RESEARCH ARTICLE OPEN ACCESS

Development of DDR2 SDRAM Module Interface Software Core

Prajakta Chandilkar*, Dr. Uday Wali**
**(M.tech Student Electronics and Communication, KLE Dr. M.S. Sheshgiri College of Engineering and Technology, Belagavi, Karnataka, India ,prajakta0704@gmail.com*
***(Professor, Dept. of Electronics and Communication, KLE Dr. M.S .Sheshgiri College of Engineering and Technology, Belagavi, Karnataka, India, udaywali@gmail.com)*

ABSTRACT
Memory module simulation is critical to testing of embedded systems. Hardware software co-design strategy requires that clock accurate system modules be available in software. DDR2 SDRAM is one of the popular memory modules used in embedded systems. Memory modules have their own built in controllers to interpret and execute commands from the processor. This paper describes an implementation of the finite state machine (FSM) of a DDR2 SDRAM module. The module can be used as a basis for hardware design of memory modules. It can also be used as a part of hardware-software co-design tool for embedded systems using DDR2 memory modules. The technology independent FSM is implemented in Verilog HDL.

Keywords Command sets, DDR, DDR2 SDRAM, Hardware-Software co-design, Verilog

Date of Submission: 29-08-2017 Date of acceptance: 09-09-2017

I. INTRODUCTION

Developments in VLSI technology are driven by the never ending demand for faster processors. Faster processors need faster memory. On chip memory is built using the same technology as the processor but is limited in size. Many microcontrollers have few kilo bytes to few hundred kilo bytes of on chip RAM. However this memory is hardly sufficient for resident operating system and associated programs. Applications on such devices will need large external RAM. Most of the contemporary data is in the form of images, sound, real time signals, etc. which require very large memory to store and operate. Using Gigabytes of RAM is not very uncommon in current processor scenario. It is often seen that memory modules are the first devices to be built and tested while introducing a new technology. In order to facilitate system integrators to build such systems, standardization of device interfaces is a necessity. Some of the standard memory interface technologies are SDRAM, RDRAM, DDR SDRAM, etc. There have been several advancements in DDR SDRAM technologies. The earliest SDRAM modules had a clock frequency of 133 MHz. DDR doubled the transfer rate and worked at double the frequency, at 266MHz. Recent technologies support DDR SDRAM running at 1.6 GHz. While the latest memory technologies are used in high end processors, embedded systems rely on DDR2 running at 266 MHz [1]. DDR family has a common hardware interface [2]. So, it is easy to modify an existing interface hardware to match any other member of the DDR family.

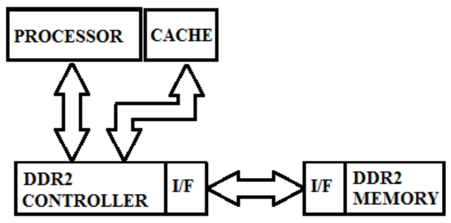
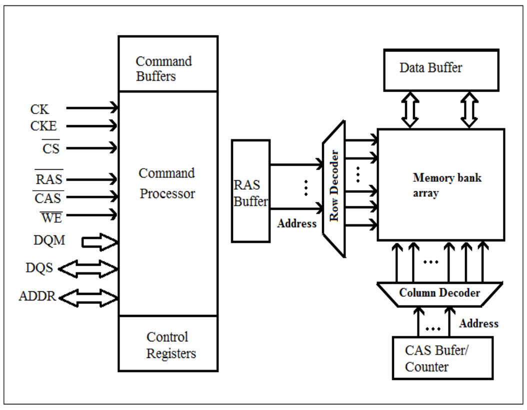


Fig1.overview of DDR2 IP

DDR2 controller and DDR2 IP are two separate hardware units. Implementation of controller has been reported in literature [3][4] but the implementation of the memory module, which has to mirror the operations of the controller is not available. While DDR2 IP works between the controller and the memory, the controller itself works between the processor and DDR2 IP. The processor makes a request for a block of memory to the DDR2 controller. The DDR2 controller will perform one or many burst reads from the DDR2 memory, stores it or forwards the contents to cache. After the process is complete, it interrupts the processor and updates the status of memory request as completed. Cache to memory and memory to cache transfers occur in terms of pages of memory. Therefore DDR should be capable of writing and reading pages of memory, without involving the CPU.

II. OVERVIEW OF DDR2 SDRAM

Figure 2 shows the general block Diagram of DDR2 SDRAM memory architecture. Interface signals are differential clock inputs CK and CKbar, chip select CS and command control signals Row Address Strobe RAS, Column Address Strobe CAS, and Write Enable WE.



Overview of DDR2 SDRAM

Fig2. Overview of DDR2 SDRAM

All the input signals and address signals are sampled at the positive edge of CK and negative edge of CKbar. CKE is the clock enable which should be kept always high throughout operations. Pulling CKE down will take memory into a low power mode. CS, RAS, CAS and WE are active low inputs. Data bus is a 16-bit bidirectional bus. Data mask DQM is used to suppress (mask) specific bits during write operation. A0 to A12 are the 13-bit address, multiplexed into row address and column address. A binary decoder is used as an address multiplexer as shown in the in fig. 2. Operations of DDR2 like Read, Write, Pre-charge, Mode Set Register (MRS); Refresh, Bank Active etc. are activated by using the control signals of the DDR2 SDRAM. As it is a DRAM, it will have some memory and refresh circuits. Additional hardware is required to implement the standard interface. Some of the components of DDR2 RAM are discussed below.

2.1 Memory bank array

Memory bank is a logical storage unit of memory. A bank consists of multiple numbers of rows and columns. The size of the memory bank is determined by bits in a column and rows. Row column organization of the bank may be configured using MRS commands.

2.2 Control Registers

Control registers is used to control the general behavior of the DDR2 command processor (see fig. 2). Control registers are used to store parameters related to communication protocol, address mode, refresh rate, command pipeline, data buffering, etc. Control registers can only be written, not read. MRS command is used to update the control registers. Address bus is used to supply data to update the registers. So, the address supplied as part of MRS command is actually content of registers to be set.

2.3 Data Buffer

Data buffer is used to temporarily store data, while is moved between controller and the memory. DQ is a bidirectional data bus which is can be used to read a data from a given address location or write a data to a given address location. During such transfers, data is stored temporarily in data buffer. Data buffers are also used during refresh and precharge cycles.

III. STATE TRANSITION DIAGRAM AND FUNCTIONAL TABLE FOR DDR2 SDRAM

Table 1 shows the functional table for the DDR2 SDRAM. This table describes seven basic commands to be implemented by the memory module. Each command has a distinctive set of control values to be used for a specific action to be implemented. Note that the values are not transitions but signal levels to be maintained at the clock transition.

Table1. Functional table for DDR2 SDRAM

Command	**RASbar**	**CASbar**	**WEbar**	**CSbar**
NOP/Idle	H	H	H	H
Active	L	H	H	L
Read	H	L	H	L
Write	H	L	L	L
Precharge	L	H	L	L
Refresh	L	L	H	L
MRS/EMRS	L	L	L	L

Fig3. describes the state transition diagram for DDR2 SDRAM. There are 10 states viz., Init, Idle, Refresh, MRS, OCD, Active, Active Power Down, Writing, Reading, Pre-charge. Command set includes ACT, RES, REF, READ, WRITE, PRE and NOP are the command sets to be issued. On power up, the DDR2 module is in the Idle state which means no operation is being performed or scheduled. It is generally during such periods that auto refresh works in the background to keep the memory contents stable. On power up, pre-charge command is applied to the memory module. If precharge command is issued during normal usage, open row of the active

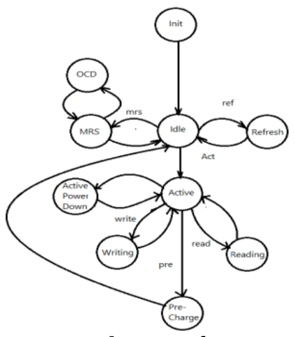


Fig3 State transition diagram for DDR2 SDRAM

bank is deactivated and the new bank is activated. Therefore, the data before and after the precharge command will not be consistent. Once all banks are pre charged, the memory module enters idle state. Then an EMRS command should be issued to enable different control signal of DDR2, e.g., DLL reset, burst length, and burst Mode etc. Next a Refresh command should be issued. This command is used to refresh data in selected bank. After precharge is done, one of the banks has to be activated. This is done by issuing a Active command. This command opens a particular bank from the memory array. At this point, the memory can be read or written. Initially, memory should be written before read. However this is not true for other kinds of memory e.g. flash RAM. At the end pre- charge command should be issued to deactivate the current open bank and to return the device in idle state.

Implementing the complete memory module requires tri state buffers to connect to the bus, address decoders to connect to the correct memory location, internal clock and a mechanism like DLL to synchronize with the external clock. Most of these are analog in nature and hence cannot be easily converted to HDL code. Only maximum delay timings for such operations are assumed.

IV. IMPLEMENTATION

Implementation of the DDR 2 IP is consists of implementing interface commands in accordance with the FSM and timing details. We have described a couple of these command implementations in this section. Other commands are similarly implemented.

4.1. Refresh command

Refresh means the Reading a data from particular address location and again rewriting data on the same address location without changing its original value. This step will restore lost charge from the selected memory location. Each location has to be refreshed within a manufacturer specified limit. This is often 64 milliseconds for DDR2. Refresh command can be invoked automatically or by issuing specific refresh command from the controller. Part of the refresh address is generated internally.

To execute the refresh command for 8 burst cycles, we have to set the registers Refresh_counter and Refresh_backlog_counter to 0 and 8. These counters are incremented, decremented respectively till refresh is complete in 8 clock cycles. In a non-pipelined implementation, the input commands have to be suppressed till the refresh completes execution. In other cases, one or two commands may be buffered while refresh command is in progress. FSM state register is set to Refresh State, which automatically disables incoming commands.

```
//Register Initialization
 D_Refresh_reg = D_RAS
 D_Refresh_reg=D_Refresh_counter;
 D_Refresh_counter = 0;
 D_Refresh_backlog_counter = 8;
 D_Refresh_reg1 = D_Refresh_reg
 D_cState = D_sRefresh;
  //local variable initialization
 ignoreInput = 1'b1;
  blockFlowDown = 1'b1;
```

To implement refresh command we need a register and a counter. As mentioned above description, refresh command is use to rewriting the data without modifying its originality. Refresh_reg1 is loaded Refresh_reg in the memory, Refresh_counter is loaded with the count value of 8 bit, and Refresh_backlog_counter to count 8. While loading the data in Refresh_reg1, Refresh_counter should be decrement by one and when it reaches to zero Refresh backlog_counter is incremented by one to reload the data from Refresh_reg1 to Refresh_reg.

4.2. Bank active command

Bank active command is used to open a particular Bank, and this is applied before Read or Write operation Bank active command is use to open a row address and the value of BS[2:0] selects the bank and the value of A[12:0] selects the row.

```
D_BSReg =[2:0] D_BS;
D_RowReg =[12:0] D_ADDR;
```

To implement bank active command we have used temporary register BS_reg. this register is use to load the bank address to open a particular bank from the memory cell array.

4.3. Precharge command

Precharge command is used to close the current opened bank to active new bank for the next operation. To start new operation precharge is must. because it will bring the system in idle state.

```
ignoreInput = 1'b1;
 D_RowReg = D_ADDR;
 D_cSCount = 1'bx;
 D_maddr [12:0] = D_ADDR;
 D_cState = D_sPreCharge;
```

4.4. MRS command

The Mode register is used to define the specific mode of DDR SDRAM operation, including the selection of burst length, burst type, CAS latency, and operating mode. The load mode register command is loaded with the address input ADDR.

```
D_ColReg [12:0] = D_ADDR;
D_BSReg = D_BS;
D_BurstLength = D_ADDR [2:0];
D_BurstType = D_ ADDR [3];
D_CASLatency = D_ADDR [6:4];
D_TestMode = D_ADDR [7];
D_DLL_reset = D_ADDR [8];
```

To implement MRS command BS_reg is loaded with the ADDR value and depending upon the address input Burst length, burst type, CAS latency has implemented. Following term shows the addresses which are required to implement MRS command

BurstLength = ADDR[2:0]
BurstType = ADDR[3]
CASLatency = ADDR[6:4]
TestMode = ADDR[7]
DLL_reset = ADDR[8]
Write recovery time = ADDR[11:9]

4.5. Read command

The Read command is used to initiate a read access to an active row. Read command is implemented by holding CSbar, CASbar at low and by pushing RASbar and WEbar at high at the CLK. DDR2 is having one special function that is CAS Latency. CAS latency is the delay occurred between the access the read command and actual output operation occurs that is output present at the data bus. Here Read command is entering at '1' Clock, but the stored data has read after clock cycle of '3'.

```
if (D_cState == D_sReading) begin
  if (D_cSCount < D_CASLatency) begin
    D_cSCount ++;
    end
  else begin//equal or greater
    if (D_burstCount < 8) begin
      D_data_out
        =X[D_BSReg][D_RowReg][D_maddr];
      D_cSCount = D_cSCount + 1;
      D_burstCount = D_burstCount + 1;
      D_maddr = D_maddr + 1;
      end
    end
  end
```

To implement read command, cSCount is initialized to zero; CASLatency is initialized to three and burst count to 8. When burstCount is three then data will write on a given address location as data is writing Burstcount is getting incrementing by 1and maddr also by one. 1

4.6. Write command

Write command is used to store data on a given address location given by the column address of the active bank. Write command is implemented by holding CSbar, CASbar and WEbar at low and by pushing RASbar at high at the clock. Write command is completed with the Write latency, write latency is the delay occurs in the clock cycle between the actual command is activated and the data is writing on the data bus.

```
if (D_cState == D_sWriting ) begin
  if (D_cSCount < D_WriteLatency)begin
    D_cSCount = D_cSCount + 1;
    D_burstCount = 0;
    end
  else begin
    if (D_burstCount < 8) begin
      X[D_BSReg][D_RowReg][D_maddr]
        =D_data_in;
      D_burstCount = D_burstCount + 1;
      D_cSCount = D_cSCount + 1;#1
      D_maddr = D_maddr + 1;
      end
    end
  end
end
```

V. SIMULATION RESULTS

The proposed system is implemented by using verilog language using Icarus verilog software [5] and waveforms are viewed on GTKWave platform [6].

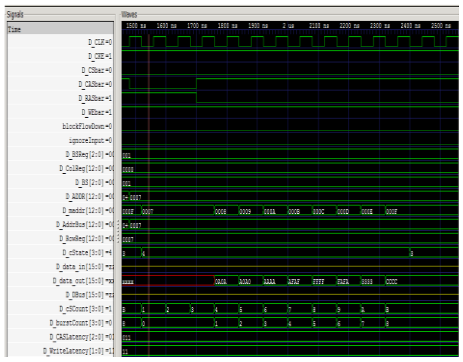


Fig 4 Simulation result for write command

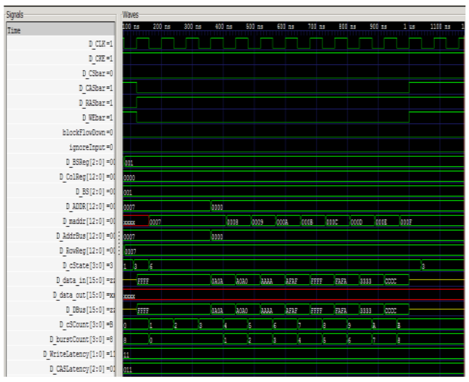


Fig 5 Simulation result for read command

Fig4. Shows the simulation result for write command. Write command is implemented by holding CSbar, CASbar and WEbar at low and by pushing RASbar at high simultaneously at the clock edge.

Fig5. shows the simulation result of read command. Read command is implemented by holding CSbar, CASbar at low and by pushing RASbar and WEbar at high at the clock.

VI. CONCLUSION

The paper present the implementation of the DDR2 SDRAM commands sets. Some of the command implementation results are shown graphically. The proposed system is used for high speed data transmission and storage purpose. The proposed method has been implemented successfully on Icarus Verilog simulation tool.

ACKNOWLEDGEMENTS

I thank the management, Principal, HOD and staff of VLSI and EC Department, K.L.E. Dr.M.S. Sheshgiri College of engineering and technology, Belagavi, Karnataka, India and my special thanks to C-Quad Research, Belgaum for encouraging me to take up this work.

REFERENCES

[1] JEDEC Standard DDR2 SDRAM Specifications, JESD79-2B (JEDEC Solid State Technology Association), 2005

[2] Transcend Corporation, “Difference between SDRAM, DDR SDRAM, DDR2 SDRAM,” Available online at https://www.transcendinfo. com/Support/FAQ-296

[3] Priyanka Bibay, Anil Kumar Sahu, Vivek Kumar Chandra “Design and Implementation of DDR SDRAM Controller using Verilog” *International Journal of Science and Research (IJSR)*, India Online ISSN: 2319-7064.

[4] TMS320DM357 DMSoC DDR2 Memory Controller, User’s Guide, Literature number SPRUG38, (Texas Instruments), 2008

[5] Ioannis Konstadelias “Icarus Verilog +GTKWave Guide with support for MIPS architecture implementation”, Available from http://iverilog.icarus.com and infserver.inf.uth.gr/~konstadel/resources/Icarus_verilog_GTKWave_guide_pdf

[6] Virtex-6 FPGA Memory interface solution, Xilinx San Jose CA, March 2011, Product Specifications DS186 VI. Issue 7

Prajakta Chandilkar. “Development of DDR2 SDRAM Module Interface Software Core.” International Journal of Engineering Research and Applications (IJERA) , vol. 7, no. 9, 2017, pp. 28–32.