AFRL-RI-RS-TR-2022-041

SMASH- SOFTWARE DEFINED MEMORY FOR OPTIMIZING APPLICATIONS ON ADVANCED ARCHITECTURES

UNIVERSITY OF SOUTHERN CALIFORNIA

MARCH 2022

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2022-041 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /
BARBARA A. FILMER
Work Unit Manager

/ S /
GREGORY J. HADYNSKI
Assistant Technical Advisor
Computing and Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
MARCH 2022	FINAL TECHNICAL REPORT	SEPTEMBER 2018	SEPTEMBER 2021

4. TITLE AND SUBTITLE

SMASH- SOFTWARE DEFINED MEMORY FOR OPTIMIZING APPLICATIONS ON ADVANCED ARCHITECTURES

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
FA8750-18-2-0034	N/A	63662D
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
		R2L4

6. AUTHOR(S)

Viktor Prasanna and Rajgopal Kannan

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Southern California, Dept. of Contracts and Grants
3720 S Flower Street, Third Floor
Los Angeles CA 90007

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)
Air Force Research Laboratory/RITB 525 Brooks Road Rome NY 13441-4505	RI	AFRL-RI-RS-TR-2022-041

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The focus of this work was to develop a Field Programmable Gate Array (FPGA) based memory controller to support shared memory access in a distributed and heterogeneous environment. The FPGA acts as an intermediate buffer between the accelerators and the shared memory. We used the internal memory of the FPGA to buffer data and reduce the memory access latency. We adopted Direct Memory Access (DMA) techniques and caching methods to optimize the memory accesses. This lightweight and reconfigurable memory controller supports a common Flow Control Unit (FLIT) to communicate with different hardware. We investigated different types of FPGAs from various vendors to identify potential candidates. We considered the number of I/O banks, Dynamic Random-Access Memory (DRAM) technologies, support for Universal Serial Bus (USB), average power consumption, and FPGA on-chip memory while selecting the FPGA. We also evaluated various DRAM technologies and DRAMs to identify possible candidates to use as shared memory. In evaluating the DRAMs to choose from, we mainly considered the support for different FPGA technologies, latency, bandwidth, and power consumption. We developed a performance model to verify that our overall system meets the intended performance requirements. Finally, we developed our memory controller using Verilog Hardware Description Language and Xilinx tools. We simulated the design using Xilinx's memory simulation environment to verify its correctness and performance.

15. SUBJECT TERMS

Software Defined Memory, Field Programmable Gate Array (FPGA), Direct Memory Access (DMA), Signal Processing Accelerators, Flow Control Unit (FLIT), Dynamic Random-Access Memory (DRAM)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
U	U	U	SAR	31

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
BARBARA A. FILMER	NA

PREVIOUS EDITION IS OBSOLETE.

TABLE OF CONTENTS

Section **Page**

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

1.0 SUMMARY

The focus of this work was to develop a Field Programmable Gate Array (FPGA) based memory controller to support shared memory access in a distributed and heterogeneous environment. The FPGA acts as an intermediate buffer between the accelerators and the shared memory. We used the internal memory of the FPGA to buffer data and reduce the memory access latency. We adopted Direct Memory Access (DMA) techniques and caching methods to optimize the memory accesses. This lightweight and re-configurable memory controller supports a common Flow Control Unit (FLIT) to communicate with different hardware.

We investigated different types of FPGAs from various vendors to identify potential candidates. We considered the number of I/O banks, Dynamic Random-Access Memory (DRAM) technologies, support for Universal Serial Bus (USB), average power consumption, and FPGA on-chip memory while selecting the FPGA. We also evaluated various DRAM technologies and DRAMs to identify possible candidates to use as shared memory. In evaluating the DRAMs to choose from, we mainly considered the support for different FPGA technologies, latency, bandwidth, and power consumption.

We developed a performance model to verify that our overall system met the intended performance requirements. Finally, we developed our memory controller using Verilog Hardware Description Language and Xilinx tools. We simulated the design using Xilinx's memory simulation environment to verify its correctness and performance.

2.0 INTRODUCTION

Even with generational improvements in DRAM technology, memory access latency still remains the major bottleneck for application accelerators, primarily due to limitations in memory interface IPs which cannot fully account for variations in target applications, the algorithms used, and accelerator architectures. There have been several techniques proposed to overcome the access latency [1]. Caches [2] are very productive in this regard if the required data fits in the cache and the data have spatial and temporal locality [3]. The ongoing approach for solving long memory access delays is to use onboard Block Random-Access Memory (BRAM) in the FPGA as a data cache and facilitate data retrieval. An alternative solution is to look at multiple memory requests as *bulk memory transfers* [4]. These techniques contribute to reducing the overall latency while maintaining high sustained bandwidth.

Accessing the shared memory of heterogeneous and distributed systems has become a challenge due to the following reasons.

I. Modern workloads require low latency memory accesses.
II. Prioritization of different workloads from different hardware
III. Different hardware (E.g., CPU, GPU, ASICs) has different types of I/Os to communicate with external memory.
IV. High bandwidth is required to satisfy the compute power of each accelerator

To satisfy the above conditions we want intermediate hardware with the following properties.

a) Support multiple communication protocols.
b) Many I/O banks that can run on a high frequency.
c) Fast limited memory to support intermediate caching techniques

Field Programmable Gate Array (FPGA) is an ideal solution that satisfies all the above properties. Here, we propose a multi-faceted approach that helps in obtaining low latency as well as high bandwidth. Our work also motivates to cumulate memory requests over time (i.e., forming a batch) and schedule them while exploring their locality in space before accessing the memory. The parallelism of FPGAs is leveraged to reorder the requests in a batch, faster and more efficiently. In the meantime, the memory controller also aims to fulfill the traditional memory hierarchy by introducing an inbuilt memory routing pipeline that serves to quickly service requests from the connected accelerators. Depending on the external memory configuration and data traffic to the FPGA, the memory controller can be modified to support batches of varying sizes. The programmability of our design allows the controller to be customized based on application requirements and performance objectives. It also facilitates the creation and modification of accelerator-specific control policies.

We develop a unified reconfigurable memory controller template that can be shared among hardware accelerators. The memory controller can be configured depending on the FPGA platform, available resources, and functional requirements of the overall system.

Figure 1 summarizes the conceptual design of our work. In our work, a GOTS accelerator and Xavier work as accelerators with shared memory. The two accelerators are connected to the FPGA using a direct I/O connection and Ethernet, respectively.

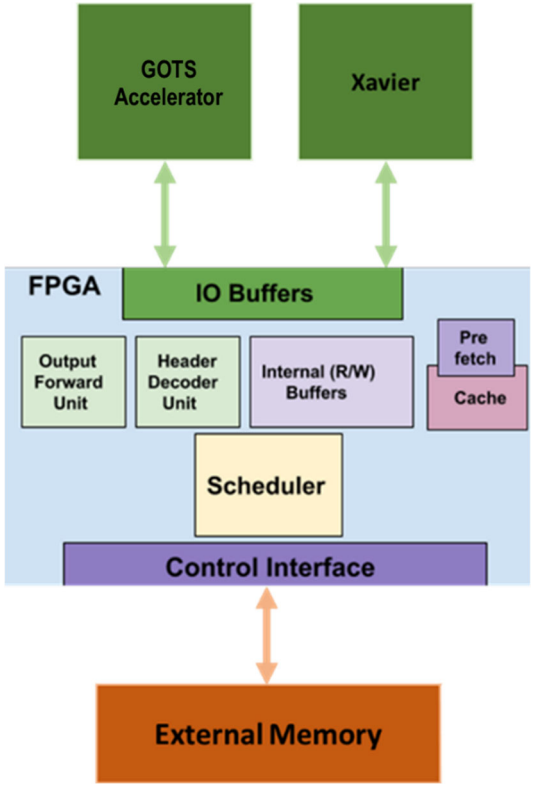

Figure 1. Conceptual design

3.0 METHODS, ASSUMPTIONS, AND PROCEDURE

3.1 Functional Requirements

The functional requirements are listed below.

I. The FPGA should have enough input and output pins to support multiple communication protocols (E.g., Ethernet, USB 3.1)
II. The memory system inside the FPGA should support the cache miss requests and DMA requests that come from accelerators
III. A unified packet structure to communicate with different types of accelerators.
IV. The memory space divide into private and public memory spaces. All the accelerators connect to the FPGA can access the public memory space. Meanwhile, private memory spaces have unique owners, and other accelerators cannot access them.
V. Accelerators can access the memory or communicate with each other through the FPGA. Therefore, FPGA should route the packets accordingly.
VI. FPGA should perform startup tests with memory and other connected devices and verify they work properly.

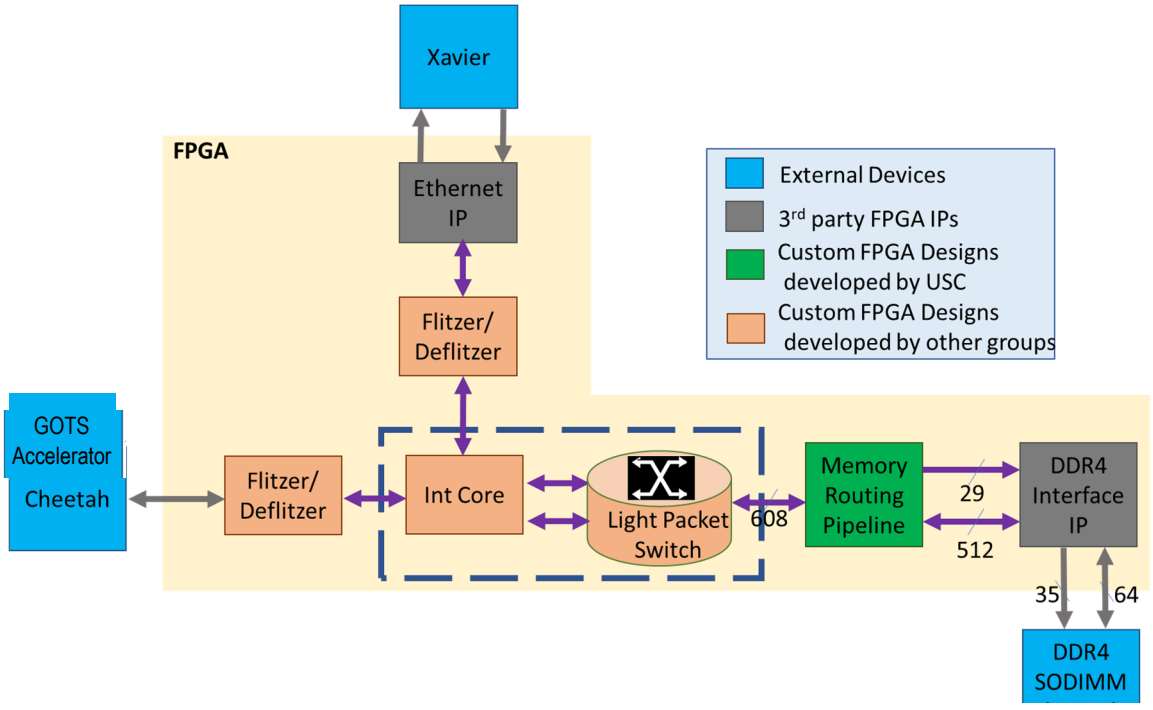

Figure 2. Top level model of FPGA design

The top-level FPGA design to satisfy the above requirements is shown in Figure 2.

3.2 Device Selection

3.2.1 Selection the FPGA. We chose the FPGAs shown in Table 1 for our analysis. Each of the FPGAs selected for analysis had a price less than $5050. Each of the FPGAs represented in this table has an I/O resources count in the 650 range. We referred to Digikey website to create Table 1.

According to third-party USB vendor, Corigine USB, the transceivers provided in the Xilinx FGPG Virtex series 6 (and below) are not fast enough to support USB 3.1, as implementation would typically require approximately 70K – 100K Look Up Tables (LUTs), depending on the function (host vs. device), and the number of endpoints. Table 1 shows a summary of the supportability of each FPGA series to USB 3.1 and DRAM technologies (refer to last 2 columns).

Table 1. FPGAs Selected for Analysis

FPGA	I/O resource count	Unit Price	Logic Element count	Total RAM (MB)	Max clock freq. (MHz)	DRAM tech. used	USB 3.1 B/W
Intel Arria V GX 5AGXBB1D6F40C6G	704	$917	300000	2.06	625	DDR3	5 Gbps
Intel Arria V GX 5AGXBB3D6F40C6G	704	$1,170	362000	2.36	625	DDR3	5 Gbps
Xilinx Kintex Ultrascale XCKU085-1FLVB1760C	676	$3,806	1088325	6.95	630	DDR4	10 Gbps
Xilinx Kintex Ultrascale XCKU095-1FFVB1760C	702	$5,027	1176000	7.21	630	DDR4	10 Gbps
Intel Arria 10 GX 10AX057K4F40E3SG	696	$2,425	570000	5.02	800	DDR4	10 Gbps
Xilinx Kintex Ultrascale XCKU060-1FFVA1517C	624	$2,652	726000	4.6	630	DDR4	10 Gbps

Dual In-line Memory Modules (DIMMs) with higher capacities (>= 32 GB) come in an x72 configuration, which requires about 140 pins. Therefore, DIMM is not a desirable choice for a design with I/O pin constraints.

In the following, we consider 3 designs with different DRAM technologies and various single chip memory capacities as example implementations and evaluate their pin count (estimated) and performance metrics (bandwidth/latency). Here, Latency is dominated by address decoding and Row Address Strobe / Column Address Strobe (RAS/CAS) time. The device options that we found, cost less than $2000 and support more than 650 I/O resources, support DDR3 only (DDR4 and Low-Power Double Data Rate options (LPDDR3 and LPDDR4) are not supported). Choosing different Double Data Rate (DDR) technologies can make a significant impact on the bandwidth and the maximum capacity of a single DRAM chip. The maximum data width and the maximum size of current available DDR3 chips are 16 bits and 2 GB (16 Gb), respectively. Meanwhile, DDR4 supports and 4 GB and 8 GB memory capacities (single chip).

We consider three scenarios. In Scenario 1, we use Xilinx Virtex-6 LXT with DDR3 chips (IS43TR16K01S2A-125KBL). In Scenario 2, we use Xilinx Kintex UltraScale/ Intel Arria 10 (Cost $3800/$2425) with DDR4 (Micron MT40A2G16SKL-062E: B). This analysis provides an insight into the trade-offs in using various DDR technologies. Further, Intel Arria 10 and Xilinx Ultra-scale FPGAs consist of a large BRAM count, which allows implementation of caches and pre-fetching techniques on the FPGA. Finally, in Scenario 3, we use Xilinx Kintex UltraScale with DDR4 (Micron MT40A16G4WPF-062H: B) for the analysis. The lead time of this DDR4 chip is not available online. But using this device, we can reduce the number of chips needed by half (8 chips) comparing to Scenario 2.

3.2.2 Scenario 1: Design 64 GB memory using 16 Gb (1 G x 16) DDR3

Figure 3. Memory design using 32 x 2 GB DRAM chips

- Selected chip: IS43TR16K01S2A-125KBL
- Technology: Synchronous Dynamic Random-Access Memory (SDRAM) – DDR3
- Capacity of a chip: 2 GB
- Data bus width: 16 x 2
- Required number of chips to implement the design: 32

Total pin requirement calculation:

Assuming memory address decoding is done externally,

Total number of address pins	= 36
Total number of data pins	= 32
Other Pins (CKE, CSn, CK, CK#)	= 6 - 12
Total required pins	= 74 – 80
Optional Hardware pins (dqs_c, dqs_t, etc)	= 18 – 24

3.2.3 Scenario 2: Design 64 GB memory using 32 Gb (2 G x 16) DDR4 chips

Figure 4. Memory design using 16 x 4 GB DRAM chips

- Selected chip: Micron MT40A2G16SKL-062E: B (Our suggestions, Lead time: 6 Weeks)
- Technology: SDRAM – DDR4
- Capacity of a chip: 4 GB
- Data bus width: 16 x 2
- Required number of chips to implement the design: 16

Total pin requirement calculation:

Assuming external memory address decoding,

Total number of address pins	= 36
Total number of data pins	= 32
Other Pins (CKE, CSn, CK, CK#)	= 6 - 12
Total required pins	= 74 – 80
Optional Hardware pins (dqs_c, dqs_t, etc)	= 18 – 24

3.2.4 Scenario 3: Design 64 GB memory using 64 Gb (16 G x 4) DDR4 chips

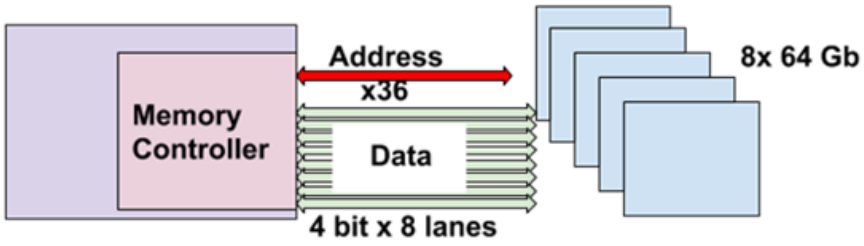

Figure 5. Memory design using 16 x 4 GB DRAM chips

- Selected chip: Micron MT40A16G4WPF-062H: B (Lead time unknown)
- Technology: SDRAM – DDR4
- Capacity of a chip: 8 GB
- Data bus width: 4 x 8
- Required number of chips to implement the design: 8

Total pin requirement calculation:

Total number of address pins	= 36
Total number of data pins	= 32
Other Pins (CKE, CSn, CK, CK#)	= 6 - 12
Total required pins	= 74 – 80
Optional Hardware pins (dqs_c, dqs_t, etc)	= 18 – 24

3.2.5 FPGA power consumption. The actual FPGA power consumption will depend heavily on the resource utilization of the implemented design. However, Xilinx provides "Xilinx Power Estimator (XPE)" to accurately estimate worst-case power consumption.

We chose Xilinx Kintex Ultrascale XCKU085-1F FPGA (One of the suggested FPGAs) for the analysis. Further, we made the following assumptions regarding the design.

- Resource Utilization: LUT - 82.4%, FF – 80.3%, BRAM – 83.3% and DSP – 62.5%
- Clock Frequency: 300 MHz
- Memory Controller: 32-bit LPDDR3 with 5.22 GB/s data rate
- The worst-case total power consumption of such a design is **8.5W**.

Given the same resource utilization, the power consumption of recent FPGA devices such as Xilinx UltraScale is less than the older FPGAs. The main reason behind this is the latest advancements in Application-Specific Integrated Circuit (ASIC) design technology. Xilinx UltraScale FPGAs are made of 20 nm technology, while the rest of the proposed Xilinx FPGAs are made of 45 nm or older technology.

The power consumption of 30W mentioned in the previous report was a generic value obtained from the User Guide of an FPGA Development Board and includes the power consumed by all board peripherals. The proposed board use case scenario is more accurately represented by the XPE evaluation described above.

Considering the budget and the requirements we chose Xilinx Kintex Ultrascale (060/085) as the candidates for this work.

3.2.6 Selecting DRAM. The observations on different DDR technologies are as follows:

- For a given I/O data width, the package size of LPDDR3 chips is smaller than DDR4. In our case, even though the data width of the selected LPDDR3 chip is 4 times greater than the DDR4 chip, the size of the LPDDR3 and DDR4 packages are similar.
- The power consumption of LPDDR3 is less than DDR4 in Active State. Further, LPDDR3 consumes significantly less energy in Idle State. LPDDR3 also has new power-saving strategies such as Deep power-down mode and Active power-down mode.
- The Latency values (Row Hit/Miss Latency, Row Conflict Latency, etc.) of LPDDR3 is higher than the DDR4.

In this project we are focusing on DDR4 and LPDDR3 technologies. A comparison between these 2 technologies is shown in Table 2.

Table 2. Comparison between DDR4 and LPDDR3 memory technologies

Parameter	**DDR4 (MT40A2G16SKL-062E: B) [5]**	**LPDDR3 (MT52L512M64D4GN-107 WT: B) [6]**
Cycle Time (ns)	0.625 ns	1.071 ns
Data Width of a chip	16-bits	64-bits
Operating Voltage (VDD)	1.2 V	1.2 V
Packaging	10.5mm x 13 mm (96-ball FBGA)	14 mm x 14 mm (256 FBGA)
Operating Frequency	1.6 GHz	933 MHz
Row Buffer size of single chip*	256 bits	1024 bits
Maximum Bandwidth (70% efficiency)	4.48 GB/s	10.45 GB/s (64-bit I/O bus width) / 8.96 GB/s (32-bit I/O bus width**)
No. of Dies	2 (Twin Dies)	4
Bank Configuration (No. of rows X No. Columns)	128 Meg X 16	32 Meg X 32
Total Number of Banks	16	32
No. of Channels	1	2
I/O (bits)	32	64
Row hit/miss latency*^ (ns)	16.7 / 30.0 [7][8]	21.6 / 40.3 [7][8]
Minimum row conflict latency# (ns)	43.3 [7][8]	59.1 [7][8]
Typical power consumption in active state [9][10][11] (mW)	~ 323.8	~ 229.46
Typical power consumption in idle state [9][10][11] (mW)	~ 85.5	~ 11.35

* Row Buffer Size per chip = Column width of a bank X # of banks

* FBGA (Fine-Pitch Ball Grid Array): A surface-mount IC packaging type

** We can implement a 32-bit bus between DDR Memory and the FPGA (even though the bit width of the chip is 64) with extra Mux/Demux on the DDR side

*^ Row hit latency: The time takes when the incoming read request data is in the active row buffer

*^ Row miss latency: The time takes when the row buffer is closed (Chip is in Idle state), and read request comes to the chip

Minimum row conflict latency: The time takes when the incoming read/write request data cannot be found in the active row buffer

3.2.7 Analysis of DDR4 and LPDDR3 chips. The LPDDR3 chips that we found with 32-bit data width is in Micron EDFB232A1MA-JD chip series. According to the Micron website, the lead time of the above chips is unknown, and they are under the "Contact Factory" production category. But using the Micron MT52L512M64D4GN-107 module (64-bit data width and lead time of 8 weeks) with Muxer/Demuxer on the DDR side, we can implement a 32-bit bus between DDR Memory and the FPGA.

3.2.8 Latency. Assume the FPGA is working in 300 MHz clock rate. According to Table 2,

The minimum row conflict latency of LPDDR3 = 59.1 ns

The minimum row conflict latency of DDR4 = 43.3 ns

Estimated FPGA clock cycles for logics in memory controller without memory access time = ~ 35 cycles (Included in previous reports)

Number of FPGA clock cycles spent on a row conflict with LPDDR3 = (59.1 ns X 300 MHz) = ~ 18 cycles

Number of FPGA clock cycles spent on a row conflict with DDR4 = (43.3 ns X 300 MHz) = ~ 13 cycles

3.2.9 Bandwidth. Assume that the FPGA logic runs in 300 MHz with a 32-bit memory interface. According to the literature, a typical FPGA Memory Controller works at 70% efficiency.

Expected Memory Bandwidth = data width (bits) × data transfer rate (1/s) × efficiency

Bandwidth using LPDDR3 (16 x 4GB – 32-bit wide data bus) = 2 × 32 bits × 933 MHz × 70% = 5.22 GB/s

Bandwidth using DDR4 (16 x 4GB – 32-bit wide data bus) = 2 × 32 bits × 1.6 GHz MHz × 70% = 8.96 GB/s

The bit width of a selected LPDDR3 chip is 64-bit. If we use 64-bit data I/O pins with LPDDR3, we can achieve a bandwidth of 10.45 GB/s.

3.2.10 Power. Assume that a DRAM chip is in the Active state 80% of the time, and the rest of the time in Idle state. Further, all the chips have uniform access the entire time.

Total power consumption = Active state power X Time period in Active state + Idle state power X Time period in Idle state

Total power consumption while using 16 of LPDDR3 chips = 16 X (229.46 X 0.8 + 11.35 X 0.2) mW = 2.97 W

Total power consumption while using 16 of DDR4 chips = 16 X (323.8 X 0.8 + 85.5 X 0.2) mW = 4.4 W

Typical power consumption of an Ultra Scale FPGA = 30 W

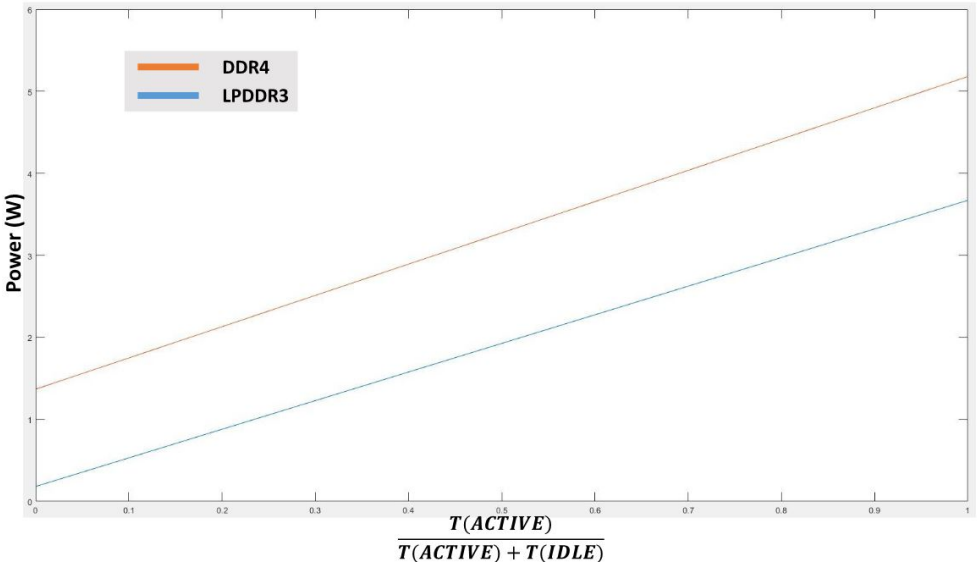

Figure 6. Power Consumption - LPDDR3

3.2.11 DRAM Selection. Depending on the budget and device compatibility, we chose DDR4 technology for our implementation. We found several So-DIMMS which are compatible with Xilinx Kintex devices (see Table 3). The internal structure of DDR4 SO-DIMMs are shown in figure 7.

Table 3. DDR4 SO-DIMM choices

Manufacturer	Model	Size
Kingston	HX429S17IB/32	32GB
Kingston	HX426S16IB/32	32GB
Axiom	APL2666SB32-AX	32GB
OWC	OWC2666DDR4S64P	32GB
Crucial	CT32G4SFD832A	32GB

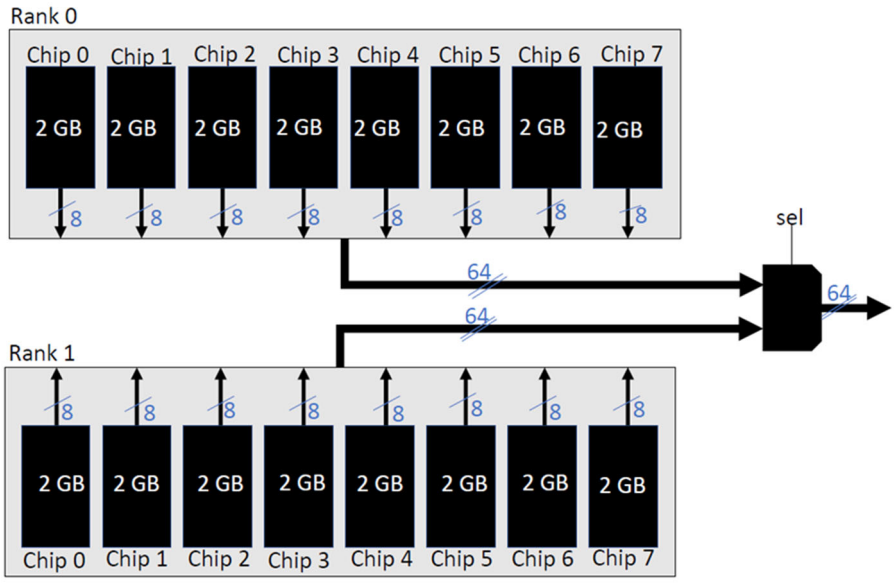

Figure 7. Internal structure of selected DDR4 SO-DIMMs

3.3 Performance Modeling and Analysis

In this section we compute maximum achievable bandwidths of the Small Outline Dual In-line Memory Module (SODIMM) and FPGA interface. We also consider the following example scenarios.

Case 1: 16 KB DMA read transfers

Case 2.1: 64B cache line restore (read)

Case 2.2: cache line store (write)

The data paths are shown in Figure 8.

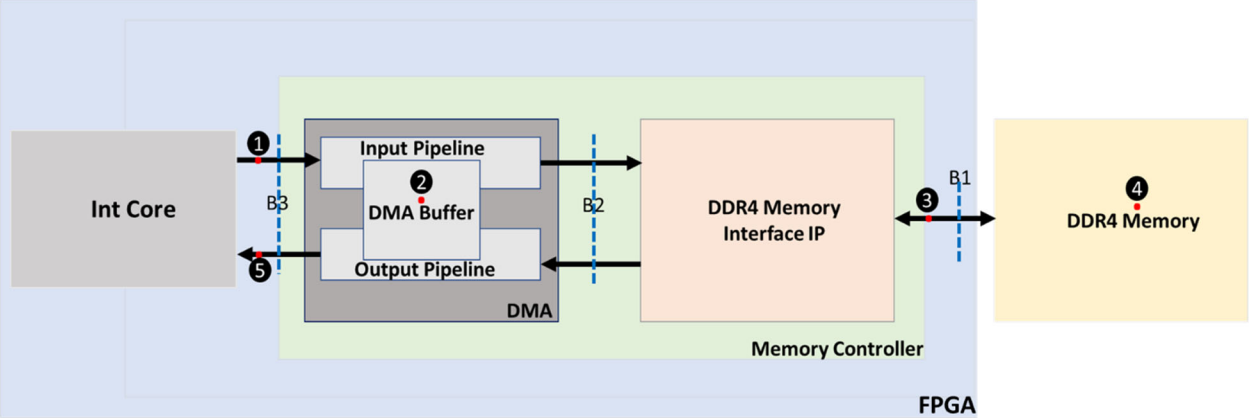

Figure 8. Conceptual design of the FPGA memory controller

3.3.1 Peak bandwidth of SODIMM (B1)

Bandwidth impact factor due to SODIMM to Memory interface IP latency

= SODIMM to Memory interface IP latency (per access) x FPGA frequency

= 25.6 ns x 0.2 GHz = 6

Peak bandwidth between SODIMM and memory interface (B1) = (2400 Mb/s x 64)/6 = 3.2 GB/s

3.3.2 Peak bandwidth of DRAM interface IP (B1)

Peak bandwidth from memory interface to memory controller (B2)

= min (DRAM interface width x FPGA clock rate, B1)

= min (64B x 0.2 GHz, 3.2 GB/s) = 3.2 GB/s

3.3.3. Example Scenarios

Case 1: 16 KB DMA read transfer

Memory controller design supports up to 16 KB DMA transfers. Therefore, we select 16 KB (maximum transfer size of DMA) to analyze DMA transfers. We assume the cache line pipeline is empty in case 1.

Time taken from the input interface to the DMA (from 1 to 2) = 4 cycles // see Figure 9

Time taken from DMA to memory (from 2 to 3) = 3 cycles

Time taken to access the DRAM memory

= first access latency + (no. subsequent accesses) x subsequent access latency

= 8 + 6 x (16K/64 -1) = 1538 cycles

Time taken to store all 16 KB requests from memory on DMA (from 3 to 4 and back to 2)

= 1538 + 4 = 1542 cycles

Time taken from DMA to output interface (from 2 to 5) = 3 + 16K/64 -1= 258 cycles

Total access time = 4 + 3 + 1542 + 258 = 1807 cycles

Effective bandwidth to read 16 KB (B3) = min (memory controller to Int-core output width x operating frequency, (total data)/ (no. cycles))

= min (64B x 0.2 GHz, (16KB/1807) x 0.2 GHz) = **1.81 GB/s**

Case 2.1: 64B cache line restore (read) – best case

It also supports cache line transfers, with each FLIT having a 64 B payload. So, we select 64 B to analyze cache line transfers. We assume that there are no queue delays due to other requests, and the DMA pipeline is empty in case 2.1 and case 2.2.

Time taken from input interface to the memory = 5 cycles

Time taken from memory to input interface (return path) = 7 cycles

Time taken to access the DRAM memory = 8 cycles

Total round-trip time taken by a cache line read request to return the data from memory to input interface (no queue delays due to other requests) = 5 + 7 + 8 = 20 cycles

Case 2.2: 64B cache line store (write) – best case

Time taken from input interface to the memory = 5 cycles

Time taken to access the DRAM memory = 8 cycles

Total time taken by a cache line write request from input interface to memory (no queue delays due to other requests) = 5 + 8 = 13 cycles

3.4 Implementation

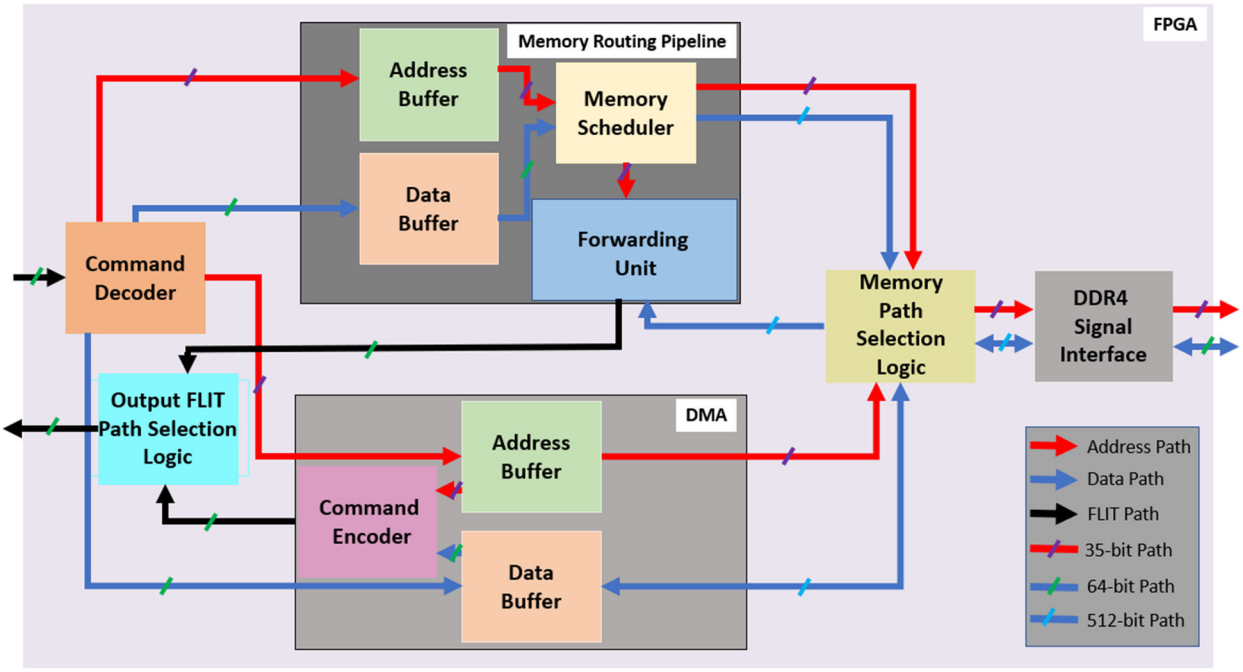

Figure 9. Overview of the memory controller

As the figure suggests, there are 2 main components that reside in the Memory Controller V0.5. They are,

1. Memory Routing Pipeline (MRP)
2. DMA

When a new FLIT reaches the Command Decoder, depending on the type of message, it is routed to either MRP or DMA. If a FLIT belongs to the DMA type, it is forwarded to the DMA; otherwise, the FLIT is forwarded to the MRP.

FLITs routed to the MRP are assumed to have highest priority; FLITs that reach the DMA will be processed less urgently. Therefore, DMA transactions occur whenever the Memory Path Selection module is idle. However, the DMA module has an internal timer. Whenever a DMA transaction cannot be completed within the time limit, the DMA transaction becomes the highest priority, stalling the MRP path. The timer length is dependent on the expected maximum latency for a DMA transaction.

3.4.1 DMA Engine. The DMA engine processes bulk transfers coming from the accelerators. A DMA engine can have several DMA buffers inside. Therefore, it can support several bulk transfer requests in parallel. The number of DMA buffers is a reconfigurable parameter. A DMA request can contain one or more FLITs. When the first FLIT of a bulk transfer reaches the DMA engine, it passes onto one of the DMA buffer controllers. The corresponding DMA controller updates its status registers to *occupy* and shares the PE ID of origin with the DMA Request Mapper. When

another FLIT with the same *ID* comes to the memory controller, it routes to the same DMA buffer. DMA buffer controllers wait until all the FLITs of the bulk transfer are available before starting the external memory access. Figure 10 shows the overall architecture of the DMA engine.

The main advantages of having a DMA engine are:

a) DMA requests can request more than one element at once, unlike the memory routing pipeline, and reduce the input traffic of the memory controller.
b) Using a DMA engine to access data with less spatial or temporal locality prevents cache pollution.
c) DMA transfers can utilize the memory bandwidth better for bulk transfers.

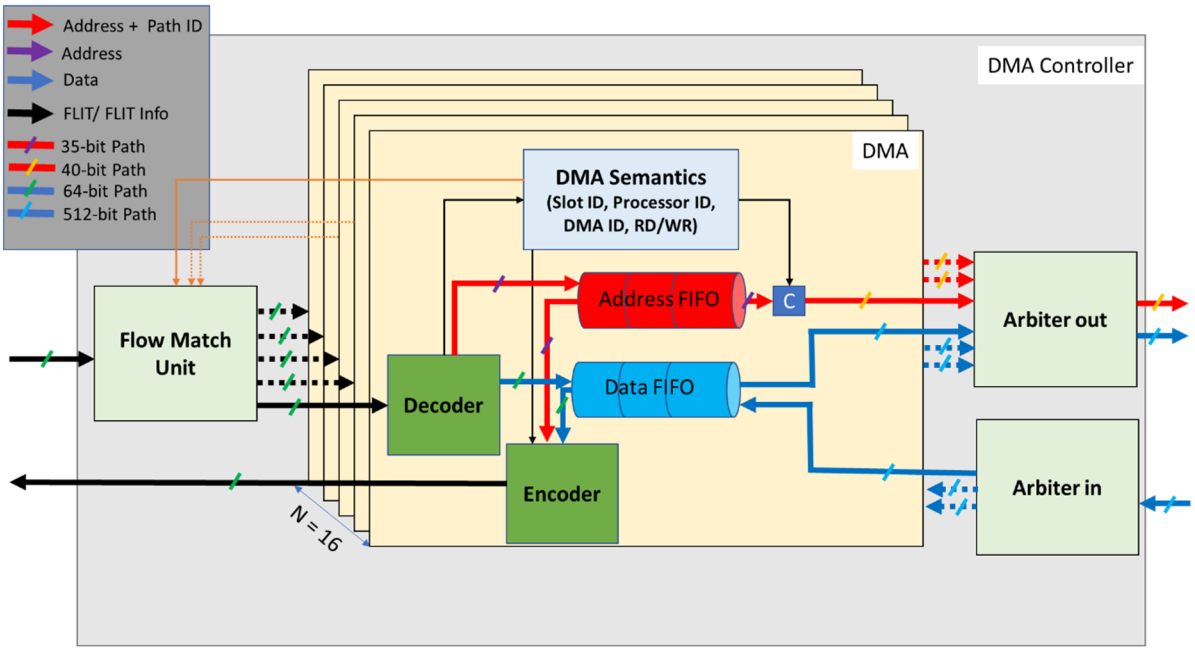

Figure 10. Overall architecture of DMA engine

3.4.1 Memory Routing Pipeline. We implemented the Memory Routing Pipeline of the proposed memory controller using Verilog Hardware Description Language. As shown in Figure 11, the initial layout consists of an instruction decoder, a data forwarding unit, a simple first-in-first-out memory scheduler, and a DDR4 memory interface.

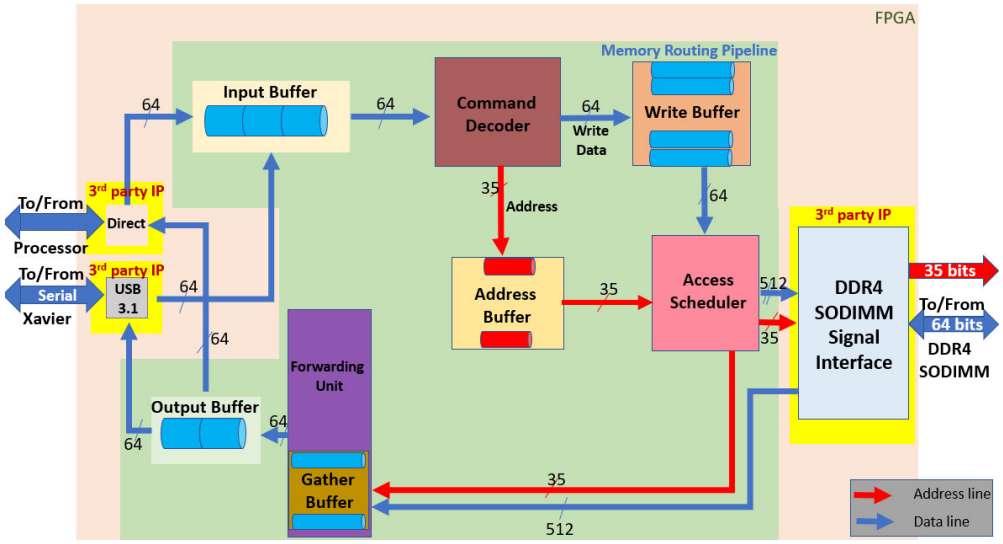

Figure 11. Memory Routing Pipeline

3.4.1 Memory signal interface. The memory signal interface IP handles communication between FPGA and DRAM Controller Circuitry while receiving memory read and write requests from the internal logic (memory routing pipeline). The Xilinx IP catalogue contains a memory signal interface IP for LPDDR3 SDRAMs. The basic IP is freely available to the University of Southern California (USC) team through the Xilinx academic program. It has a low I/O pin count, thereby satisfying memory signalling requirements. Figure 12 shows the complete signalling interface.

The signal interface IP consists of three main components.

1. User Interface Block - Communicates with the FPGA internal logic while using FIFO interfaces for input data, output data, and address.
2. DRAM State Handler - Generates low-level SDRAM signals such as pre-charge, row open, row close, refresh, read request, write request, etc. We can eliminate external DRAM Controller Circuitry because the DRAM State Handler within the IP generates all the low-level signals to control the DRAM chips.
3. Physical Layer - Connects with the physical layer of SDRAMs. The I/O pins required to interact with SDRAMs are shown in Figure 12. 120 FPGA I/O pins are required to support a total of 32 GB memory with a 64-bit data bus.

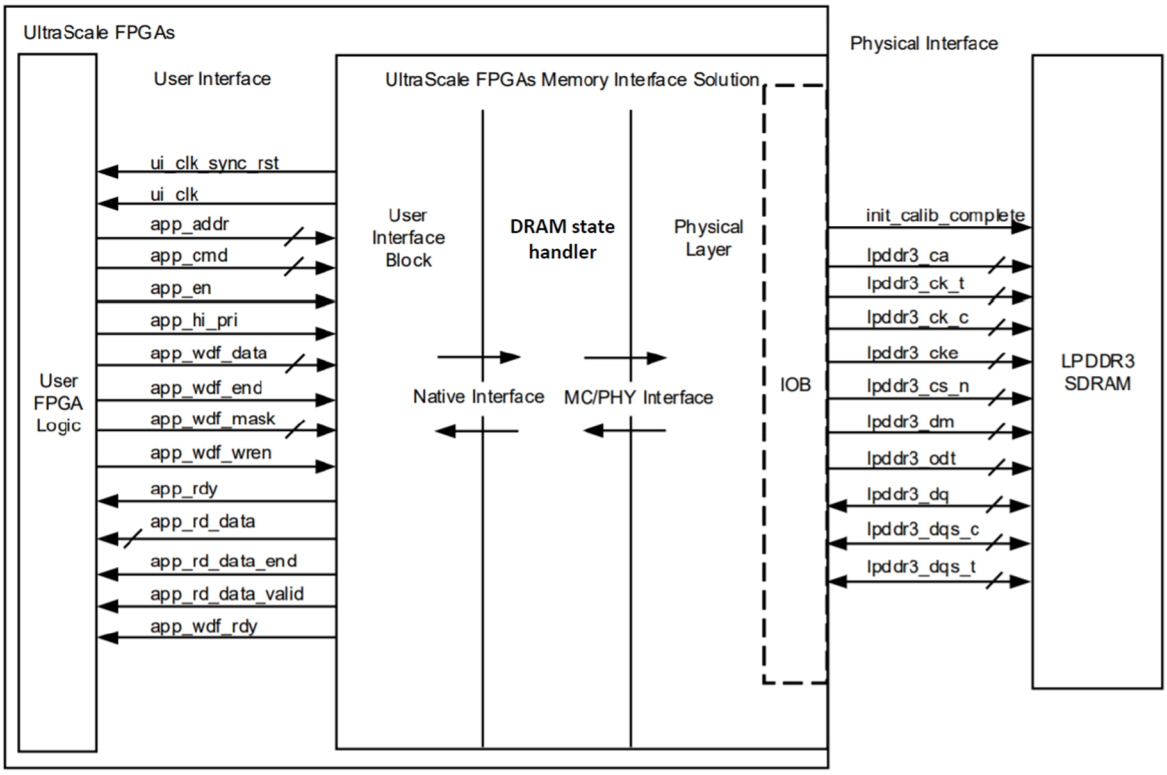

Figure 12. Xilinx UltraScale Architecture-Based FPGA Memory Signal Interface Core

4.0 RESULTS AND DISCUSSION

4.1 Results

We used Xilinx default DDR4 simulation environment to simulate our design with an external DDR4 chip. The Xilinx default simulation environment provides 8 GB DDR4 chip (MTA8ATF1G64HZ-2G3) with an operating frequency of 1.60 GHz. We simulated our FPGA design while maintaining a FPGA clock frequency of 200 MHz. Table 4 shows a summary of resource utilization. The total power consumption of the design is around 2.2 W. A detailed analysis of power consumption is shown in Figure 13. Further, performance values of the design in simulation are shown in the Table 5 as well.

Table 4. Resource utilization of memory controller

Resource	Consumption
LUTs	4%
Registers	6%
BRAM	10%

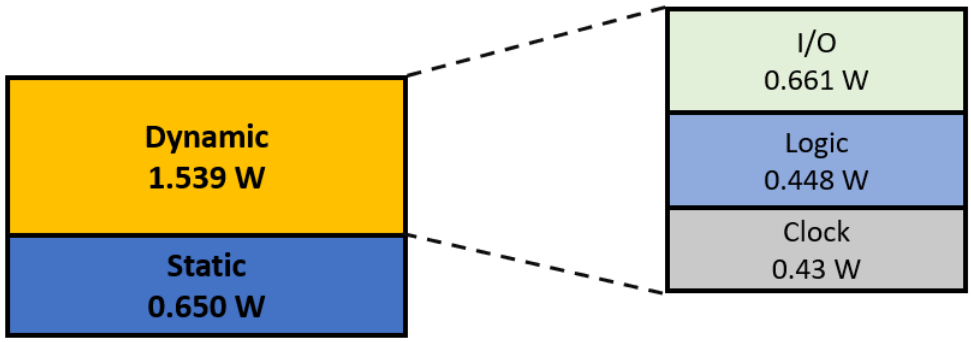

Figure 13. Power consumption

Table 5. Performance of the memory controller

Maximum bandwidth achieved	3.0 GB/s
Maximum Clock Frequency	200 MHz
Average time to process read request	Worst: 72 Cycles Best: 26 Cycles
Average time to process write request	Worst: 54 Cycles Best: 8 Cycles

4.2 Deliverables

We delivered a completed Verilog code the memory controller. It consists of the Memory Routing Pipeline and DMA Engine as described in section 3.4. It supports the FLIT structure designed by the Air Force Research Laboratory (AFRL) / Computing & Communications Division (RIT).

5.0 CONCLUSION

In this work, we developed a multifaceted FPGA-based memory controller with shared memory. The main memory is shared with multiple accelerators connected to the FPGA. Since the design is lightweight, the FPGA can be shared with computation tasks as well. The memory controller can be configured depending on the FPGA platform, available resources, and functional requirements of the overall system.

6.0 REFERENCES

[1] Sally A. McKee, "Reflections on the memory wall," *In Proceedings of the 1st conference on Computing frontiers*, New York, NY, USA, (2004).

[2] A. D. S. Gil, J. I. B. Benitez, M. H. Calvino and E. H. Gómez, "Reconfigurable Cache Implemented on an FPGA," *2010 International Conference on Reconfigurable Computing and FPGAs*, (2010).

[3] Monica D. Lam, Edward E. Rothberg, and Michael E. Wolf, "The cache performance and optimizations of blocked algorithms", *SIGPLAN Not*es, Vol 26, No. 4, pp 63–74, 1991.

[4] Xiaoyu Ma, Dan Zhang, and Derek Chiou, "FPGA-Accelerated Transactional Execution of Graph Workloads," *In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '17)*, New York, NY, USA, (2017).

[5] Micron, "Specification of MT40A2G16SKL-062E: B device", URL: https://www.micron.com/products/dram/ddr4-sdram/part-catalog/mt40a2g16skl-062e, last modifiend 2021. accessed September 7, 2021.

[6] Micron, "Specification of MT52L512M64D4GN-107 WT device", URL: https://www.micron.com/products/dram/lpdram/part-catalog/mt52l512m64d4gn-107-wt, last modifiend 2021. accessed September 7, 2021.

[7] I. Bhati, M. Chang, Z. Chishti, S. Lu and B. Jacob, "DRAM Refresh Mechanisms, Penalties, and Trade-Offs," *in IEEE Transactions on Computers*, vol. 65, no. 1, pp. 108-121, 1 Jan. 2016.

[8] Saugata Ghose, Tianshi Li, Nastaran Hajinazar, Damla Senol Cali and Onur Mutlu, "Understanding the Interactions of Workloads and DRAM Types: A Comprehensive Experimental Study", *CoRR*, Vol abs/1902.07609, 2019.

[9] Micron, "DDR4 power estimates", URL: https://www.micron.com/-/media/client/global/documents/products/technical-note/dram/tn4007_ddr4_power_calculation.pdf, last modified 2020. accessed August 12, 2020.

[10] Micron, "DDR3 power estimates", URL: https://www.micron.com/-/media/Documents/Products/Technical%20Note/DRAM/TN41_01DDR3_Power.Pdf, last modified 2020. accessed August 12, 2020.

[11] Micron, "LPDDR3 vs DDR3 power estimates", URL: https://blogs.synopsys.com/committedtomemory/2014/01/10/when-is-lpddr3-not-lpddr3-when-its-ddr3l, last modified 2020. accessed August 12, 2020.

[12] Xilinx, "Xilinx UltraScale Memory Interface IP", URL: https://www.xilinx.com/support/documentation/ip_documentation/ultrascale_memory_ip/v1_4/pg150-ultrascale-memory-ip.pdf, last modified 2021. accessed September 7, 2021.

7.0 LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

Acronym/Abbreviation	Meaning
ASIC	Application-Specific Integrated Circuit
AFRL	Air Force Research Laboratory
BRAM	Block Random-Access Memory
CK	Clock
CKE	Clock Enable
CPU	Central Processing Unit
CSn	Chip Select – n referring to the chip #
DDR	Double Data Rate
DDR#	Double Data Rate #
DIMM	Dual In-line Memory Module
DMA	Direct Memory Access
DRAM	Dynamic Random-Access Memory
FBGA	Fine-Pitch Ball Grid Array
FIFO	First In First Out
FLIT	Flow Control Unit
FPGA	Field Programmable Gate Array
GB	Gigabyte
GOTS	Government Off-The-Shelf
GPU	Graphics Processing Unit
I/O	Input/Output
IP	Internet Protocol
KB	Kilobyte
LPDDR#	Low-Power Double Data Rate #
LUT	Look Up Table
MRP	Memory Routing Pipeline
RAM	Random-Access Memory
RAS/CAS	Row Address Strobe / Column Address Strobe
SDRAM	Synchronous Dynamic Random-Access Memory
SODIMM	Small Outline Dual In-line Memory Module
USB	Universal Serial Bus
USC	University of Southern California
XPE	Xilinx Power Estimator