International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:6, 2008

Bandwidth, Area Efficient and Target Device Independent DDR SDRAM Controller

T. Mladenov, F. Mujahid, E. Jung, and D. Har

***Abstract*—**The application of the synchronous dynamic random access memory (SDRAM) has gone beyond the scope of personal computers for quite a long time. It comes into hand whenever a big amount of low price and still high speed memory is needed. Most of the newly developed stand alone embedded devices in the field of image, video and sound processing take more and more use of it. The big amount of low price memory has its trade off – the speed. In order to take use of the full potential of the memory, an efficient controller is needed. Efficient stands for maximum random accesses to the memory both for reading and writing and less area after implementation. This paper proposes a target device independent DDR SDRAM pipelined controller and provides performance comparison with available solutions.

***Keywords*—**DDR SDRAM, controller, effective implementation

I. Introduction

DYNAMIC memories and in particular the synchronous dynamic random access memory (SDRAM) are gaining more and more popularity dominating the market in the past decade. What makes SDRAM so attractive is its simple structure, resulting in compact and inexpensive device, together with the ever increasing speed. Several revisions have been introduced to the market in the past decade. Starting from SDR (single data rate) through DDR (double data rate) and finally DDR2 (double data rate 2) the maximum bandwidth has reached 800 Mb/s.

Although more than 90% of all manufactured SDRAM memories go for the personal computer industry, they grow more and more popular outside this primary field of application. Nowadays almost every embedded application, especially in the field of signal processing, requires significant amount of memory. The most used type of dynamic memory for that purpose is DDR SDRAM. The starving for better and better quality in image, video and sound processing brings the need of more and more cheep and fast memory.

While the semiconductor industry is increasing the maximum possible bandwidth of the dynamic memories, the very basic design idea of SDRAM memories puts the main responsibility for the effective utilization in the controller.

For FPGA (field programmable gate array) design the IC (integrated circuit) manufacturers are providing commercial memory controller IP cores working only on their products. Main disadvantage is the lack of memory access optimization for random memory access patterns. The 'data path' part of those controllers can be used free of charge. This paper propose an architecture of a DDR SDRAM controller, which takes advantage of those available and well tested data paths and can be used for any FPGA device or ASIC design. Furthermore an area efficient mechanism for maximum utilization of the available bandwidth during random memory access is proposed.

The paper is organized as follows. Section II introduces the interface of a single DDR SDRAM chip and briefly explains how it works. The proposed controller architecture is described in section III. Section IV presents the test results. Finally, section V concludes this paper.

II. DDR SDRAM Operation

Dynamic memories are more complex then their static counterparts. Knowing their special features and characteristics is vital for designing an optimal memory controller. The organization and operation of DDR SDRAM memory is to be briefly presented.

A. Memory Organization

The DDR SDRAM memory is organized in four banks with equal size, which is shown on Fig.1. The memory is addressed by first selecting the bank, then the raw and finally the column. Reading and writing is done in bursts. Two, four or eight column addresses can be accessed at a time. The amount of memory words can be calculated according to formula (1):

$$MemorySize = RowNum * ColNum * BankNum \quad (1)$$

'RowNum' represents the number of rows and 'ColNum'- the number of columns. 'BankNum' can be substituted with four and here is given only for completeness.

Manuscript received September 30th, 2006. This work has been supported in part by the Center for Distributed Sensor Network at GIST, in part by the GIST Technology Initiative (GTI), in part by the MIC, Korea, under the ITRC support program supervised by the IITA" (IITA-2005-C1090-0502-0029), and in part by the Korea Science and Engineering Foundation (KOSEF) through the Ultrafast Fiber-Optic Networks (UFON) Research Center at Gwangju Institute of Science and Technology (GIST).

T. Mladenov is a masters' student at Gwangju Institute of Science and Technology, Gwangju, South Korea (phone: 82-62-970-2267, e-mail: todor@gist.ac.kr).

F. Mujahid is a PhD candidate at Gwangju Institute of Science and Technology, Gwangju, South Korea (e-mail: fahad@gist.ac.kr).

E. Jung is a post doctoral researcher at Gwangju Institute of Science and Technology, Gwangju, South Korea (egjung@gist.ac.kr).

D. Har is a full time professor at Gwangju Institute of Science and Technology, Gwangju, South Korea (e-mail: hardon@gist.ac.kr).

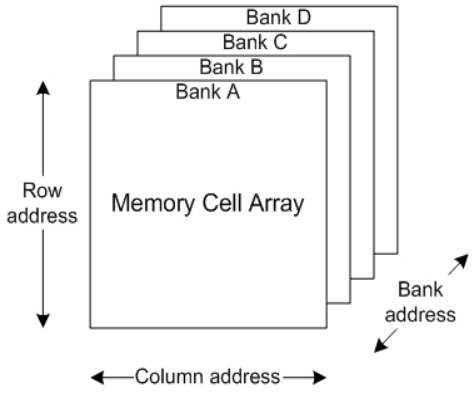

Fig. 1 Memory organization

'MemorySize' gives the number of word cells in the memory. Each cell varies form 4 bit to 16 bit, depending on the particular device.

B. Interface

The data is transferred between controller and memory through a bi-directional data bus – ddr_dq as it is depicted on Fig. 2. The width of this bus depends on the internal memory word (cell) organization and as it was pointed out can be four, eight and sixteen bits. The data is transferred at twice the clock frequency, which is the main feature of DDR SDRAM memory. The clock signal is send to the memory via two differential inputs ddr_clk and ddr_clkn (n stands for inverted). All addresses and control input signals are sampled on the crossing of the positive edge of ddr_clk with the negative edge of ddr_clkn. SDRAM memory is controlled through commands which are encoded with following signals: ddr_rasn , ddr_casn, ddr_wen, ddr_csn. Data is read and written under the bi-directional strobe signal ddr_dqs. Every eight bits of the data bus have individual strobe signal (in case of data word wider than eight bits). Additional mask signal (ddr_dm) is available again for every eight data bits providing

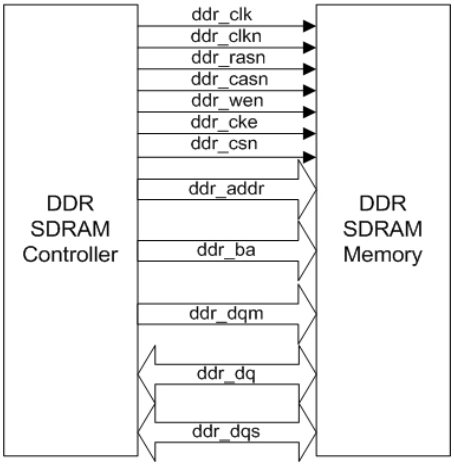

Fig. 2 Memory interface

a mean for accessing only individual addresses within a burst. The row and column addresses are provided through a ddr_addr bus, which has the width of the wider row address. Additional ddr_ba two bit wide bus provides the bank address. In combination with ddr_addr, during a specific command, ddr_ba sets up memory internal parameters like burst length, CAS latency, etc. Ddr_cke signal controls the power down functionality of the memory.

C. Initialization

As it can be seen from Fig. 2, there is no reset signal on the memory chip. For proper operation the memory has to be initialized after power up before it can be used. If a not allowed command is issued, the memory has to be reinitialized to secure proper operation. The initialization procedure can be found in [1],[3],[6] and [7].

D. Memory Refresh

DDR SDRAM memories require a refresh of all rows in any rolling 64 ms. The refresh in normal operating mode is done by issuing an auto refresh command. The average interval tREFI for issuing this command can be found by (2).

$$tREFI = \frac{64ms}{RowNumPerBank} \quad (2)$$

The address is internally generated and the same row is refreshed in all four banks at the same time with one command. Therefore 'RowNumPerBank' stands for the number of rows in one bank. If they are 4098, the refresh has to be done every 15.6 ms. Up to eight auto refresh commands can be issued one after another and the time interval to the next refresh command will be extended accordingly.

E. Write Operation

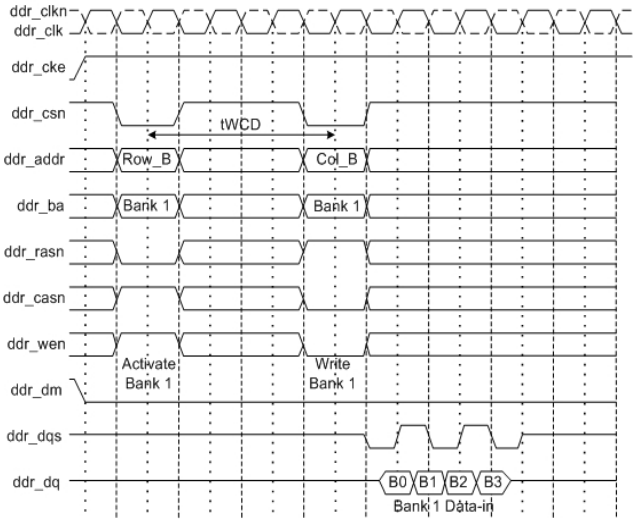

Fig. 3 Write operation

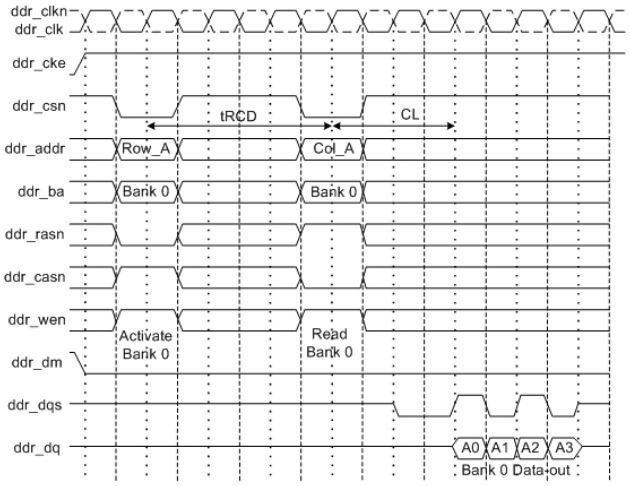

Fig. 4 Read operation

The waveforms on Fig. 3 represent the write of four bytes into the DDR SDRAM memory. Before issuing a write command the controller has to 'open' the row, which is to be accessed. This is done by a "Row Active" command. Together with it the bank and row addresses are provided. After the row has been activated a specified amount of time has to pass before issuing a read or write command. For some memories this time is different for write and read operations. On Fig.3 it is depicted with 'tWCD' for write operation.

Data is transferred between memory controller and memory in a burst manner. The length of the bursts is set during the initialization of the memory and can be changed during operation with internal register set up command. The available burst modes are two, four and eight. The data in those bursts is written in sequential or interleave column addresses.

The first byte of the burst has to be send to the memory one clock after the write command with deviation of quarter clock cycle. Together with the data a strobe signal is provided. Data is transferred on both edges of the strobe signal. There are specific requirements about the pre-amble and post-amble time of the strobe signal. Mask signal helps for odd number of words to be written in the memory. Two bytes will be written during the rising and falling strobe edges if the value of the mask signal is zero. The write burst can be interrupted with another write or read burst. The memory hardware is pipelined, which provides continuous, uninterruptible access to the memory within an 'opened' row.

After a write operation the current row has to be 'closed' by a row precharge command or by write command with auto precharge, before any other row to be activated ('opened').

F. Read Operation

The waveforms on Fig. 4 present the read operation of four bytes from DDR SDRAM memory. Similar to write operation, the accessed row has to be activated. After 'tRCD' time a read command is issued. The row address is given with the row activation command and the column address with the read command. The first byte of the read data is coming after 'CL' (CAS latency) cycles at the rising edge of the strobe signal. Data is transferred on both edges of the strobe signal. The data mask signal has no effect during read operation.

The CAS latency is set during initialization. It can be changed during operation with a specific command. Typical values are 2, 2.5, 3 and 4. They depend on the particular device and the frequency of operation.

The row stays 'open' after row active command for up to 100,000 clock cycles and has to be 'closed' within this interval with a precharge command or by a read command with auto precharge. The read operation can be interrupted with other read or write command. Full description and specifications of a DDR SDRAM memory chip is given in [3].

III. Memory Controller Implementation

As it is pointed out in [2] there are three critical design decisions, which complicate the implementation of DDR SDRAM controller. First the edge of the output data coming from the memory is aligned with the clock. Second the data strobe edges are not in the middle of the data eyes. Furthermore the data strobe signals are bi-directional and they can not be free-running clocks and have to be driven by the controller.

On Fig. 5 is presented the main block diagram of a DDR SDRAM memory controller. The three main blocks will be discussed in details.

A. Data Path

The data path is responsible for the main feature of the DDR SDRAM memory. It provides the double data rate operation. There are many requirements complicating the implementation of this block. The data is latched on both the rising and falling edge of the strobe signal. This requires the strobe signal to have constant 50% duty cycle. The signal delay in both blocks on Fig. 6 and on the transmission lines to the memory has to be the same for the strobe and the data signals. Specific line driving abilities are required from the I/O buffers in both blocks.

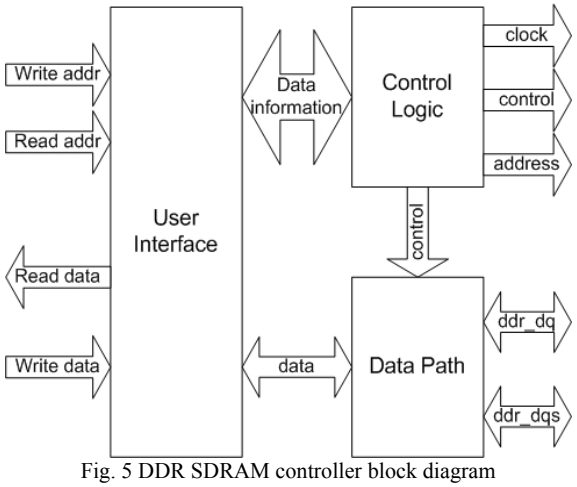

Fig. 5 DDR SDRAM controller block diagram

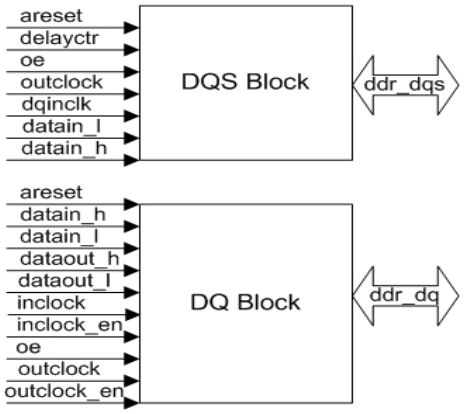

Fig. 6 Data Path in Stratix II FPGA

The implementation depends mainly on the targeted platform. For ASIC those tough requirements would take a significant design time and are a challenge, which is not within the scope of this paper. The controller, which is being described, is implemented on a FPGA device. There is significant variety of FPGAs and most of them provide a mechanism for using dual data transmission resources usually in form of IP core functions. In some cases those functions make use of the common logic available on the device and the I/O buffers.

Fig. 6 shows the implementation of data path on Stratix II FPGA [4]. Here the double data rate functional blocks DQS (handling the strobe signals) and DQ (handling the data) are fully realized in the I/O blocks without using additional general purpose logic. One strobe signal is handling maximum eight data lines. The described controller uses sixteen bit data bus and two strobe signals. The data path block can be changed according to the targeted device.

The required strobe shifting delay for both writing and reading may be accomplished dynamically with DLL circuit. The minimum frequency at which the DLL circuit of Stratix II device is working is 100MHz. A static delay shift can be used, but usually it needs calibration after initialization of the memory.

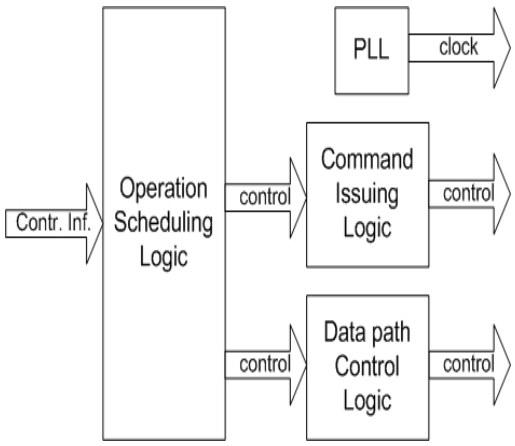

Fig. 7 Control Logic Block Structure

B. Control Logic

The structure of the control block is presented on Fig. 7. A PLL circuit is providing the differential frequency for the controlled memory and for the controller itself [5].

From the waveforms of the read and write operations it seems very reasonable the control of the commands to the memory and data to be separately implemented. Every command is described in separate state of one state machine. Data path control is implemented with combinational logic. This functional division provides the flexibility needed for optimal utilization of the maximum available bandwidth. A third block named "Operation scheduling logic" on Fig. 7 is needed to synchronize the work of the previous two. The initialization and periodic refresh of the memory are handled by this block. Pipelining of it increases the through put and takes full advantage of the available bandwidth in case of linear memory addressing.

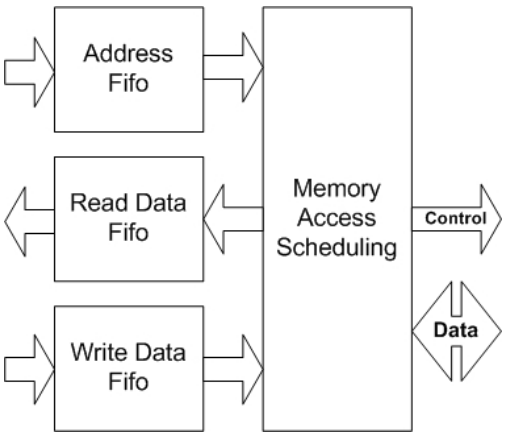

Fig. 8 User Interface

C. User Interface

The user interface is the last and probably most significant part of the memory controller from efficiency point of view. Usually the lowest frequency of operation for standard DDR SDRAM memory is around 80 MHz and reaches 166 MHz. For FPGA implementation it is difficult to achieve high frequency after implementation, when the complexity of the application is significant and takes more than half of the device resources. In those cases an incremental compilation can be used, which allows for the time critical parts to be separately optimized and incorporated into the whole design. The cost is extra logic due to partial optimization of the design. Therefore the controller is very often working at a different frequency from the rest of the design. Higher frequency of operation also increases the throughput.

A block diagram of the 'User Interface" part is presented on Fig. 8. For interfacing between different clock domains,

mostly fifos are used. The depth of the fifos is a major design parameter, which depends upon the particular application. It has to be enough to compensate for the latency, normal for fifos working with two clocks. As there are three fifos, if their size is significant they will also occupy significant resources.

On Fig. 8 there are two fifos handling the data flow in each direction. The third fifo is responsible for the addresses. The write or read operation is encoded with the address as a flag situated as additional most significant bit.

D. Address Reordering Mechanism

The "Memory Access Scheduling Block" on Fig. 8 is responsible for the effectiveness of the controller. Its structure depends mainly on the application. For a video codec, where memory is mostly accessed in blocks with linear increment of the address, it has no more functionality but to transfer addresses and data. When the DDR SDRAM memory is accessed randomly, optimization mechanism shown on Fig. 9 turns to be very effective according to Table II. It incorporate memory and logic for regrouping the addresses in order to have longer memory access bursts with less switching between rows and banks. Additional regrouping of the read data is needed for it to be available in the order it was requested. The depth of the memory for regrouping increases the efficiency of the memory controller but also increases the area and data latency. Buffering of up to eight words per row turns to add less area, according to Table I, and improves significantly the performance according to Table II.

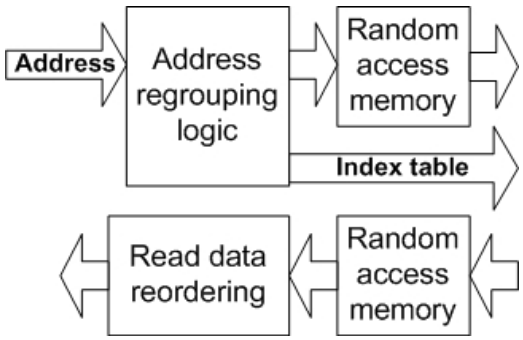

Fig. 9 Address reordering

IV. Results

All the implementation tests and simulations are done on Altera Stratix II FPGA device with following parameters: 24,176 ALMs (adaptive logic modules), 48,352 ALUTs (adaptive look-up tables), 60,440 equivalent LEs (logic elements), 2,544,192 total memory bits, 36 DSP blocks and 12 PLLs.

Table I shows an area comparison between the proposed controller and Alera's one which has no mechanism for optimizing memory access. Comparison with other manufacturers' controllers can not be done in the same device and therefore will not be fair.

TABLE I
AREA COMPARISON

Company	Proposed Controller	Altera's Controller
Stratix II Device 16 bit word	ALUTs	ALUTs
Area	667 and 870*	800

* With address reordering mechanism

Table II proves the efficiency of the proposed address reordering mechanism. The test was done by accessing two different locations for read and write operations with linear increment of the address. Two bytes are read or written at a time. The big difference in clock cycles is due to the row precharge time, which address reordering reduces significantly.

TABLE II
ADDRESS REORDERING EFFECT

	With reordering	Without reordering
Clock cycles for 128 operations	336	2496

Table III gives a speed comparison. Lattice's controller is given for completeness [8], [9].

TABLE III
SPEED COMPARISON

Speed	Proposed Controller	Altera's Controller	Lattice's Controller
Device:	Stratix II	Stratix II	ORCA 4
Max Frequency, MHz	200	200	147

V. Conclusion

This paper aims to present to the HDL developing public a non commercial DDR SDRAM controller, with structure proven to be efficient. Additionally an address reordering mechanism was introduced, which saves a lot of clock cycles on the price of comparatively less extra area.

References

[1] Double Data Rate (DDR) SDRAM Specification, JEDEC STANDARD, JESD79E, May 2005
[2] The Love/Hate Relationship with DDR SDRAM Controllers, Graham Allan, MOSAID, www.design-reuse.com.
[3] 128Mb DDR SDRAM, Device Specification, Hynix, April 2006
[4] altdq & altdqs Megafunction, User Guide, Altera, March 2005
[5] PLLs in Stratix II & Stratix II GX Devices, April 2006
[6] How to Use DDR SDRAM, User's Manual, Document No. E0234E40, ALPIDA, September 2005
[7] Initialization Sequence for DDR SDRAM, Technical Note, TN-46-08, Micron.
[8] DDR SDRAM Controller, Reference Design RD1020, Lattice Semiconductor Corporation, April 2004.
[9] DDR SDRAM Controller Using Virtex-4 FPGA Devices, Oliver Despaux, Application Note, March 27, 2006.