MANCHESTER
1824
The University of Manchester
The University of Manchester Research

Unexpected Diversity: Quantitative Memory Analysis for Zynq UltraScale+ Systems

Unexpected Diversity: Quantitative Memory Analysis for Zynq UltraScale+ Systems

Author

Email: author@blinded.org

***Abstract*—Memory throughput is one of the major bottlenecks for accelerator performance. Now that Zynq UltraScale+ systems are being deployed at exascale to edge, it is important to understand its limitations and optimizations possible for developers. In this paper, we extensively evaluate the memory performance and behaviour for various AXI ports combinations, burst sizes, access patterns, and the number of accelerators per AXI port. Our results on ZCU102 and Ultra 96 boards show that 1) effective throughput of these systems is only 75% and 92.5% of theoretical maximum respectively, 2) 128 and 192 byte burst size is often optimal, 3) AXI ports of the same type may not always exhibit similar behaviour, 4) multiplexing accelerators in PL can provide better throughput distribution compared to multiplexing in PS, and 5) using all AXI ports does not lead to the highest performance.**

I. Introduction

Multicore and Multiprocessor SoC (MPSoC) systems like Zynq UltraScale+ are being deployed for exascale systems [1] to edge hubs [2] and other embedded systems due to their performance and energy benefits in the post-Moore era. The adoption can be largely attributed to the combination of an ARM CPU core for control-oriented tasks and an FPGA fabric for application acceleration, allowing rapid development and performance optimisation while keeping the energy consumption minimal. However, one common problem for high-performance systems is the memory wall, i.e. accelerators often starve due to the lack of memory throughput and high latency [3]–[5]. With systems moving towards edge hubs and cloud environments, multiple applications will be required to run concurrently in these systems [6], further worsening the memory contention problems.

Hence, developers for these systems must understand the nature of the memory sub-system of these MPSoCs and design accelerators which capitalise on it for high performance. The main system design parameters which dictate the memory throughput for such systems are:

- AXI ports: Different AXI ports can potentially provide different memory behaviours based on internal priorities of the memory subsystems.
- AXI data widths: The data width dictates how many bits can be transferred per clock cycle and is directly proportional to the memory throughput.
- Burst sizes: Depending on the memory controller, the burst size may allow bulk read/write operations faster than by using fine-grained memory accesses.
- Memory organisation: Depending on the internal structure of a memory controller (e.g., row-bank or bank-row), the level of row reuse may change leading to low or high throughput.
- Access patterns: Depending on the memory controller's ability to reorder and analyse the trends in transactions, it may perform large read/write accesses for higher performance.
- Transaction frequency: This directly corresponds to how many transactions can be transferred between memory and the accelerators per second.
- Number of accelerators per AXI port: This corresponds to the amount of multiplexing overhead in the programmable logic as well as to the pollution in memory controllers at the level of row and bank accesses.

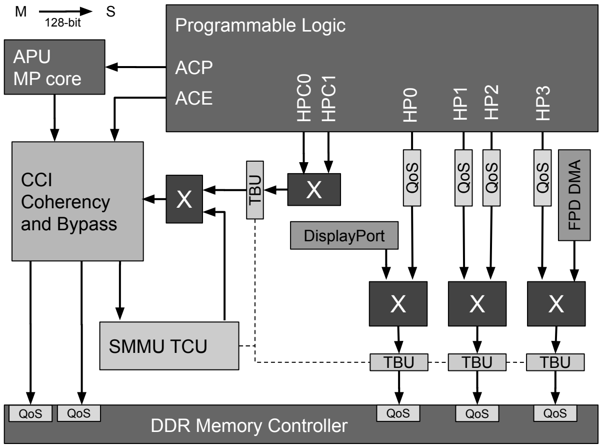

Fig. 1: AXI interconnect map for memory on Zynq UltraScale+ devices [7].

In this paper, we thoroughly analyse these parameters for the Zynq UltraScale+ FPGAs on ZCU102 and Ultra-96 boards and highlight trends and provide insights into memory systems for accelerator and system developers. Additionally, to the best of our knowledge, this study presents one of the first memory bandwidth analysis for FPGA accelerators on Zynq UltraScale+ boards. Further, the complete set of results (4800 data points of 50 seconds runtime each) are publicly available[1]. Our key contributions in this paper are:

- Design of a testing harness for memory performance evaluation with varying parameters (Section III).
- Quantitative assessment of memory behaviour in various configurations (Section IV).
- Design guidelines for system design and application performance tuning (Section IV, V, and VI).

II. Zynq UltraScale+ Memory System

According to the Zynq UltraScale+ Device Technical Reference Manual from the vendor (Xilinx [7]) the ARM system-on-chip which consists of processors, AXI interconnects and the memory controller is identical for all Zynq UltraScale+ devices. Fig. 1 shows a simplified view of the the connections between programmable logic (PL), ARM Cortex-A53 (refer

[1] All the graphs and data can be found at https://bit.ly/2yQp3O5

TABLE I: Platform specification for ZCU102 and Ultra 96 board.

	ZCU102	Ultra96
FPGA	XCZU9EG-2	XCZU3EG-1
BlockRAM36K	912	216
DSP	2,520	360
Logic Slices	34,260	8,820
Processing System	Cortex-A53 + R5	Cortex-A53 + R5
APU Frequency	up to 1.5GHz	up to 1.5GHz
Level 1 Cache	32 KiB	32 KiB
Level 2 Cache	1 MiB	1 MiB
Bank Organisation	4 Bank Groups × 4 Banks each	2 Ranks × 8 Banks each
DDR Capacity	4GB	2GB
DDR throughput	2400 MT/s × 64-bit	1066 MT/s × 32-bit

to as APU), and the memory controller. The interconnect provides four types of AXI ports to programmable logic which are used for communication with the CPUs and memory:

- High-Performance (HP) port: Four 128-bit wide instances without any coherency support.
- High-performance Coherent (HPC) port: Two 128-bit wide instances with one way I/O coherency support.
- Accelerator Coherency Port (ACP): One 128-bit wide instance with coherency support for I/O and CPU's L2 cache.
- AXI Coherency Extension (ACE): One 128-bit wide instance with full coherency support.

There are three types of memories available in Zynq Ultrascale+ systems: i) On-chip memory (which consists of block RAMs and distributed memory), ii) level 1 and level 2 caches inside the CPUs, and iii) off-chip DDR memory. This means that depending on the memory used, state information is located inside the CPU, inside the programmable logic or in external RAM. Table I shows details of these memories for some popular boards.

There also exist other components between PL and the DDR memory controller (see Fig 1): Translation Buffer Unit (TBU), Multiplexers, a Display-port, a Full Power Domain DMA engine (FPD DMA), and Quality of Service (QoS) buffers. These can potentially affect the behaviour and achievable memory throughput of hardware accelerators. However, these are often not considered by the developer as they exist inside the ARM SoC and Vivado does not provide direct control over these components.

Overall, an accelerator developer usually expects the following behaviour from the memory system:

1) Same type of AXI ports should show same performance behaviour.
2) Increasing burst size should improve both read and write performance until it saturates (i.e. a logarithmic relation).
3) Increasing the number of AXI ports to memory should increase memory throughput either linearly or sub-linearly.
4) Memory behaviour should be similar across different boards with the same memory controller except for the highest throughput achievable from the memory SODIMM available on a particular board.
5) Multiplexing in the PL and PS should provide similar throughput behaviour except for the higher overhead (latency and logic) posed by the soft-logic AXI multiplexing.
6) Sequential memory access patterns should provide considerably higher performance than random access pattern.

Our memory evaluation results reveal that 4 out of these 6 assumptions (1-4) do not hold and the remaining 2 (5 and 6) do only in certain circumstances as detailed in Section IV.

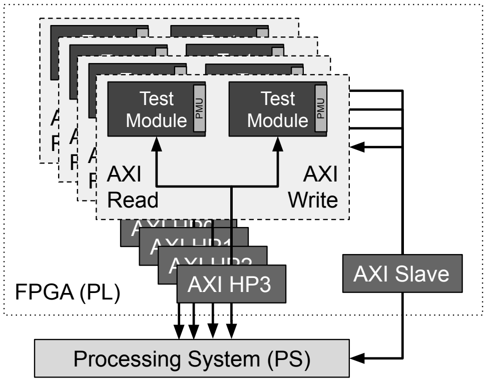

Fig. 2: Experiment setup in the programmable logic for memory test patterns.

III. Experiment Setup

In our experiments, we consider two widely available Zynq UltraScale+ boards - a ZCU102 Evaluation Board and an Ultra96. Details of these boards are shown in Table I.

Our prime objective for evaluation is the DDR memory alone as in most cases caches are useful only for the CPUs due to their small size. This is because using on-chip memory as a scratchpad memory in the programmable logic (PL) commonly provides better control, throughput and latency for accelerators than caches. Further, the memory performance can become more unpredictable due to the interference caused by applications running on CPUs and the accelerators in PL. The goal of this paper is to examine memory effects that preliminary relate to accelerators located in the PL part of the system but that operate on DDR memory connected to the ARM SoC. We therefore do not examine any caching effects in this paper. For the analysis of available bandwidth from the Cortex-A53 and Cortex-R5 cores, we refer the readers to [8].

A. Hardware Setup

We have implemented and placed RTL modules in the Programmable Logic (PL). These modules independently use the read and write ports of the 4 High-Performance AXI ports. Our modules can simulate the placement of up to eight accelerators — four write memory intensive (e.g., mandelbrot frame rendering) and four read memory intensive (e.g., a MapReduce accelerator) or combinations of these. Fig. 2 depicts the connection of these modules with the PS. Each test module is fully programmable from the host CPU, allowing us to change its state at runtime for adjusting the number of operations, memory address spaces and burst lengths. To ensure there are no conflicts between accelerators, we reserve a unique address space in the DDR memory for each accelerator. All modules are instrumented with an integrated Performance Monitoring Unit (PMU), which allows us to obtain *real-time information* about read/write operations executed, active runtime, as well as average and maximum read latencies. The PMU registers are memory-mapped and are controlled using the AXI Slave as shown in Fig. 2.

B. Test Case Generation

The base experimental setup consists of all AXI combinations configured to measure read-only, write-only, and read-write performance for each setup. The test modules are directly connected to the PS HP AXI ports without any additional interconnect as an intermediary. Further, these modules are controlled from the CPU in bare-metal mode to minimize memory overhead and interference caused by running applications and an operating system. Moreover, our tests are long and free-running, i.e. each configuration runs for 5 seconds before being stopped to capture the results (which include many refresh cycles). Every configuration is tested 10 times to minimize and quantify errors.

The system runs with a global clock of either 100MHz or 300MHz in different test scenarios. The read-only and write-only tests are executed for both ZCU102 and Ultra96 at 300MHz to achieve maximum available single-AXI performances, while our base read-write tests run at 300MHz and 100MHz on the ZCU102 and Ultra96 respectively because many systems may not be able to run their accelerators at 300MHz.

The ARM CPU cache line size is 64 bytes in Zynq Ultrascale+ devices. Given that the DDR memory controller is also supplied by ARM, someone would anticipate that it is largely optimised to operate on memory accesses of this burst size. Moreover, cache lines are always aligned in memory, which should also be considered when using the DDR controller. In our experiments, small bursts (up to 512 Bytes) are multiple of 16 and large burst sizes are always the multiple of the ARM cache line size to ensure optimal operation of the DDR controller. Additionally, all of our accesses are memory aligned to the burst size for the test (16 Byte bursts will have data aligned to 16 byte boundaries). Data alignment and positioning is a common technique that can overcome DDR controller alignment issues as well as optimal cache line utilization from the software [9]. The maximum achievable AXI burst size for 128-bit AXI configuration is 4 KiB, thus we evaluate AXI performance with configurations of up to 4 KiB. Note that the DDR memory controller is capable of executing memory requests out of order and has QoS modules that can define the order of requests to the memory [7]. In situations where the controller can not select more optimal execution ordering, read requests are prioritised over write requests [7].

We use a DDR controller configuration to map the DDR address space in Row-Bank-Column fashion for our base case, which is the default and widely used configuration in computing systems, as it results in bank interleaving when accessing large sequential arrays of data. Additionally, we set the System Memory Management Unit (SMMU) into by-pass mode for the accelerators to minimize the throughput overhead caused by the Translation Buffer Units (TBU).

With these base settings, we exhaustively test every AXI and frequency combination for multiple different burst sizes to characterise memory performance under various scenarios. Individual changes are made in the base setup to evaluate the impact of access patterns, multiplexing in PL and Quality of Service (QoS) in isolation (i.e. keeping all other parameters constant).

IV. Evaluation

In the following subsections we evaluate eight primary areas: 1) peak performance, 2) performance of AXI ports, 3) frequency, 4) access patterns, 5) memory organisation, 6) multiplexing overhead in PL, 7) quality of service and 8) performance distribution impact. Based on our experiments we found that throughput scales linearly with AXI width size and, hence, to quantify the maximum throughput achievable all the experiments from here on use 128-bit AXI connections.

A. Peak Performance

We evaluate the peak memory performance provided by the boards using read-only and write-only test scenarios. The normalised results are presented in Fig. 3. Since the DDR theoretical performance for ZCU102 is larger than the available throughput of a single AXI port, we also include the test case using multiple AXI configuration that achieves the largest throughput (using HP0, HP1 and HP3 simultaneously). We observed that:

- The write-only throughput is larger than the read-only throughput with on average by 11-13%.
- At a burst size of 128 Bytes, the throughput reaches (near) maximum for all configuration.
- Burst sizes using a multiple of 64 Bytes yield local peak performances except for the burst sizes which are also multiple of 256 Bytes as they show decreased throughput for various AXI ZCU102 read-only configurations (see the strong oscillating behaviour in Fig. 3).

The Ultra96 configuration uses a single AXI that issues sequential bursts on either read or write port. This avoids request multiplexing or changing between read and write mode in the DDR controller, thus achieving a high maximum throughput of 92.5% of the theoretical DDR memory peak on the Ultra96. In contrast, the ZCU102 needs to utilise multiple AXI ports to achieve the highest throughput, which leads to some multiplexing in the DDR controller between the requests and does not scale linearly with the number of ports. The peak performance in ZCU102 was found at 128 Byte bursts, which is 75% of the theoretical peak for the DDR memory.

Note, contrary to the common assumption, of *using all AXI ports does not lead to the highest throughput* in either of the boards as HP1 and HP2 are multiplexed in the PS (see Fig. 1).

B. Performance of AXI Ports

We execute read/write base tests utilising all AXI combinations and burst sizes, to observe their respective behaviours. The tests are executed at 100MHz and 300MHz respectively for both Ultra96 and ZCU102.

1) Standalone: When a single AXI is performing memory accesses, both of its read and write ports typically utilise different memory regions, which can essentially lead to row pollution of each other. We find that all AXIs in ZCU102 behave similarly, while HP0 in Ultra96 behaves differently than HP1-3 (see Fig. 4). In particular, Fig. 5 shows that the read/write distribution of Ultra96 HP1-3 ports are balanced near 50% each, whereas, Ultra96 HP0 and all ZCU102 AXI ports tend to majorly prioritise read operations over write operations.

We can rationalise the behaviour discrepancy between the ports on Ultra96 by the fact that HP0 shares its DDR connection with the DisplayPort that uses DDR for frame buffering. However, the same behaviour discrepancy is not observed for ZCU102 which objects our explanation. We therefore examined the differences we found in Quality of Service (QoS) module configurations for an answer in Section IV-G. However, this is highly unexpected since according to technical documentation [7], the QoS module of HP0 does not influence the behaviour of the memory accesses of the DisplayPort.

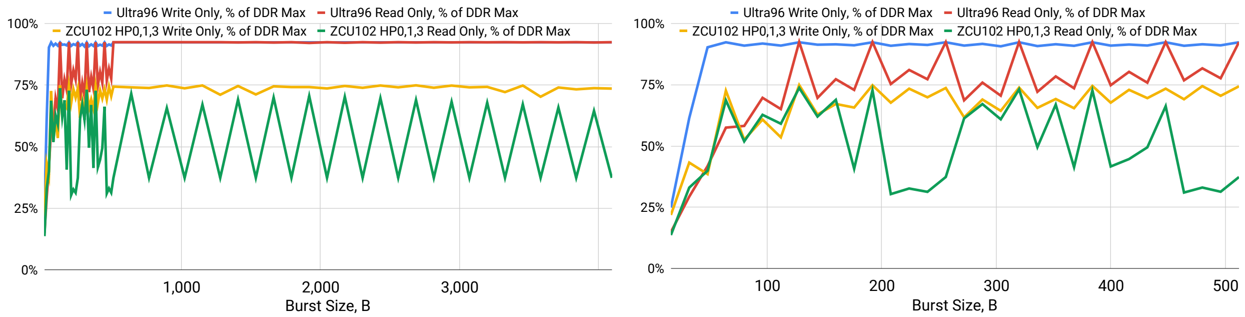

Fig. 3: Effective throughput achieved in read-only and write-only configurations with respect to theoretical peak DDR throughput.

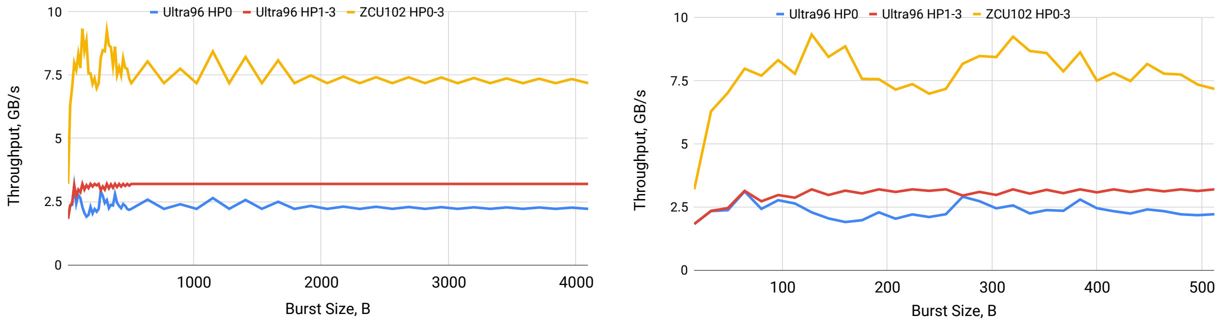

Fig. 4: Measured throughput in single AXI base case

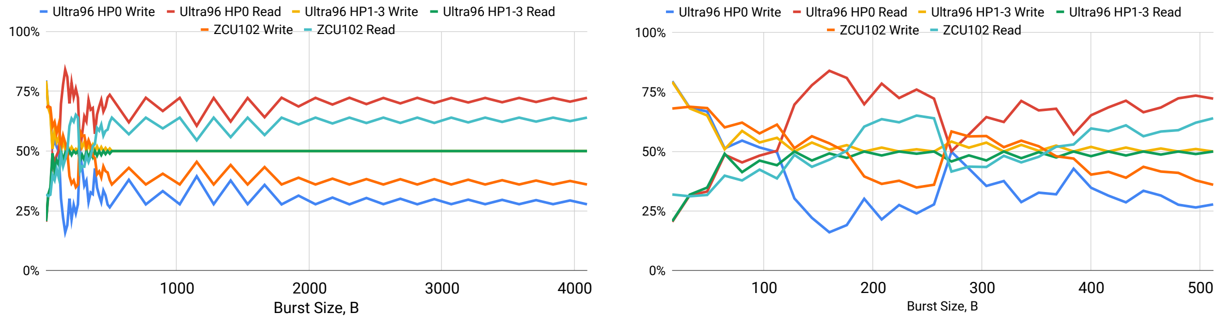

Fig. 5: Measured Read/Write distribution in single AXI base case

Overall, we can conclude that *1) same type of AXI ports may not necessarily show same performance behaviour and 2) after a point, increase in burst size may backfire depending on port's QoS settings*.

The average read latency in all single AXI configurations on both boards was found between 250 and 10,000 clock cycles and scales linearly with respect to burst size.

2) Combinations: When multiple AXIs perform memory requests on the same memory region, significant row pollution might occur. Fig. 6 and Fig. 7 show the effective memory throughput in this scenario with different AXI combinations.

The trends on Ultra96 are similar for all configurations with at most about 20% difference between best and worst-performing configurations. We found that the best performing configuration is the one including HP1-3. This configuration also has balanced read/write distribution, but due to the multiplexing of HP1 and HP2 in PS, HP3 receives 50% of the available throughput, leaving HP1 and HP2 with only a 25% share for each (see Fig. 14). In all configurations with HP0 enabled, HP0 obtains a significantly larger portion of the available throughput and read requests deliver at least 20% more throughput than write requests. The ZCU102 in contrast always has a balanced distribution between different AXIs (except HP1 and HP2, which share an AXI port to DDR). Configurations HP0,1,3 and HP0,2,3 show oscillating throughput behaviour with respect to the burst size (see Fig. 7) and achieve the highest measured throughput of 13,721 MB/s at 192-byte bursts. Whereas all two-AXI configurations except HP1-2 achieve a more stable throughput trend, peaking at 128-byte bursts to 11,600 MB/s and 320-byte bursts to 12,640 MB/s. Notably, *the more AXIs in a configuration, the higher the imbalance between read and write throughput*, with 4 AXI combinations and burst sizes of more than 128 Bytes, the write throughput reaches only 1% of the total.

On the Ultra96, the average read latency of HP0 in all AXI combinations is the same as standalone HP0. In configurations including HP0, all other AXI ports experience an average latency increase of up to an order of magnitude higher than their standalone cases (see Fig. 10). This is because the read port of HP0 is configured to a higher priority by default, which pollutes the read (and write) requests from the other AXIs. On the ZCU102, the average read latency in multiple AXI configurations increases to about 2× for 3-AXI and 4-AXI configurations (see Fig. 11). In both devices, HP1 and HP2

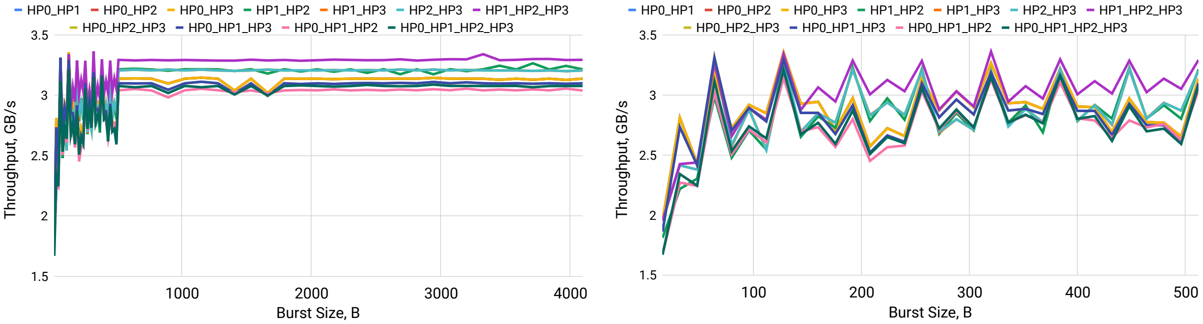

Fig. 6: Measured throughput in multiple AXI configurations on the Ultra96

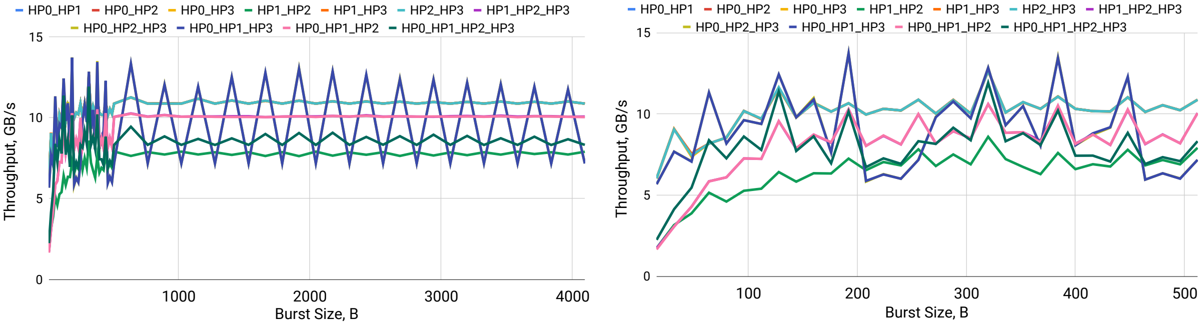

Fig. 7: Measured throughput in multiple AXI configurations on the ZCU102

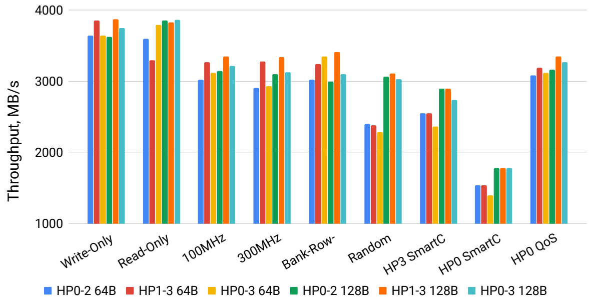

Fig. 8: Measured throughput in multiple AXI configurations on the Ultra96

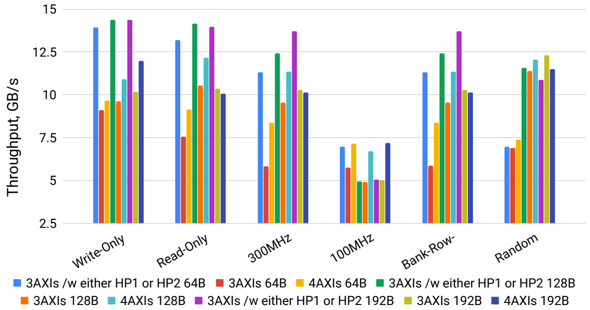

Fig. 9: Measured throughput in multiple AXI configurations on the ZCU102

observe up to a 2× latency increase over the other AXI ports if they are simultaneously used in a configuration.

We do not observe any large periods of AXI read unresponsiveness, that might be caused by long DDR refresh procedures. The DDR controller manages to hide the refresh cycles as (other than HP0 which is polluting the other AXIs in Ultra96) we observe maximal AXI read inactivity of about 200-300 cycles in our tests.

C. Frequency

As both boards feature the same ARM SoC, the theoretical aggregated throughput between PS and PL is 12.8 GB/s (4 AXI ports at 2x1.6 GB/s each) when running the PL at 100MHz. When we test the ZCU102 at 100MHz, we observe a peak performance of 8,800 MB/s when operating all AXI ports at bursts of 384 Bytes. This is only 14% lower than the same configuration running at 300MHz but is 36% lower than the HP0,1,3 and HP0,2,3 configurations running at 300MHz (see Fig. 9). Additionally, the read requests take all the available AXI read ports throughput and the mentioned configuration achieves 6.4 GB/s read throughput but only 2.4 GB/s write throughput. Since all 4 AXI read ports achieve their theoretical maximum of 1.6 GB/s, this reveals that the HP1-HP2 multiplexing in PS is happening after clock domain crossing at a higher PS frequency.

Whereas, running Ultra96 at 300MHz results in a major increase of throughput in configurations including the use of HP0 except burst lengths of 64B and 128B (see Fig. 8). The distribution between throughput to the different AXIs is much more spread, as HP0 reaches peak performance of ≈3GB/s, while other AXIs in the configuration achieve peaks of up to 200-600 MB/s only (see Fig. 14).

Overall, the higher frequency can translate to higher memory throughput depending on the AXI combinations and memory SODIMM available on the board. However, *even a 300MHz PL clock speed in the best case only gains 55.6% over the best case of 100MHz PL configuration* in our experiments.

D. Access Pattern: Sequential vs Random

We also run the experiments using a random access patterns to identify and measure the loss of performance caused by the rapidly changing memory sections in multi-tenant environment. Note, here the access pattern implies address differences

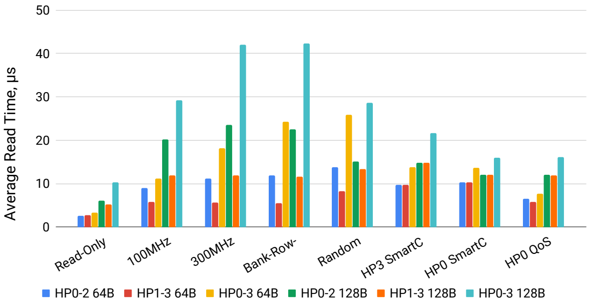

Fig. 10: Average measured read latency in multiple AXI configurations on the Ultra96

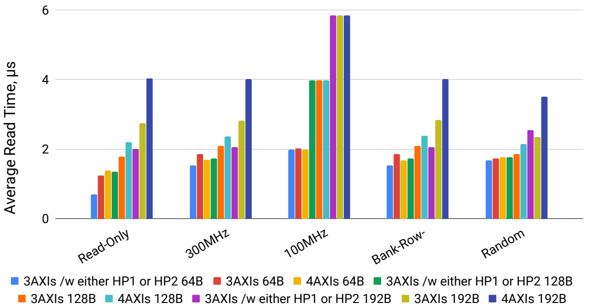

Fig. 11: Average measured read latency in multiple AXI configurations on the ZCU102

between separate burst requests and that the memory accesses inside a burst request is always sequential.

For Ultra96 we observe a decreased throughput of up to 70% for burst sizes of only 16 Bytes and up to 2-10% decrease for large burst sizes compared to the best performing configurations from the base case (see Fig. 12). This is expected, since large burst sizes compensate the associated overhead of switching rows in the DDR memory, while for small bursts, the overhead is too large relative to the work performed. Overall, the effective throughput peaks at bursts multiple of 64B, achieving 3.1GB/s for 128B and about 3.2GB/s for 192B and longer bursts (see Fig. 8).

For the same experiment performed on the ZCU102, we observe less predictable behaviour than Ultra96. Some configurations achieve larger throughput than the same configuration in the base case (see Fig. 9). These points however are local peaks for a particular configuration and all configurations in our random test provide at least 6% less throughput than the peak configuration from the base case.

Overall, *for multi-tenant systems, burst sizes of $\geq$ 512 Bytes minimize the overhead of changing access pattern in the DDR controller*. When using small burst sizes it is recommended to maintain sequential access patterns as much as possible for the highest memory throughput. This also indicates that the DDR controller attempts to perform bulk operations if possible.

E. Memory Organisation: Row-Bank vs Bank-Row

Since FPGAs provide a high degree of flexibility in how we organise and use the hardware, customising the memory architecture along with hardware needs is a possible optimization avenue. Hence, we also ran experiments on Bank-Row-Column address space organisation along with Row-Bank-Column in our DDR memory.[2] To try and utilise this into our advantage, we place the work memory regions of our testing accelerators into different banks, such that each of our accelerators takes two reserved banks - one for read accesses and one for write accesses.

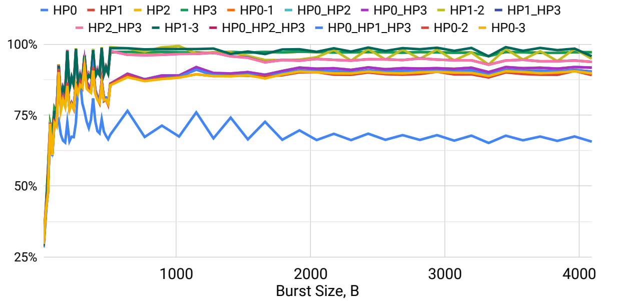

Fig. 12: Throughput of random access pattern on Ultra96 normalised to best results from base case.

In Bank-Row-Column DDR memory configurations on Ultra96 with burst sizes of at least 128 Bytes, compared to Row-Bank-Column configuration we observed a 3-8% increase in throughput in all AXI configurations that exclude HP0 and up to 7% decrease in throughput for the configurations including HP0 (see Fig. 8). A more drastic increase in throughput is observed for very small bursts, most notably 31% throughput increase in all AXI configuration at burst sizes of only 16 Bytes. The latter is expected behaviour since in this configuration, we do not have an overhead for changing loaded rows in the DDR banks between the different small bursts. Otherwise, for large burst sizes, the advantage of this address organisation decreases as larger bursts can inherently overcome some of the overheads associated with row loading in DDR memories.

On the contrary, on ZCU102, running same experiment resulted in a very minor difference in throughput pattern compared to the Row-Bank-Column. Most points in our results are within statistical error of around 0.5% compared to our base test results. The only notable difference in throughput is when using both HP1 and HP2, which yields about 7% decrease in throughput for some burst lengths. These unexpected results on the ZCU102 might be explained by the DDR memory used, which is organised into bank groups in which banks share pre-charge components and our experiment setup used 8 banks that are organised into only 2 bank groups. Read latency does not change with respect to the base case (see Fig. 10 and Fig. 11).

Overall, *changing the address space organisation mostly affects performance at small burst sizes for selected ports (up to 31% improvement) and depends on whether the memory is organised in bank groups or not*. For large burst sizes there is no significant change in throughput behaviour.

F. Multiplexing in PL vs Multiplexing in PS

Xilinx provides an interconnect IP called SmartConnect [10], as a default option to connect one or multiple accelerators to the PS AXI port. It is also used as a system interconnect in designs containing multiple IPs. However, multiplexing in the PL can result in a limited throughput by the PS AXI port: 9.6 GB/s at 300MHz and 3.2 GB/s at 100MHz for a single PS AXI connection.

Given that performance distribution across PS AXI ports is heterogeneous (as shown in Section IV-B) and that SmartConnect is often used as interconnect between accelerators, it is important to identify if there is any performance loss or gain when using additional interconnect layer. To evaluate this, we

[2] By configuring the address map inside DDR settings for Zynq IP core.

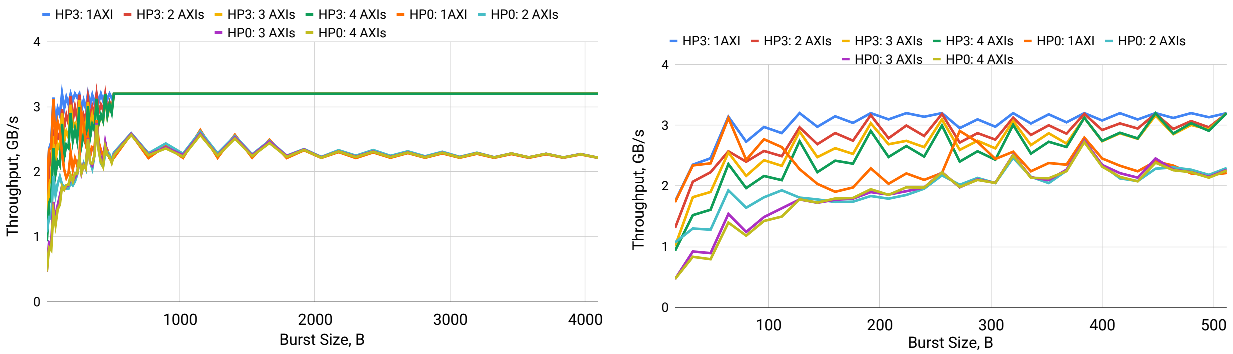

Fig. 13: Measured throughput of 1-4 PL AXI ports connected through Xilinx SmartConnect to either HP0 or HP1 PS AXI connections.

placed the SmartConnect on the Ultra96 between test modules and the physical AXI ports HP0 and HP3 at 100MHz.

The observed results show that using port HP3, we reach maximum connection throughput of 3.2 GB/s at burst sizes of more than 512 Bytes, while the HP0 connection peaks to 2.8 GB/s at 384 Byte bursts and then oscillates to an equilibrium of 2.25 GB/s for very large burst sizes (see Fig. 13). We observed local peaks at burst sizes multiple of 64 bytes. The R/W distribution is the same pattern as standalone HP0 and HP3 experiments, i.e. HP0 prioritises read operations, while HP3 has mostly equal distribution.

In our experiments, SmartConnect distributes the throughput equally between multiple PL AXI ports. We run additional tests and observed that the behaviour of this interconnect module is to interleave operations separately at each read and write ports in a round-robin fashion. This results in an equal distribution if the accelerators use the same burst sizes, but in cases where the burst sizes of two accelerators are different, the resulting throughput is distributed linearly with respect to the burst length. Note that the 128-bit SmartConnect instance with 4 PL AXI ports to 1 PS AXI port from our experiments utilises 13,257 LUTs, 18,044 FFs implemented in 2,514 CLBs. This is a rather small amount of logic for most modern FPGAs, but corresponds to 28.5% of the CLBs on the Ultra96.

Overall, *an additional interconnect layer increases the latency by up to 30% and does not impact the R/W performance distribution and throughput if the burst sizes are equal*. Note, this latency is directly proportional to the burst size.

G. Quality of Service

Upon further examination of the AXI interconnect and DDR subsystem to identify the cause of difference in behaviour across different boards, we found that there are differences in QoS buffer modes and R/W priorities. The mode for QoS can be 1) High Priority (low latency), 2) Best Effort (bulk transfers) and 3) Isochronous (regular, time sensitive, e.g., audio and video traffic) [7]. On ZCU102 all HP AXI ports share same Isochronous mode and memory access priorities on both read and write ports, whereas on Ultra96 AXI HP0 is in Isochronous mode while the other AXI ports (HP1-3) are Best Effort and the write ports are assigned higher priority than read ports. Changing the mode and priorities on Ultra96's AXI HP0 port to the same configuration as the other Ultra96 HP ports, produced results where all AXI ports behave similarly with slightly higher throughput, lowered average read latency, and more fair AXI and R/W distributions (see Fig. 8, Fig. 10 and Fig. 14). Note, despite having same priorities for read and write, in practice, we observed that ZCU102 AXIs and Ultra96's HP0 prioritises read operations, while Ultra96's HP1-3 have balanced R/W ratios. This is due to general DDR memory controller prioritisation of read over write operations. This default prioritisation in the controller can be explained by the nature of most CPU applications, where applications frequently stall for read operations before actual compute. However, one must note that FPGA applications have varying needs for access patterns and read-to-write ratios, as they can utilize Block RAMs as internal buffers and operate in highly parallel or streaming manner.

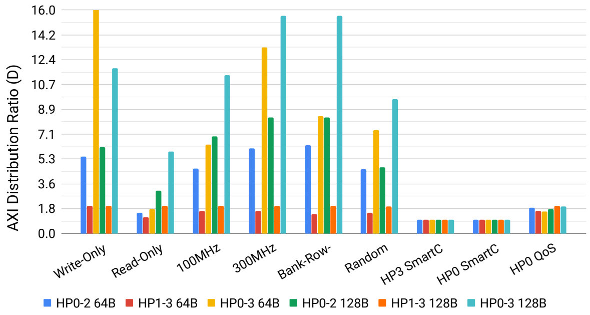

Fig. 14: Performance distribution ratio across AXIs in different configurations on Ultra96. Note, HP0-4 64B has ratio of 5165 and is not captured entirely in the graph.

Hence, when optimising accelerators on Zynq Ultrascale+ boards, *developers must select correct QoS mode and read-write priorities for their application for best performance* (see also our case study in Section V).

H. Performance Distribution

To understand the performance distribution in multi-tenant environments, we measure the AXI distribution ratio D as A_{max}/A_{min}, where A_{max} is the throughput of highest performing AXI port and A_{min} is the throughput of lowest-performing AXI in a configuration. The results of the distribution are shown in Fig. 14. *Most configurations have an uneven distribution which implies that it is very easy to have one accelerator steal all the bandwidth in a multi-tenant environment.* Two configurations standout as possible options for multi-tenancy on Ultra96. 1) PL multiplexing which achieves ideal distribution ($D = 1$) but is subjected to change if accelerators use different burst lengths. 2) Multiplexing in PS with same QoS mode for each AXI ports. This ensures that the default high priority ports do not steal performance. Note that for configurations involving both HP1 and HP2, the ideal distribution is $D = 2$ as they are multiplexed in the PS and that activating DisplayPort or FPD DMA components may affect the performance correspondingly of HP0 and HP3.

V. Application Case-Study

To evaluate the behaviour of a memory system under real application workload, we use Ultra96 and replace the test modules with matrix multiplication accelerators for 4x4 convolution filters. The application is highly memory intensive and mostly read-bound problem due to its higher ratio of read requests.

Our accelerator utilises 4 BRAM36Ks for each of the read memory locations and the write port for minimizing stalling hazards and enabling the use of large burst lengths. We used two accelerators - one connected to HP0 and one connected to HP3. Our first implementation operates by is using a single read burst in round-robin fashion on the read arrays and issues bursts of 128 Bytes. We observe a total of 13.767 Million Convolutions per second (MC/s) between the two accelerators, while the HP0 accelerator achieves a 4.5× higher performance than the HP3 accelerator. The achieved effective throughput is 62% of the DDR peak, which limits acceleration throughput correspondingly. For the second version, we change the accelerators to keep issuing read bursts until they fill the corresponding read buffer before starting to fill another buffer, i.e. more sequential accesses. Rerunning the tests show a performance of 17 MC/s (23% improvement) with 76% effective memory throughput, but similar distribution between HP0 and HP3 accelerators. Since this problem is read-bound, we now change the R/W priority of HP3 to be the same as HP0, thus prioritising read operations. Additionally, we equalise the programmed priorities between HP0 and HP3 within the PS interconnect and change QoS mode of HP0 to Best Effort. This way, the DDR controller should minimize the memory requests polluting each other. With the resulting configuration, we observe an additional increase in performance of 8.6% and a new effective DDR throughput of 83%. Additionally we observed a perfect distribution between the achieved performance of the two accelerators. Moreover, now that memory requests do not delay other ongoing memory requests, we can increase the burst lengths to achieve a better DDR row reusage. When we increase the burst sizes to 4 KiB we observed an additional increase in performance by 9%, reaching 20.1 MC/s. While keeping perfect distribution, our accelerators now manage to reach 90.46% of peak DDR throughput. Note, throughout these experiments we did not change the main compute logic of the accelerator.

Overall, by only performing memory optimizations as discovered in the Section IV, we achieved 46% more MC/s than a standard implementation.

VI. Conclusion

In this paper, we presented a quantitative memory analysis for Zynq UltraScale+ systems using Ultra96 and ZCU102 boards. Our evaluation was based on the synthetic and real world applications with varying parameters such as the number of AXI ports, combinations of AXI ports, burst sizes, frequency, access patterns, address space organisation, multiplexing in PL vs PS, and Quality of Service. Our findings show that 1) 4 out of 6 common assumptions about memory behaviours do not hold and the remaining 2 do only in certain circumstances (see Section II and IV), 2) the achievable peak throughput is 92.5% and 75% of the theoretical DDR throughput for Ultra96 and ZCU102 respectively, and 3) the default memory behaviour across boards as well as the AXI ports of the same type on same board is highly diverse. Hence, it is important for developers to perform the right memory optimizations not only in terms of how the access are performed but also set the right QoS mode and priorities for optimal application performance (as shown by an improvement of 46% just by optimising the memory subsystem at identical cost in Section V). In particular the following general conclusions can be drawn for when working with Zynq UltraScale+ systems:

- Using all AXI HP ports does not always guarantee highest read and write throughput. Some AXI combinations perform better than other.
- The same ARM DDR controller chip and AXI interface on different boards can lead to different memory behaviours based on the pre-programmed QoS and priority settings. It is recommend to set these parameters explicitly for the applications.
- Read operations are prioritised heavily over write operations. While this does not necessarily hurt performance, having read-bound accelerators with write-bound accelerators will put the latter at a major disadvantage.
- On average, 128 and 192 Byte bursts often provide highest throughput.
- Multiplexing the AXI in programmable logic (PL) with AXI SmartConnect IP provides better performance distribution than multiplexing in the ARM SoC at the cost of higher latency and use of FPGA logic.
- Using large burst sizes reduces the throughput overhead of multiplexing AXIs in PL to almost negligible.
- Using higher frequency in programmable logic allows better utilisation of memory throughput but the benefits depend on AXI combinations and memory SODIMM characteristics.
- It is essential to use large burst sizes for accelerators in multi-tenant environment for minimal throughput overhead due to rapidly changing access pattern at the DDR controller.

We released our benchmark suite with tutorial at **blinded.org**[3] in order to allow the research community to exploit our observation for, in many cases, free (in terms of FPGA logic) performance tuning.

References

[1] I. Mavroidis et al., "ECOSCALE: Reconfigurable Computing and Runtime System for Future Exascale Systems," in *DATE*, 2016.

[2] S. Liu et al., "Edge Computing for Autonomous Driving: Opportunities and Challenges," *Proceedings of the IEEE*, pp. 1–20, 2019.

[3] M. Göbel et al., "A Quantitative Analysis of the Memory Architecture of FPGA-SoCs," in *Applied Reconfigurable Computing*, 2017.

[4] M. Sadri et al., "Energy and Performance Exploration of Accelerator Coherency Port Using Xilinx ZYNQ," in *10th FPGAworld*. ACM, 2013.

[5] V. Sklyarov et al., "Analysis and Comparison of Attainable Hardware Acceleration in All Programmable Systems-on-Chip," in *DSD*, 2015.

[6] A. Vaishnav et al., "A Survey on FPGA Virtualization," in *FPL*, 2018.

[7] Xilinx, "Zynq UltraScale+ Device Technical Reference Manual," 2019.

[8] A. Bansal et al., "Evaluating the Memory Subsystem of a Configurable Heterogeneous MPSoC," in *OSPERT*, 2018.

[9] M. Ghasempour et al., "Dream: Dynamic re-arrangement of address mapping to improve the performance of drams," in *MEMSYS '16*, 2016.

[10] Xilinx, "SmartConnect v1.0 LogiCORE IP Product Guide," 2019.

[3] Blinded for review process. It will be updated in camera-ready version.