ELEKTROTEHNIŠKI VESTNIK 89(5): 239–245, 2022
ORIGINAL SCIENTIFIC PAPER

# A memory interface for DDR3 SDRAM memory in Xilinx 7 Series FPGAs

**Jari Beguš[1,†], Andrej Žemva[2], Damjan Zadnik[3]**

[1]*Jari Beguš, 1000 Ljubljana, Slovenia*
[2]*Laboratory for Integrated Circuit Design, Faculty of Electrical Engineering, University of Ljubljana, 1000 Ljubljana, Slovenia*
[3]*Iretec d.o.o., 4000 Kranj, Slovenia*

† *E-mail: jari.begus@gmail.com*

**Abstract.** For FPGAs to communicate with external memory, a memory interface acting as a two-way converter between the memory signals and the FPGA's internal logic is needed. For this purpose, a custom DDR3 SDRAM interface for Xilinx 7 Series devices is developed in the Verilog HDL. The interface sequential access speed is tested at memory frequencies of 125 MHz and 325 MHz on a Digilent Arty S7-50 development board which houses a Xilinx Spartan 7 FPGA and a 2 Gbit x16 DDR3L SDRAM chip. The interface's FPGA utilization and performance are compared with two existing DDR3 SDRAM solutions for the 7 Series platform.

**Key words:** FPGA, DDR3 SDRAM, memory interface, memory controller, PHY

### Pomnilniški vmesnik za pomnilnik DDR3 SDRAM v vezju FPGA Xilinx serije 7

Za komunikacijo z zunanjim pomnilnikom je v vezju FPGA potrebna implementacija pomnilniškega vmesnika, tj. vmesne plasti, ki deluje kot dvosmerni pretvornik med signali pomnilniškega čipa in aplikacijo znotraj čipa FPGA. S tem namenom je bil v strojno opisnem jeziku Verilog razvit pomnilniški vmesnik za pomnilnik DDR3 SDRAM za čipe FPGA iz serije 7 proizvajalca Xilinx. Zaporedni dostop do pomnilnika je bil preizkušen pri frekvencah 125 MHz in 325 MHz na razvojni plošči Digilent Arty S7-50, ki vsebuje čip FPGA Spartan 7 in pomnilniški čip DDR3L x16 s kapaciteto 2 Gbit. Vmesnik je bil po zasedenosti vezja FPGA in hitrosti prenosa podatkov primerjan z dvema že obstoječima pomnilniškima vmesnikoma DDR3 za čipe FPGA Xilinx serije 7.

## 1 INTRODUCTION

Embedded systems based on FPGAs enable a large degree of parallelization which accelerates data processing. To store data values before, during, and after processing, such systems require random access memory. Modern FPGA devices may include memory elements, such as BRAM and LUTRAM, but the capacity of the integrated memory is severely limited even in the most expensive FPGA models.

In applications where the integrated FPGA memory capacity is insufficient, external memory modules in the form of SRAM, DRAM, or SDRAM are used. With DDR SDRAM, data is transferred at both the rising and the falling clock edges. Due to its high memory capacity and low price, DDR SDRAM is a common choice in computer and embedded systems.

To communicate with external memory, such as DDR SDRAM, integrated circuits need a memory interface to act as a buffer between the fast memory signals and the slower logic signals within the IC, be it an ASIC, microcontroller, or FPGA. The reconfigurable nature of the latter enables the implementation of different memory interfaces.

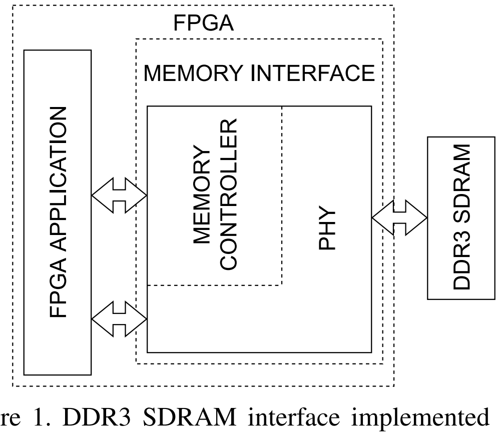


Figure 1. DDR3 SDRAM interface implemented in an FPGA.

Typically, such interfaces are composed of two parts (see Figure 1)[1]. The physical layer (PHY) uses the FPGA's input-output blocks (IOBs) and serializer-deserializer (SERDES) primitives to transfer data between the fast memory clock domain, which determines the data transfer rate, and the domain of the slow clock,

*Received 15 November 2022*
*Accepted 1 December 2022*

which drives the logic elements within the FPGA. The memory controller, operating in the slow clock domain, guarantees the correct values of the command, data, and address buses.

Xilinx, Inc. offers two memory interfaces for use with its 7 Series FPGAs. One is free, but its large FPGA utilization limits it to large FPGA chips. The other has lower FPGA utilization, but is not affordable. The goal of this work is to develop a third option that would enable similar transfer speeds as the established solutions while utilizing a smaller portion of the FPGA.

## 2 DDR3 SDRAM

In this work, a DDR3L chip is used. The suffix "L" indicates a low-voltage variant which can operate at either 1.35 V or 1.5 V, whereas standard DDR3 memory operates only at 1.5 V. The use of DDR3 and DDR3L memory is described in the JEDEC JESD79-3 standard [2].

The memory array architecture of dynamic memory has seen little change in the past decades. Each memory cell stores one bit of information within a capacitor whose high or low voltage corresponds to a logic 1 or 0.

Because the capacitor is not ideal, its electric charge is lost over time. To prevent data loss, dynamic memory cells are refreshed periodically. A refresh circuit samples the data in the capacitor before re-writing the same data into the same memory cell.

To read data from a dynamic memory cell, the low-voltage value stored in the capacitor is amplified by a sense amplifier before the data can be made available on the external data bus. The latter operates at 1.35 V or 1.5 V in the case of DDR3L memory. The sense amplifier in DDR3 SDRAM also serves as the refresh circuit [3].

Individual memory cells are grouped into columns, which form rows, which further form memory banks. While each DDR3 SDRAM chip is made up of eight banks, the number of rows and columns can vary depending on the data bus width and memory capacity. A 2 Gbit x16 chip, as used in this work, is composed of 16384 rows, where each row is composed of 1024 columns, each 16 bits wide. The external data bus width equals the width of individual columns. A simplified diagram of the internal memory structure and internal address buses is shown in Figure 2.

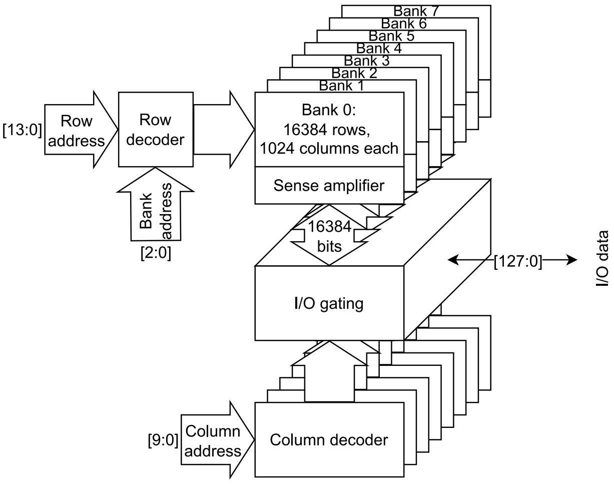


Figure 2. Structure of a 2 Gbit x16 DDR3 chip with $2^3$ banks, each consisting of $2^{14}$ rows. Each row is composed of $2^{10}$ columns.

The input differential clock signal (CK, CK#) is generated by a memory interface. JESD79-3 specifies a memory clock frequency in the range of 300 to 1066 MHz, the latter corresponding to a maximum data throughput of 2133 MT/s[a]. The frequency of the data bus signals requires that the connections between the FPGA and DDR3 SDRAM be regarded as transmission lines. To minimize reflections on the data bus, the impedances of the data source, PCB trace, and data sink must be properly matched. Accordingly, configurable on-device termination (ODT), a new feature in DDR3 SDRAM, allows the input and output impedance of the data bus to be adjusted as divisors of $R_{ZQ} = 240\,\Omega$ within the memory chip itself.

The data bus is comprised of the data lines (DQ), differential data strobe lines (DQS, DQS#) and data mask lines (DM). Due to the high data rates, the flight time of the signal may not be negligible with regard to the duration of the time when a data bit is made available on the transmission line connecting the data source and data sink. If the data read from the memory chip is in phase with the clock at the data source (the memory chip), there is no guarantee that the phase relationship will not change by the time it reaches the data sink (the memory interface). This problem is solved by source synchronous interfaces.

The solution, as implemented in DDR SDRAM, is the DQS strobe signal. Its rising and falling edges are used to trigger the sampling of the DQ line by the data sink.

The read data is in phase with the strobe signal, not with the clock signal. As the strobe signal is used to sample the data on the data lines, neither DQ nor DQS are expected to be in phase with the clock signal when the read data reaches the memory interface.

Because the DQ and DQS lines are in phase during reads, it is up to the memory interface to sample the DQ lines when the DQ data eye is most open. To achieve this, the DQS edges must be delayed, as demonstrated in Figure 3.

Conversely, when data is written to DDR SDRAM, the DQS and DQ lines are not in phase. The data source

[a] JESD79-3 also specifies an optional "DLL off" mode wherein DDR3 SDRAM may be run at frequencies of 125 MHz and lower. Manufacturers are not required to implement this feature.

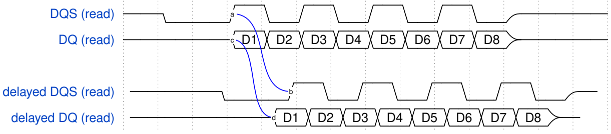


Figure 3. DQS line edges delayed into the center of the DQ data eye when data is read from DDR SDRAM.

(the memory interface) is expected to align the data lines and the data strobe lines so that the strobe edges are placed in the center of the data line's data eye. Normally, this would require an output delay of the DQ and DQS signals to be controlled granularly. Many (low-speed) memory interfaces do not implement such output delay circuitry. Instead, a common solution is to assume that a 90° phase relationship between the DQ and DQS lines will suffice [4].

The command bus pin names originate from the first DDR SDRAM specification, published in JESD79, but have lost their original meanings over time. The CS# (chip select) line enables the memory chip. The RAS# (row address strobe) line activates (ACT) a row within a chosen bank. The WE# (write enable) line selects whether a read (RD) or write (WR) operation will take place whenever the CAS# (column address strobe) line is used to begin the selected operation on the chosen column.

The precharge command (PRE) closes the banks that were previously activated by the ACT command. The memory interface uses the mode register set command (MRS) to configure the memory chip. During the initialization sequence, the ODT circuit must be calibrated using the ZQ calibration command (ZQCL). The no-operation command (NOP) is analogous to deselecting the memory chip using the CS# line.

The bank address bus (BA[2:0]) selects the memory bank. The address bus (A[13:0] in 2 Gbit x16 DDR3 SDRAM) specifies the memory row during the ACT commands and the memory column during the RD or WR commands.

The above commands are listed in Table 1.

Table 1. Truth table of the commands used in the presented work.

| Symbol | CS# | RAS# | CAS# | WE# | BA[2:0] | A10 | A[13:11],<br>A[9:0] |
|---|---|---|---|---|---|---|---|
| MRS | L | L | L | L | MR | Setting | |
| REF | L | L | L | H | X | X | X |
| PRE | L | L | H | L | Bank | X | X |
| ACT | L | L | H | H | Bank | Row | |
| WR | L | H | L | L | Bank | L | Column |
| RD | L | H | L | H | Bank | L | Column |
| NOP | L | H | H | H | X | X | X |
| DES | H | X | X | X | X | X | X |
| ZQCL | L | H | H | L | X | H | X |

Internally, DDR SDRAM functions as a finite-state machine. Issuing commands over the command bus triggers state transitions. The durations of these transitions are accurately defined in memory chip datasheets. These timing constraints may be expressed in units of time (ns or ps) or in periods of the memory clock (expressed as CK or nCK).

Table 2 lists the timing constraints between individual commands implemented in the presented work and given in device datasheets and the JEDEC specification.

Table 2. DDR SDRAM state transitions are not instant. The above symbols represent the time that must pass after a state transition before a new command may be issued. The symbols are a subset of those used in JESD79-3 and in individual memory part datasheets.

| Current state | Next state | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | MRS | PRE | ACT | RD | WR | REF | ZQCL | IDLE |
| CKE | $t_{XPR}$ | | | | | | | |
| MRS | $t_{MRD}$ | | | | | | $t_{MOD}$ | |
| ZQCL | | | | | | $t_{ZQINIT}$ | | |
| PRE | | | | | | | | $t_{RP}$ |
| ACT | | | | $t_{RCD}$ | $t_{RCD}$ | | | |
| RD | | $t_{RTP}$ | | $t_{CCD}$ | | | | |
| WR | | $t_{WR}$ | | | $t_{CCD}$ | | | |
| REF | | | $t_{RFC}$ | | | $t_{RFC}$ | | |

## 3 FPGA

FPGAs allow for modular implementations of combinational and sequential logic circuits. Xilinx 7 Series FPGAs are composed of configurable logic blocks (CLBs) where each CLB is composed of two logic slices. Four look-up tables (LUTs), eight flip-flops (FFs), and the accompanying multiplexers and arithmetic carry logic form a slice [5].

The XC7S50-CSGA324 FPGA from the Xilinx Spartan 7 family of devices, used in this work, includes 8150 slices for a total of 32600 LUTs and 65200 FFs.

The FPGA includes purpose-specific circuits, known as primitives, such as the mixed-mode clock manager (MMCM), which combines a PLL to generate clock signals with variable frequencies and a digital clock manager (DCM) which acts as a DLL to control the phase relationships between the generated clock signals [6].

The FPGA internal structure, or the FPGA fabric, shown in Figure 4, also includes input-output blocks (IOBs). The IOB primitives used in the developed memory interface include input-output tristate buffers, which can be either single-ended (IOBUF) or differential (IOBUFDS), as well as output buffers, which may also be single-ended (OBUF) or differential (OBUFDS). Also used are the input delay primitives (IDELAY) and serial-to-parallel (ISERDES) and parallel-to-serial converters (OSERDES).

When FPGAs are used to communicate with other devices, it may be difficult for the logic part of the interfaces to reach timing closure at the frequencies at which

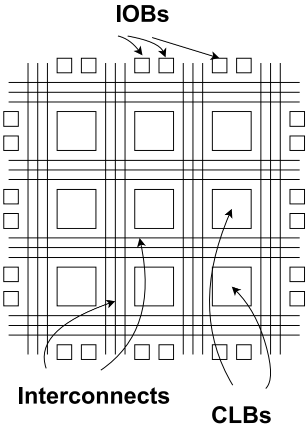


Figure 4. Basic FPGA fabric.

external data buses operate. This is why transceiver physical layers often rely on SERDES elements. These transfer data between the fast clock domain of the data bus and the slow clock domain of the internal logic. In the case of 7 Series FPGAs, SERDES primitives may be configured in a double data rate (DDR) mode such that the parallel SERDES connections are 4 bit wide. Thus, the frequency of the fast memory clock is double the frequency of the slow logic clock, giving a frequency ratio of 2:1 [7].

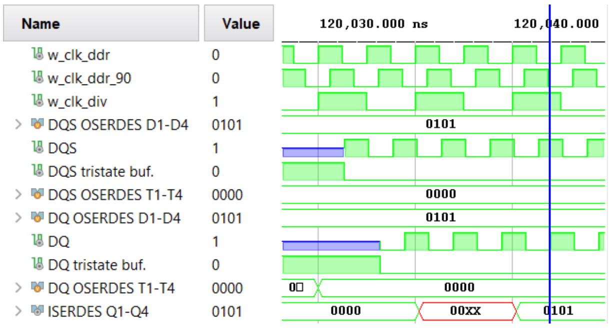
To choose a correct configuration, the SERDES primitives are first tested in simulation and in hardware. An application utilizing two OSERDES and one ISERDES is developed.

One of the OSERDES generates a serial 1-bit data bus (DQ) and the other generates a signal which is analogous to the data strobe signal in DDR SDRAM (DQS). The two signals are routed to the FPGA output pins, where they are observed with measuring equipment, and back to the FPGA input pins. From there, the two signals are connected to an ISERDES which uses the DQS edges to sample the DQ signal.

Figure 5 shows an application simulated by the Vivado 2019.2 development suite. D1-D4 are the parallel data inputs and T1-T4 are the parallel tristate inputs of the OSERDES primitives.

## 4 Memory interface

Memory interfaces are commonly composed of two parts. The PHY acts as a link between the memory chip and the logic part of the interface, the so-called memory controller. As the development of these components requires different technical skills, they are often developed by separate departments, or even different companies [1]. The DFI protocol was developed to allow for a standardized connection between the memory controller and PHY [8]. This work does not employ the DFI protocol.



| Name | Value |
|---|---|
| w_clk_ddr | 0 |
| w_clk_ddr_90 | 0 |
| w_clk_div | 1 |
| DQS OSERDES D1-D4 | 0101 |
| DQS | 1 |
| DQS tristate buf. | 0 |
| DQS OSERDES T1-T4 | 0000 |
| DQ OSERDES D1-D4 | 0101 |
| DQ | 1 |
| DQ tristate buf. | 0 |
| DQ OSERDES T1-T4 | 0000 |
| ISERDES Q1-Q4 | 0101 |

Figure 5. Behavioral simulation of the SERDES primitives.

The developed memory controller functions entirely in the slow clock domain. The user read and write commands, along with the associated address and the write data, are issued into a FIFO. The controller design is based on three state machines.

The first FSM determines the future states of the two remaining FSMs. After initialization is complete, the future states are based on the current state, the state of internal timers, and the state of user-facing signals. The state machine is described in Figure 6 and by the following pattern:

- The refresh operation takes precedence over all other commands. Any read or write operation is interrupted, all banks are precharged, and the refresh state is entered.
- If the command FIFO is not empty, a value is read from it. The selected bank is activated and, depending on the user command, the controller enters either the write or read state.
- The open bank is precharged if (a) a free-running timer signals that the memory must be refreshed, or (b) the next user command requests a different operation, or (c) the next user command is addressed to a different memory row or memory bank, or (d) the FIFO is emptied.
- If the FIFO is empty and the memory controller is not in the refresh state, the idle state is assumed.

The second state machine uses the current and upcoming memory states to determine the proper delay between the states, as discussed in Section 2 and as shown in Table 2. The memory timings are entered into the Verilog code and are thus embedded into each FPGA configuration bitstream.

The third state machine determines the values of the address and command buses, according to the upcoming memory state, as determined in the first FSM.

The PHY is responsible in part for forwarding the command and address bus values to the the memory chip via the IOB. Each command and address pin is assigned one OSERDES.

As the command sequence is generated in the slow

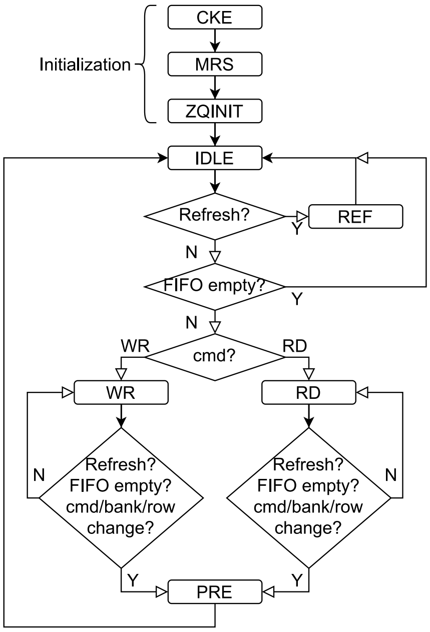


Figure 6. State diagram of the state machine used to determine the upcoming memory state.

clock domain, the memory interface utilizes the so-called 2T command rate. Valid commands are interleaved with the deselect command. This is achieved by toggling the memory chip select pin with the frequency of the slow clock signal (see Figure 7).

To satisfy the memory's setup and hold timing constraints with regard to the command and address buses, the memory clock is shifted by 180° with regard to the internal fast clock signal.

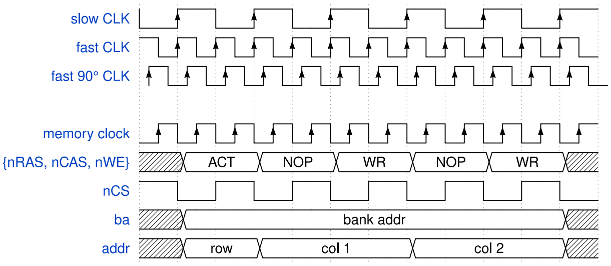


Figure 7. Developed memory interface employing a 2T command rate.

The PHY read and write datapath is more complex. It is implemented as given in Section 3.

For write operations, an example of which is shown in Figure 8, the data strobe signals are phase-aligned with the memory clock, while the phase shift of the data signals relative to the memory clock is -90°. The DQS preamble and postamble are generated as specified by JESD79-3.

The memory interface also respects the specified CWL delay between a write command being issued and the first rising edge of the data strobe signals.

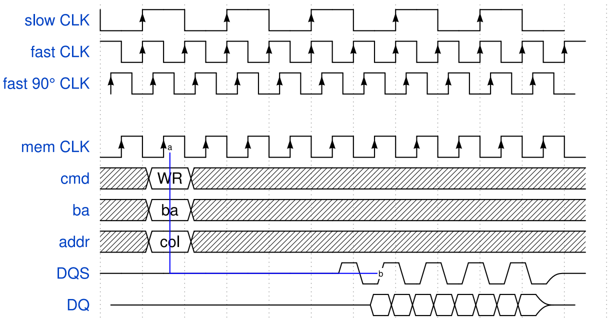


Figure 8. During writes, the data bus is controlled by the PHY. In this example, CWL equals six memory clock cycles. Note the DQS preamble and postamble.

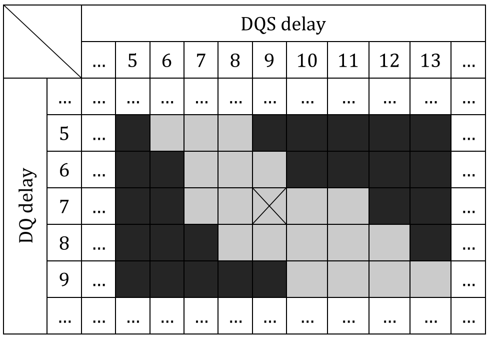
During reads, the memory chip generates the DQS and DQ signals which are in phase (see Section 2).

To place the DQS edges in the center of the DQ data eye (Figure 3), the IDELAY primitives are used. A specially developed read calibration module is used to test every combination of the DQ and DQS IDELAY delay values.

This is achieved by writing a string of interchanging logic ones and zeroes to the memory. Each time the written string is read from the memory, the IDELAY tap values are changed. If the read data corresponds to the written data, the combination of the DQ and DQS delay values is logged as acceptable. When all the delay value combinations have been tested, the optimal DQ and DQS delay values are chosen.

The optimal DQ delay value is the lowest one at which the highest number of the acceptable DQ and DQS combinations is attained. The optimal DQS delay value is the average of the lowest and highest acceptable DQS delay values at the optimal DQ delay value.

The calibration algorithm is graphically shown in Figure 9.



| | | DQS delay | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | ... | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | ... |
| DQ delay | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | 5 | ... | ■ | ▨ | ▨ | ▨ | ■ | ■ | ■ | ■ | ■ | ... |
| | 6 | ... | ■ | ■ | ▨ | ▨ | ▨ | ■ | ■ | ■ | ■ | ... |
| | 7 | ... | ■ | ■ | ▨ | ▨ | ☒ | ▨ | ▨ | ■ | ■ | ... |
| | 8 | ... | ■ | ■ | ■ | ▨ | ▨ | ▨ | ▨ | ▨ | ■ | ... |
| | 9 | ... | ■ | ■ | ■ | ■ | ■ | ▨ | ▨ | ▨ | ▨ | ... |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

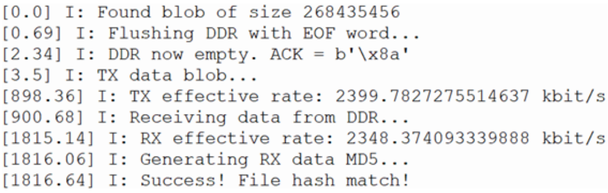
Figure 9. Graphical representation of the implemented read calibration algorithm. The chosen delay values in this example are 7 for the DQ line and 9 for the DQS line.

## 5 Results and Conclusion

The developed interface's performance is tested by two applications at a memory frequency of 325 MHz on a Digilent Arty S7-50 development board which houses a Xilinx Spartan 7 XC7S50CSGA324 FPGA and a PMTech PMF511816EBR-KADN 2 Gbit x16 DDR3L SDRAM chip [9].

The first testing application is written in the Python scripting language. It is run from a PC which communicates with the development board via UART. The script generates a blob of up to 256 MB of random data, which is transferred over UART and the developed memory interface into the DDR3 memory on the development board. After the data is written, it is read from the memory and sent back to the computer. The read and write data blobs are compared and the comparison result is logged.

Figure 10 shows the script output log after a successful transfer of 256 MB (268435456 B) of data into and from the memory at a memory frequency of 325 MHz.



```
[0.0] I: Found blob of size 268435456
[0.69] I: Flushing DDR with EOF word...
[2.34] I: DDR now empty. ACK = b'\x8a'
[3.5] I: TX data blob...
[898.36] I: TX effective rate: 2399.7827275514637 kbit/s
[900.68] I: Receiving data from DDR...
[1815.14] I: RX effective rate: 2348.374093339888 kbit/s
[1816.06] I: Generating RX data MD5...
[1816.64] I: Success! File hash match!
```

Figure 10. Python script output log showing that the 256 MB of data read from and data written to the DDR3L memory are identical.

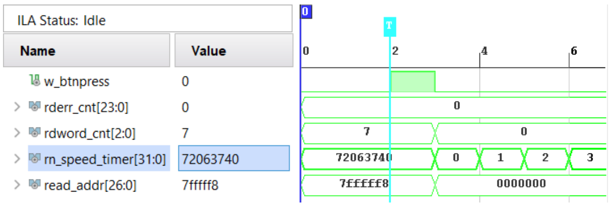
The developed Python script shows that the memory interface is functional and there data loss does not occur.

The data transfer speed between the computer and the DDR3 memory is limited by the UART protocol. At an effective data rate of 2.4 Mbit/s the developed Python script takes approximately 30 minutes to transfer 256 MB of data to and from the development board.

To test sequential memory access, another application is written in which the entire memory chip is filled sequentially with a predefined data pattern. The data is then read sequentially, and any mismatch between the written pattern and the read data increments an error counter. A timer counts the clock cycles between the first write command and the last received read word.

The application has no user interface. Instead, its functionality may be analyzed with the integrated logic analyzer (ILA) IP provided by Xilinx. Figure 11 shows an example of the data captured from the application using the ILA IP.

In this example, the data transfer to and from memory is completed in 72063740 clock cycles of a 162.5 MHz clock, or in approximately 433 ms. The average data transfer speed is thus 605 MT/s. As anticipated, this is lower than the nominal 650 MT/s data rate, as the memory must be periodically refreshed, and the operations of bank activation and precharging are not instantaneous.



ILA Status: Idle

| Name | Value |
|---|---|
| w_btnpress | 0 |
| rderr_cnt[23:0] | 0 |
| rdword_cnt[2:0] | 7 |
| rn_speed_timer[31:0] | 72063740 |
| read_addr[26:0] | 7fffff8 |

Figure 11. Data captured by ILA. The application counts 72063740 clock cycles of a 162.5 MHz clock. No data transfer errors are found.

Xilinx, Inc. recommends the use of two DDR3 memory interfaces with its 7 Series FPGAs. The Xilinx Memory Interface Generator (MIG) is tested to compare it with the developed memory interface. The interface is available free of charge with readable code written in VHDL, but is not open-source [10].

At a memory clock of 325 MHz, the MIG DDR3 interface achieves a sequential data transfer speed of 575 MT/s, which is 30 MT/s or approximately 5% slower than the developed memory interface.

Another DDR3 interface recommended by Xilinx is the logiMEM interface developed by Xylon. The licensing options start at 3540 € per year. The delivered product is encrypted and the code is not readable. This memory interface is not tested, but the nominal data rates and FPGA LUT and FF utilization are available in the product datasheet [11].

The FPGA utilization of the developed interface, MIG, and logiMEM are compared in Table 3.

Table 3. Comparison of FPGA utilization between the existing solutions and the developed interface.

| | LUT | FF | Slices |
|---|---|---|---|
| Own work | 635 | 861 | 197 |
| Xilinx MIG | 4111 | 3561 | 1416 |
| Xylon logiMEM | 1269 | 910 | No data |

The developed interface utilizes 1.9 % of the available LUTs compared to 12.6 % for MIG and 3.9 % for logiMEM. The FF utilization is 1.3 % for the presented work, 5.5 % for MIG, and 1.4 % for logiMEM.

The actual FPGA area usage is determined by slice utilization. The developed interface utilizes 2.4 % of the FPGA, while MIG takes up 17.4 %. As area utilization is not given in the logiMEM documentation, no comparison is made.

The performance of the presented interface is hampered by precharging all banks whenever the memory controller exits the read or write states. This further lowers the already low random access speeds of DDR SDRAM. Future work will improve the interface's random access speeds. To accurately determine the random access speeds of the interface, a new testing application will be developed.

Testing the frequency limits of the developed memory interface is limited by the developed testing applications which are not designed for operation at high frequencies. They fail timing before the FPGA chip reaches its highest clock frequency, i.e. 464 MHz.

The low FPGA utilization enables the use of larger HDL modules on less expensive FPGA chips, while retaining DDR3 functionality.

The developed DDR3 SDRAM interface for Xilinx 7 Series FPGAs works at a memory frequency of 325 MHz, at which a data rate of 1210 MB/s was attained. It is one of few interfaces that supports DDR3 operation at frequencies below 125 MHz.

The presented interface utilizes 2.4 % of the area of the Spartan 7 chip used in this work. This is more than 7-times lower than the 17.4 % required by the MIG IP provided by Xilinx. A direct comparison with the logiMEM interface by Xylon is not possible.

Future work will improve and test random memory access speeds. The interface will be tested at higher frequencies. To increase the memory data rate above 928 MT/s, a faster FPGA chip will be used.

## REFERENCES

[1] Lattice Semiconductor Corporation, “DDR3 PHY [Online].” Available: https://www.latticesemi.com/products/designsoftwareandip/intellectualproperty/ipcore/ipcores02/ddr3phy. [Accessed: 26. 8. 2022].

[2] “JESD79-3F,” standard, JEDEC, 2012.

[3] B. Keeth, R. J. Baker, B. Johnson in F. Lin, *DRAM Circuit Design*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2008.

[4] Xilinx, Inc., “High-Performance DDR2 SDRAM Interface Data Capture Using ISERDES and OSERDES Application Note (XAPP721), [Online].” Available: https://docs.xilinx.com/v/u/en-US/xapp721. [Accessed: 22. 10. 2022].

[5] Xilinx, Inc., “7 Series FPGAs Data Sheet: Overview (DS180) [Online].” Available: https://docs.xilinx.com/v/u/en-US/ds180_7Series_Overview. [Accessed: 25. 8. 2022].

[6] Xilinx, Inc., “7 Series FPGAs Clocking Resources User Guide (UG472), [Online].” Available: https://docs.xilinx.com/v/u/en-US/ug472_7Series_Clocking. [Accessed: 26. 8. 2022].

[7] Xilinx, Inc., “7 Series FPGAs SelectIO Resources User Guide (UG471), [Online].” Available: https://docs.xilinx.com/v/u/en-US/ug471_7Series_SelectIO. [Accessed: 26. 8. 2022].

[8] ddr-phy.org, “About DFI [Online].” Available: https://www.ddr-phy.org/pages/about-dfi. [Accessed: 26. 8. 2022].

[9] “Arty S7 - Digilent Reference [Online].” Available: https://digilent.com/reference/programmable-logic/arty-s7/. [Accessed: 9. 9. 2022].

[10] Xilinx, Inc., “7 Series FPGAs Memory Interface Solutions User Guide (AXI) [Online].” Available: https://docs.xilinx.com/v/u/1.0-English/ug586_7Series_MIS. [Accessed: 16. 9. 2022].

[11] Xylon d.o.o., “IP Cores - logiMEM [Online].” Available: https://www.logicbricks.com/Products/logiMEM.aspx. [Accessed: 16. 9. 2022].

**Jari Beguš** received his B.Sc. and M.Sc. degrees in electrical engineering from the University of Ljubljana (UL) in 2019 and 2022, respectively.

**Andrej Žemva** received his B.Sc., M.Sc. and Ph.D. degrees in electrical engineering from the University of Ljubljana (UL) in 1989, 1993 and 1996, respectively. He is Professor at the Faculty of Electrical Engineering, UL, teaching courses on Digital integrated circuits design, Electronic circuit testing and Linear electronics. His current research interests include digital systems design, logic synthesis and optimization, test pattern generation and fault simulation, electronic system-level verification and real-time image processing

**Damjan Zadnik** is working on his Ph.D thesis at the Faculty of Electrical Engineering in Ljubljana. His current research and development work is focused on iris recognition technology for smart-city access control applications. He is also the CEO of the company Iretec d.o.o. (www.iretec.eu), which operates in the field of iris recognition technology. Damjan Zadnik received the B.Sc degree in 2000 and the M.Sc degree in 2005, both at the Faculty of Electrical Engineering in Ljubljana. In 2013, he also received the M.Sc degree in Business and administration at the Faculty of Economics in Ljubljana. His 22 years long professional career has been dedicated to research and development, product design, product CE certification, electronics design, firmware design, and FPGAs. He was the main design engineer or project leader of 15 developed and CE certified products, including medical laser systems.