[1]Dr. Chenchukrishnaiah G,
Dr. N.Venkatramanaiah,
I. Pavani,
N. Ramadevi

Design and Verification of a DDR SDRAM Memory Controller for DSP Processors Using Verilog

Journal of Electrical Systems

Abstract: **-** Synchronous DRAM (SDRAM) is widely adopted in system designs due to its high speed, burst access capability, and pipelining features. In high-performance applications utilizing processors, the processor's built-in peripheral module typically manages the interface with SDRAM. However, for other applications, the system designer must create a dedicated controller to handle SDRAM initialization, read/write operations, and memory refresh cycles. DDR SDRAM, or Double Data Rate SDRAM, utilizes a dual-edge clocking mechanism to enable faster data transfers by transmitting data on both the rising and falling edges of the clock signal. The DDR controller serves as an intermediary between the DDR SDRAM and the processor. This paper presents the implementation of a DDR controller in Verilog, using Xilinx ISE 14.5 for the design.

Keywords: processor, implementation, processor

1. INTRODUCTION

Image processing data pipelines require significant amounts of data to be buffered, and memory serves as the storage medium for this data within integrated circuits (ICs). Various types of memory can be used for this purpose, each with its own advantages and drawbacks. Static RAM (SRAM) is known for its fast access speeds and low power consumption, but its low storage density—due to the multiple transistors needed to store a single bit—makes it more expensive and less suitable for large-scale storage.

In contrast, Dynamic RAM (DRAM) offers higher storage density and is more cost-effective, though it has slower performance and higher power consumption. These characteristics make DRAM a more practical choice for the main memory in digital imaging systems. There are several types of DRAM, including older, asynchronous versions like Fast Page Mode (FPM) DRAM and Extended Data Out (EDO) DRAM, which are considered too slow for modern applications. Today, Synchronous DRAM (SDRAM), Double Data Rate (DDR) SDRAM, and Rambus DRAM (RDRAM) are the standard memory types used in the PC industry.

SDRAM operates synchronously, meaning its performance is directly tied to the clock signal driving the device. Compared to earlier DRAM technologies, SDRAM offers significantly reduced latency, and its input/output signal interfacing is more streamlined. While SDRAM is still in use, it is increasingly being replaced by DDR SDRAM.

Rambus DRAM (RDRAM) reduces the memory bus width to 16 bits and operates at higher speeds (up to 800 MHz). Although it offers increased bandwidth compared to SDRAM, RDRAM comes with a major drawback: it is a proprietary technology.

DDR SDRAM is a natural progression from SDRAM. It doubles the effective data transfer rate by transferring data on both the rising and falling edges of the clock, thereby increasing the bandwidth.

While DDR SDRAM offers similar performance improvements to RDRAM, it is more cost-effective. Due to its widespread adoption in the PC industry and better long-term availability compared to SDRAM, DDR SDRAM is an ideal choice for imaging applications.

With advancements in modern process technologies (13µm and smaller), FPGAs are increasingly being considered for high-volume applications. FPGAs have significantly evolved in terms of complexity and density, bringing their performance and capacity closer to that of ASICs. As a result, they are now suitable for demanding

[1] Audisankara College of Engineering & Technology1, 2, 3, 4
Corresponding Author: Dr.Chenchukrishnaiah G1
E-mail: gurramchenchukrishnaiah@gmail.com1, nvr.personal@gmail.com 2 pavaniinduru@gmail.com3 , nramaece@gmail.com4

applications that were previously beyond their capabilities. For example, many Xilinx and Altera FPGAs now offer enough resources to implement high-speed memory controller designs. These FPGAs support the necessary features to interface with DDR SDRAM, including I/O compatibility with the SSTL-II electrical standard, DDR registers, and flexible phase-locked loops (PLLs) that can handle varying clock frequencies and meet precise data capture requirements.

Some Altera FPGA families have introduced hard-macro delay chains for data strobes, a feature typically reserved for specialized ASIC cores. The FPGA-based intellectual property (IP) core market now offers memory controllers optimized to leverage these advanced features. In conclusion, FPGAs have evolved to a point where they can effectively implement DDR interfacing for both products and prototypes. Modern FPGA families, when combined with IP cores, provide the capabilities needed to achieve bandwidth levels comparable to those of ASIC designs. The goal and innovation of this paper is to consolidate design techniques for implementing high-speed memory controllers. This paper outlines the key features to consider when selecting an FPGA for high-speed DDR SDRAM interfacing, presents a memory controller data path architecture suitable for Altera and Xilinx FPGAs, and provides design guidelines to meet timing constraints. We present timing measurements of a Virtex-II FPGA interfacing with a Samsung K4H560838C DDR SDRAM at 60 MHz and perform a read data capture timing analysis for a Stratix FPGA interfacing with a Samsung M470L6524BT0 DDR SDRAM at 80 MHz.

Single Data Rate (SDR) SDRAM drives and latches data and command signals on the rising edge of the clock. DDR SDRAM, a variant of SDR SDRAM, builds on its predecessor's technology to achieve faster speeds and lower power consumption. DDR SDRAM is widely used in various embedded applications such as signal processing, networking, and image/video processing, where fast and cost-effective memory is essential. DDR SDRAM was developed to enhance the performance of traditional SDRAM by using techniques like tighter timing controls, which effectively double the data transfer rates. These stringent timing requirements often necessitate the use of phase-locked loops and self-calibration methods to ensure precise timing accuracy.

DDR SDRAM achieves a data transfer rate that is twice the clock frequency by utilizing a 2-n bit prefetch architecture. In this design, 2n bits of data are transferred from the memory cell array to the I/O buffer with each clock cycle.

The data in the I/O buffer is then output in n-bit chunks on both the rising and falling edges of the clock signal. The proposed memory controller, along with the CoreDDR module, is used to interface the memory with the rest of the embedded system. CoreDDR provides a high-performance interface for double data rate (DDR) SDRAM, accepting read and write commands through a simple local bus interface and converting them into the required command sequences for the SDRAM devices. It also manages memory initialization and refresh operations through the memory controller. This memory controller design has been implemented in Verilog. The primary goal of this work is to design and implement a DDR SDRAM controller that serves as the memory interface between the DDR SDRAM module and the main embedded system.

2. LITERATURE SURVEY

2.1 Overview of Storage Architectures

The literature review began with an exploration of various storage architectures designed to handle extremely large volumes of data, continuing with an examination of storage elements capable of managing massive datasets, on the order of Terabytes or even Zettabytes. A key aspect of this process was selecting an appropriate memory controller to manage the different functions of these storage elements.

Sang-Woo Jun, Ming Liu, and colleagues [1] developed an innovative high-performance storage architecture called BlueDBM (Blue Database Machine). This architecture is designed for scalable, distributed flash storage, aiming to achieve high-performance, high-capacity storage with random access capabilities. The system utilizes multiple flash chips, sharing them to minimize latency and maximize throughput. Flash controllers, integrated with Ethernet, manage the chip-to-chip communication network. Positioned between the flash storage element and the host, these controllers facilitate hardware acceleration, significantly reducing latency. Data transfer is handled via PCIe interfaces. The BlueDBM system was specifically implemented for Big Data applications, where direct communication links between nodes are established through flash controllers, enhancing system performance. The BlueDBM architecture consists of nodes with storage resources, and data access is accelerated

through reconfigurable fabric. Nodes are interconnected via an FPGA-based network, improving overall system efficiency. The developers also created a small-scale prototype, where system data transmission scales proportionally with the number of nodes. In this model, the average latency for user applications accessing flash storage is less than 70µs, including a 3.5µs network overhead.

Duncan G. Elliott and colleagues [5] developed an architecture known as Computational RAM, which integrates processing capabilities directly into memory. This "processor-in-memory" design maximizes bandwidth utilization within the internal memory. In the customized model of Computational RAM, a host CPU can read and write to any memory location during an external memory cycle. During a processing cycle, all processing units perform the same basic operation and access the same memory space within their dedicated memory partitions. Essentially, Computational RAM functions as a SIMD (Single Instruction, Multiple Data) processor with non-shared, consistently addressed memory. Communication between processors is facilitated by a vehicle, which is essential for performing combinational operations. Additionally, a linear interconnect, which expands into two dimensions at the ends of rows, supports both 1D and 2D nearest neighbor operations. Computational RAM (or C-RAM) can operate as either a standard memory device or as a SIMD computer. When used as memory, Computational RAM competes with traditional DRAM in terms of access time, packaging, and cost.

Kermin Fleming et al. [6] address the challenge of low performance in hardware designs spread across multiple FPGAs, which is typically caused by the inefficiency of maintaining timing across separate FPGA units. In their work, they propose a system that efficiently partitions complex designs across multiple FPGAs using specifically tailored, latency-insensitive interconnects. They describe the automatic integration of a high-performance system for managing these inter-FPGA connections. By mapping large research models onto multiple FPGA platforms, the authors demonstrate significant improvements in design feasibility, compilation time, and even clock performance. The main challenges in creating an inter-FPGA communication system for latency-insensitive connections are execution and correctness. By using latency-insensitive FIFOs, the system only transports explicitly queued data. However, to achieve high performance, the system must exploit the inherent parallelism of the pipeline in the partitioned design. Since multiple connections can pass between FPGAs, there is a need to multiplex the physical connections between units. The authors introduce a language extension and compiler that leverages latency-insensitive design principles to achieve high-performance execution across multiple FPGAs. This approach allows for the construction of larger research models, improved compilation times, and, in many cases, better performance than single FPGA systems. The compiler performs especially well when partitioning digital signal processing applications, which often involve high bandwidth and computational requirements. These applications are more resilient to the latency introduced by chip-to-chip communication and can see significant performance gains when scaled to multi-FPGA systems. However, applications with larger amounts of input, such as processor models, may experience performance degradation compared to single FPGA systems due to latency. Nonetheless, these applications still benefit from enhanced resource access, better scalability, and reduced setup times.

3. PROPOSED METHOD

Double Data Rate Synchronous Dynamic Random-Access Memory (DDR SDRAM) is an advanced version of standard SDRAM, offering higher bandwidth and faster speeds. DDR SDRAM provides improved data transfer rates compared to traditional SDRAM, allowing transactions on both the rising and falling edges of the clock cycle. For instance, DDR SDRAM running at 133 MHz can achieve a maximum transfer rate of about 1,064 million transfers per second (MT/s) to the L2 cache. This clock-doubled variant of SDRAM began replacing traditional SDRAM during the 2001-2002 period.

The Control Interface module in the SDRAM system receives commands and associated memory addresses from the host, decodes these commands, and passes them to the Command module. The Command module then processes the decoded commands and addresses to generate the appropriate control signals for the SDRAM. Meanwhile, the Data Path module handles the data flow during both write and read operations. The top-level module also includes two Phase-Locked Loops (PLLs) used in the CLOCK_LOCKS.

DDR SDRAM Control Interface Module

The Control Interface Module interprets commands from the host, such as NOP, WRITEA, READA, REFRESH, PRECHARGE, and LOAD_MODE, along with their corresponding addresses, and forwards them to the Command Module. It also decodes LOAD_REG1 and LOAD_REG2 commands to load the REG1 and REG2 registers with values from the address input. The REFRESH_REQ signal is triggered when the refresh counter reaches zero and remains active until the Command Module acknowledges the refresh request. Upon acknowledgment, the counter is reloaded using the value in REG2, and the process repeats. REG2 holds a 16-bit value that represents the interval between REFRESH commands, and its value is determined by the ratio of REF_PERIOD to CLK_PERIOD.

DDR SDRAM Command Module

The Command Module receives decoded commands from the Control Interface and refresh requests from the refresh control logic, and then generates the appropriate commands for the SDRAM. It includes a simple arbitration mechanism that prioritizes commands from the host interface and refresh requests from the refresh logic.

DDR SDRAM Data Path Module

One of the more complex aspects of designing a DDR SDRAM controller is handling data transmission and reception at double data rates. The Data Path Module is responsible for transmitting data to and from the memory. Its main function is to store write data and compute the values for the read data path. This module interfaces with the host's data lines, accepting write data on the DATAIN port during WRITEA commands and outputting data on the DATAOUT port during READA commands. The width of the data path into the controller is twice that of the data path leading to the DDR SDRAM devices. Specifically, the DATAIN and DATAOUT ports in the Data Path Module are set to 32 bits, while the DQ port is set to 16 bits. To accommodate larger data widths, multiple Data Path Modules can be cascaded, enabling the controller to handle data paths wider than 32 bits.

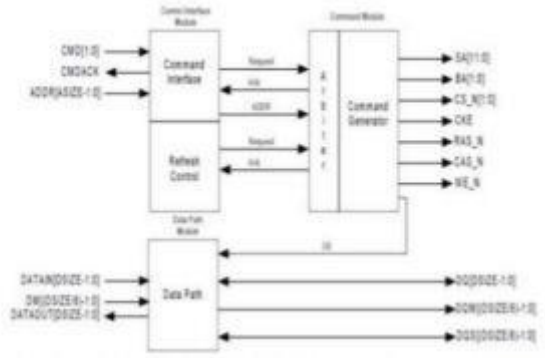
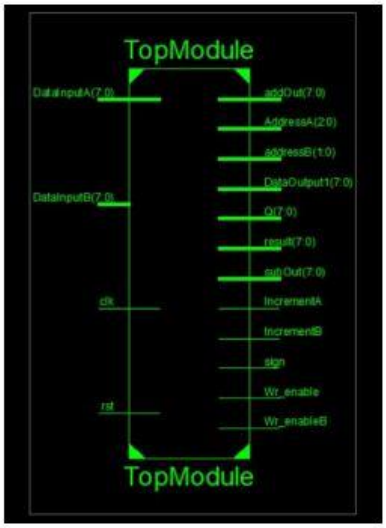
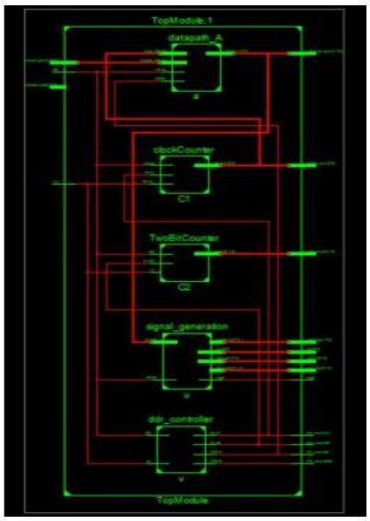
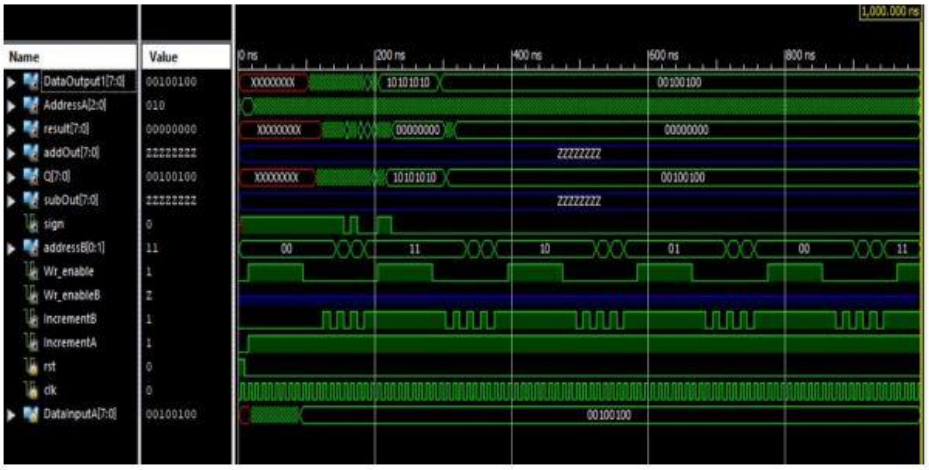


Fig: Functional Block Diagram of proposed DDR SDRAM Memory controller core

4. RESULTS







5. CONCLUSION

An efficient and fully functional DDR SDRAM controller has been designed and verified using Verilog HDL. The controller generates various timing and control signals, ensuring proper synchronization and management of the operation flow. The memory system operates at twice the frequency of the processor, maintaining performance without any degradation. This allows for a reduction in the data bus size. However, the main limitation of this controller is its complex design, which involves a large number of buffers, potentially leading to increased delay in the system.

REFERENCES

[1]. R. C. Baumann, “Soft errors in advanced computer systems,” IEEE Design Test. Comput., vol. 22, no. 3, pp. 258–266, May/Jun. 2005.

[2]. N. P. Rao and M. P. Desai, “A detailed characterization of errors in logic circuits due to single-event transients,” in Proc. Euromicro Conf. Digit. Syst. Design, Funchal, Portugal, 2015, pp. 714–721.

[3]. B. Narasimham et al., “Characterization of digital single event transient pulse-widths in 130-nm and 90-nm CMOS technologies,” IEEE Trans. Nucl. Sci., vol. 54, no. 6, pp. 2506–2511, Dec. 2007.

[4]. M. P. Baze and S. P. Buchner, “Attenuation of single event induced pulses in CMOS combinational logic,” IEEE Trans. Nucl. Sci., vol. 44, no. 6, pp. 2217–2223, Dec. 1997.

[5]. S. Buchner, M. Baze, D. Brown, D. McMorrow, and J. Melinger, "Comparison of error rates in combinational and sequential logic," IEEE Trans. Nucl. Sci., vol. 44, no. 6, pp. 2209–2216, Dec. 1997.

[6]. J. Benedetto et al., "Heavy ion-induced digital single-event transients in deep submicron processes," IEEE Trans. Nucl. Sci., vol. 51, no. 6, pp. 3480–3485, Dec. 2004.

[7]. M. A. Bajura et al., "Models and algorithmic limits for an ECC-based approach to hardening sub-100-nm SRAMs," IEEE Trans. Nucl. Sci., vol. 54, no. 4, pp. 935– 945, Aug. 2007.

[8]. T. Bonnoit, M. Nicolaidis, and N.-E. Zergainoh, "Using error correcting codes without speed penalty in embedded memories: Algorithm, implementation and case study," J. Electron. Test., vol. 29, no. 3, pp. 383–400, Jun. 2013.

[9] P. Papavramidou and M. Nicolaidis, "Test algorithms for ECC-based memory repair in ultimate CMOS and post-CMOS," IEEE Trans. Comput., vol. 65, no. 7, pp. 2284–2298, Jul. 2015.

[10]. R. Vemu, A. Jas, J. A. Abraham, S. Patil, and R. Galivanche, "A lowcost concurrent error detection technique for processor control logic," in Proc. DATE, Munich, Germany, 2008, pp. 897–902.

[11]. S. Ghosh, N. A. Touba, and S. Basu, "Synthesis of low power CED circuits based on parity codes," in Proc. 23rd IEEE VLSI Test Symp., May 2005, pp. 315–320.

[12]. V. Sapozhnikov, V. Sapozhnikov, D. Efanov, and A. Blyudov, "On the synthesis of unidirectional combinational circuits detecting all single faults," in Proc. EWDTS, 2014, pp. 1–8.

[13]. C. A. Lisboa, F. L. Kastensmidt, E. H. Neto, G. Wirht, and L. Carro, "Using built-in sensors to cope with long duration transient faults in future technologies," in Proc. ITC, 2007, pp. 1–10.

[14]. R. P. Bastos, F. S. Torres, G. Di Natale, M. Flottes, and B. Rouzeyre, "Novel transientfault detection circuit featuring enhanced bulk builtin current sensor with low-power sleep-mode," J. Microelectron. Rel., vol. 52, nos. 9–10, pp. 1781–1786, 2012.

[15]. M. Nicolaidis, "Double-sampling design paradigm—A compendium of architectures," IEEE Trans. Device Mater. Rel., vol. 15, no. 1, pp. 10–23, Mar. 2015.

[16]. D. Ernst et al., "Razor: A low-power pipeline based on circuit-level timing speculation," in Proc. MICRO, 2003, pp. 7– 18.

[17]. S. Das et al., "RazorII: In situ error detection and correction for PVT and SER tolerance," IEEE J. Solid-State Circuits, vol. 44, no. 1, pp. 32–48, Jan. 2009.

[18]. M. R. Choudhury and K. Mohanram, "Low cost concurrent error masking using approximate logic circuits," IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., vol. 32, no. 8, pp. 1163–1176, Aug. 2013.