasing on-chip memory sizes. However, the small image size has no direct impact on the experimental results and will not effect memory throughput or latency. These values change with varying access patterns, but not with varying image sizes.

The first experiment measured the throughput and latency of the MPMC memory controller exactly as it is implemented in the MORPHEUS silicon prototype. A C application, which mimics the MC and ME memory access patterns, was used to program a DMA to handle all NoC transactions.

At the time of our experiments, the provisions to assemble multi-word NoC transfers were not implemented in the NoC infrastructure for use in applications, including the one used in the above test. Therefore, two experiments were designed to place requests directly on the ports of the memory controllers. This allowed the comparison of each controllers' handling of large, multi-word `NoC Data` packets, assuming ideal translation to AHB in the MPMC case.

	MPMC (GiByte/s)		CMC (GiByte/s)	
Block Size	*App.-controlled*	*AHB Ports*	*STNoC Ports*	*Speedup*
MC: 4x4 Write / Read	0.026 / 0.012	**0.022 / 0.015**	**0.314 / 0.157**	**5.00 / 10.18**
MC: 8x8 Write / Read	0.019 / 0.011	**0.019 / 0.015**	**0.433 / 0.298**	**20.02 / 19.28**
MC: 16x16 Write / Read	0.017 / 0.012	**0.018 / 0.015**	**0.662 / 0.604**	**35.65 / 38.98**
MC: 32x32 Write / Read	0.017 / 0.012	**0.018 / 0.016**	**0.662 / 0.604**	**36.84 / 38.94**
ME: Sequential Write / Read	0.017 / 0.012	**0.018 / 0.015**	**0.662 / 0.598**	**37.08 / 38.62**

Table I
WRITE / READ THROUGHPUT FOR BLOCK-BY-BLOCK ACCESS

It must be noted that the MPMC controller supports Flash, SRAM, SDRAM and DDR-SDRAM memories [2]. However, to reduce complexity of the physical IO interface as well as costs for the prototype, all SDRAM ports were removed and SRAM was used for external memory. Therefore, while CMC results are presented for DDR-SDRAM, it was only possible to perform the above MPMC experiments with SRAM, which has faster access times than SDRAM at single-data-rate operation. With the SDRAM ports available, the MPMC could again utilize DDR-SDRAM memory. However, improvements provided by double-date-rate access are immediately negated by the required MPMC data bus width reduction from 32 to 16 bits when using DDR-SDRAM [2]. The CMC's use of DDR-SDRAM memory with a 64-bit data bus presents itself here as an appropriate expansion over both the 32-bit SRAM implementation found in the MORPHEUS prototype and the available 16-bit DDR-SDRAM MPMC configuration. Performance improvements due solely to data bus width increase and double-data-rate operation are duly attributed below, thereby making the comparison fair.

B. Throughput Results and Analysis

Tables I and II present the results of the block-by-block and full image throughput experiments described above. The tables present write and read data rates in the same column. Measured values were the same for each block in an image. Bold numbers indicate the values selected for the speedup calculation. These values were obtained by measuring throughput rates directly on the MPMC AHB ports and the CMC STNoC ports, which allows for the most fair comparison. For all throughput experiments, purely application-controlled results, which were generated by the application running on the simulation platform, are presented for the MPMC only, as this was not possible with the CMC (see Section VI-A). However, this is not a critical issue for our tests, since, as clearly shown in Fig. 5, the observed values show marginal differences between application-controlled memory requests and those obtained by stimuli that reconstruct application behavior during simulation.

The results show that in the block-by-block MC experiments in Table I, the CMC throughput bests the MPMC throughput by a minimum factor of 5 and a maximum factor of 39. In all cases, a factor of 2 can be attributed to the CMC's use of DDR-SDRAM, where two data words are written/read per clock cycle, and an additional factor of 2 speedup is due to the CMC's 64-bit data bus. However, the remaining increase must be credited to the CMC's access optimization techniques and bandwidth-optimized design. CMC data rates increase with block size until 16×16, while MPMC data rates remain nearly constant for all block sizes in both read in write directions. A single row of a 16×16 block contains 64 bytes, which is exactly the size of a full `NoC Data` burst, as well as the size at which the CMC performs optimally [9]. This explains why no significant increases are seen above this block size. Rows of blocks smaller than 16×16 are read and written by a burst smaller than a full 64-byte data burst. However, because these read and write requests are smaller in size, less data is read/written per time unit than the CMC supports. For all block sizes 16×16 or greater, this is no longer the case, and the CMC's burst-oriented design and larger data path are fully utilized. When block-size is not an issue, as in the case of the sequential ME writes/reads in Table I, similar data rates are observed. This is due to the fact that this access pattern, like those of block size 16×16 or greater, also accesses memory in full 64-byte bursts, allowing maximum throughput.

Fig. 5 concisely illustrates this analysis in graphical form. These observations hold for both read and write requests. A read graph is omitted, as it exhibits an identical relationship.

Table II presents results when considering the entire image. The increased precision shown here is due to extremely small values in the 4×4 block case. When processing an entire image, the time required to reprogram DMAs involved in the transfer becomes relevant, as these reprogramming times can reach more than 7000 cycles and occur when accessing pixels from different blocks. This occurs frequently when the MC processes an entire image, and the frequency increases with decreasing block size. It should be noted that although the CMC's connection to the NoC does not contain a DMA, a DMA belonging to another peripheral must be programmed to execute a transaction with the CMC; therefore, DMA reprogramming time is still relevant to the CMC throughput when processing an entire image. This large delay dominates the measurement window, which leaves both memory controllers idle for the majority of the time period under examination and reduces the relative significance of CMC's performance gains over the MPMC. Accordingly, the speedups in Table II are more modest than those in Table I. At block size 32×32, which matches the size of the entire image, as well as in the sequential pixel-by-pixel ME write/read test, there is no longer interblock reprogramming time. Therefore, the optimized design of the CMC again greatly outperforms the MPMC when the dominating idle cycles can be removed. Countermeasures to optimize DMA operations to automatically reduce such delays are outside the scope of this paper.

C. Latency Results and Analysis

Despite its bandwidth-optimized designed and deep internal buffers to facilitate memory access optimization, the CMC consistently offers significantly lower latencies than the MPMC. The results of several latency experiments are provided in Tables III and IV. Data is presented only for write data due to similarities in the read and write results. Table IV examines a special case, the average latency of the final word in a block, as it represents a worst case for the

	MPMC (GiByte/s)		CMC (GiByte/s)	
Block Size	*App.-controlled*	*AHB Ports*	*STNoC Ports*	Speedup
MC: 4x4 Write / Read	0.00093 / 0.00064	**0.00094 / 0.00078**	**0.00098 / 0.00082**	**1.01 / 1.05**
MC: 8x8 Write / Read	0.00310 / 0.00266	**0.00349 / 0.00282**	**0.00425 / 0.00341**	**1.19 / 1.21**
MC: 16x16 Write / Read	0.00960 / 0.00702	**0.00984 / 0.00817**	**0.02109 / 0.01680**	**2.10 / 2.06**
MC: 32x32 Write / Read	0.01703 / 0.01183	**0.01779 / 0.01550**	**0.66227 / 0.60359**	**36.84 / 38.94**
ME: Sequential Write / Read	0.01702 / 0.01183	**0.01779 / 0.01548**	**0.66227 / 0.59791**	**37.08 / 38.62**

Table II
WRITE / READ THROUGHPUT FOR ENTIRE IMAGE

CMC. When the final word of a block is transferred, the CMC has already experienced a "warm-up" period and its internal request buffers, used to optimize memory requests, are at their fullest. This could imply increased access latencies.

It should be noted that the CMC was designed to communicate using the NoC's native protocol and consequently is not subjected to increased latencies caused by complex protocol translations. These differences can been seen by comparing the MPMC application-controlled column with the CMC column in Table III. Since this gives the CMC a clear advantage but is not specifically related to the memory controller architecture itself, a more interesting comparison is between the MPMC AHB port and CMC columns. Here, the protocol translation times are removed from the MPMC, providing a more direct comparison between the bold columns. These values are used for the percent reduction calculations. The CMC still outperforms the MPMC by 67% in the average case and 37% in the worst case (maximum latency). The worst case is again examined for all access pattern types in Table IV. For each pattern, the CMC completes memory requests faster than the MPMC. This is a significant victory for the SDRAM-based CMC, since SRAM latencies are much faster than SDRAM latencies when only the memory module (without controller) is considered. Therefore, even if the time required for the MPMC protocol translation is made negligible, it is clear that the MORPHEUS platform using the CMC will experience greatly reduced memory access latencies.

VII. CONCLUSION

In this paper, we have examined a heterogeneous reconfigurable architecture and quantified the performance of its scalable memory subsystem using two memory controllers.

	MPMC		CMC	
	App.-controlled	*AHB ports*	*STNoC ports*	% Reduction
Latency	cycles	cycles	cycles	
Avg.	126.19	**70.94**	**23.31**	**67%**
Max	161.00	**84.00**	**31.00**	**37%**
Min	41.00	**24.00**	**15.50**	**35%**

Table III
WORST CASE LATENCY FOR WRITE ACCESSES, BLOCK SIZE 4×4

	MPMC	CMC	
	AHB ports	*STNoC ports*	% Reduction
Access Type	cycles	cycles	
4x4	84.00	29.00	63%
8x8	84.00	29.00	65%
16x16	84.00	18.00	79%
32x32	84.00	18.00	79%
Sequential	84.00	18.00	79%

Table IV
WORST CASE LATENCY OF FINAL WORD WRITES

Through thorough experimentation and careful analysis, we have shown vast throughput and access latency improvements offered by a high-performance memory controller solution, which enables real-time processing of significantly larger resolutions on the heterogeneous reconfigurable platform. This demonstrates the potential of the MORPHEUS platform for data-intensive applications such as the FGNR.

A final important observation is that these results allow developers of high-performance image processing algorithms to make accurate estimates on the performance of potential MORPHEUS applications, as well as quickly calculate, based on the integrated memory solution, the application scalability with respect to image size. For example, the CMC offers a throughput speedup factor of just over 36 for sequential read and write accesses. Since the amount of raw image data scales quadratically with image resolution, a developer can quickly see that the CMC enables real-time processing of resolutions 6 times greater than those supported by the MPMC.

REFERENCES

[1] S. Whitty and R. Ernst, "A Bandwidth Optimized SDRAM Controller for the MORPHEUS Reconfigurable Architecture," in *Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, April 2008.

[2] ARM Ltd., *PrimeCell MultiPort Memory Controller (PL175)*, ARM Ltd., 2003.

[3] N. Voros, A. Rosti, and M. Hübner, Eds., *Dynamic System Reconfiguration in Heterogeneous Platforms - The MORPHEUS Approach*. Springer Netherlands, 2009, vol. 40.

[4] F. Thoma, M. Kühnle, P. Bonnot, E. M. Panainte, K. Bertels, S. Goller, A. Schneider, S. Guyetant, E. Schüler, K. D. Müller-Glaser, and J. Becker, "MORPHEUS: Heterogeneous Reconfigurable Computing," in *Proceedings of 17th International Conference on Field Programmable Logic and Applications (FPL07)*, August 2007.

[5] V. Baumgarte, G. Ehlers, F. May, A. Nückel, M. Vorbach, and M. Weinhardt, "PACT XPP—A Self-Reconfigurable Data Processing Architecture," in *The Journal of Supercomputing*, 2004.

[6] F. Campi, A. Deledda, M. Pizzotti, L. Ciccarelli, P. Rolandi, C. Mucci, A. Lodi, A. Vitkovski, and L. Vanzolini, "A dynamically adaptive DSP for heterogeneous reconfigurable platforms," in *Proceedings of the Conference on Design, Automation and Test in Europe*. ACM Press New York, NY, USA, 2007, pp. 9–14.

[7] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra, "Spidergon: a novel on-chip communication network," in *Proceedings of the International Symposium on System-on-Chip*, 2004, pp. 16–18.

[8] S. Whitty, H. Sahlbach, R. Ernst, and W. Putzke-Röming, "Mapping of a Film Grain Removal Algorithm to a Heterogeneous Reconfigurable Architecture," in *Proceedings of Design, Automation and Test in Europe (DATE)*, April 2009, pp. 27–32.

[9] S. Heithecker, "Communication and Memory Scheduling in Reconfigurable Image Processing Systems," Ph.D. dissertation, 2008.

[10] G. de Haan, P. Biezen, H. Huijgen, and O. A. Ojo, "True motion estimation with 3D recursive search block matching," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 3, pp. 368–379, October 1993.

[11] S. Eichner, G. Scheller, U. Wessely, H. Rückert, and R. Hedtke, "Motion compensated spatial-temporal reduction of film grain noise in the wavelet domain," in *SMPTE Technical Conference, New York*, 2005.

[12] S. Heithecker, A. do Carmo Lucas, and R. Ernst, "A High-End Real-Time Digital Film Processing Reconfigurable Platform," *EURASIP Journal on Embedded Systems, Special Issue on Dynamically Reconfigurable Architectures*, vol. 2007, pp. Article ID 85 318, 15 Pages, 2007.