CHAPTER – 5

Design and Implementation DDR SDRAM Controller

Summary

Nowadays the prerequisite of memory is expanding step by step as it has ended up as one of the essential part of our everyday hardware's that we utilize, for example, telephones, PCs, and so forth the principle explanation behind the prominence is its redesigned highlights like fast of operation, simple to design, little in size and subsequently possess insignificant region, enhanced inerterness, and superior. It has gotten to be conceivable because of development controller that is utilized nowadays to give vital flag and charges to the SDRAM to perform getting it done and give exceptional features to the clients. Thus it is particularly important to design a controller for the SDRAM memory to improve its usefulness. So, this section focuses on clarifying the design and usage of SDRAM controller that has been intended for rapid interface with PCI Bus that can deal with immense amount of data and can be utilized for different applications of storage, for example, Big Data. The SDRAM controller composed involves a portion of the essential functionalities like read, write operation, refresh, synchronization, and so forth.

5.1 Introduction to SDRAM

All most all the electronic devices used these days make use of memory element to perform various operations. As we know the memories are used for data storage and need of large data storage is increasing every day. All the electronics equipment’s right starting from video games for kids of high resolution camera or even satellite are in need of memory as per the requirements. This increasing demand of memory for the devices has thrown challenges to the engineers to design something robust and reliable that can be used for all level of engineering. Not only is the usage of memory looked into the event but also the speed of operation is important. Various medical and aerospace electronic devices are used these days that need very efficient, high speed and large memory on board for the application.
In earlier days registers were used as a storage element for the devices as the requirements was small the registers were working very fine for the application. As the days passed and the requirements started increasing array of registers were used to serve the purpose. However, due to continuous increase in the demand the register bank started using more area on chip and also consumed more power and even the latency was increased. Hence, there was an urgent need of some new element to be developed. Therefore, Static RAM (SRAM) and Dynamic RAM (DRAM) were introduced which are the two most important type of memory that are known to use these days. SRAM are generally made up of flip-flops. These have low latency, low memory density and are comparatively expensive. However these memory have excellent operating speed hence, these can be found in all the application which have high speed requirement and cannot be comprised in this area. Various application such as medical and aerospace make use of this kind of memory.

Due to high cost and low density of memory it cannot be used in all the application. Dynamic RAM (DRAM) are comparatively low cost and also have more memory density. But the operating speed of the same is very low and can be used at all the places where the timing is not the matter of concern, application such as video games, Camera and also some small electronic equipment. To improve the speed of DRAM there is need for an additional design that work as a communicator between the device and the DRAM and also provide all the necessary signals for the functionality of the memory element. The mediator is called as controller or memory controller. It will perform all the basic command generation and provide the synchronization between memory and the device under operation.

SDRAM inertness is not inalienably lower (speedier) than non-concurrent DRAM. DDR SDRAM is the most well-known sort utilized because accessibility is easy and cost is less. In order to improve the latency and bandwidth synchronous protocol is used by the designer while developing DRAM based architecture. This makes the whole architecture as a synchronous DRAM (SDRAM). This type of architecture is very well known among the designers as it gives the user various added features such as improved speed, burst access and pipelined features. SDRAM controller can be built as per the application and used by the user as the only task of the controller is to initialize the memory and provide some commands like refresh, Read, Write, etc.
Figure 5.1 shows the generalized block diagram for the memory element connected to the device via memory controller. In order to improve the latency and bandwidth synchronous protocol is use by the designer while developing DRAM based architecture. This makes the whole architecture as a synchronous DRAM (SDRAM). This type of architecture is very well known among the designers as it gives the user various added features such as improved speed, burst access and pipelined features. SDRAM controller can be built as per the application and used by the user as the only task of the controller is to initialize the memory and provide some commands like refresh, Read, Write, etc.

As the microprocessors are getting faster these days the greatest challenge persists in improving the design of a controller to meet the technology. Development of Double data rate is one of the fantastic feature of SDRAM that improves the speed with double data rate and also provides the excellent throughput. For example if the design is working at 160MHz and transfers the data at the same rate based on the frequency in an ordinary SDRAM based design. Then by the use if DDRSDRAM the working frequency is improved to 320 MHz and at the same time the data rate is doubled.
5.2 General description of DDR SDRAM

DDR (Double Data Rate) SDRAM was presented as a trade for SDRAM memory running at bus speeds more than 75MHz. DDR SDRAM is comparable in capacity to the general SDRAM. However, duplicates the transmission capacity of the memory by exchanging information twice per cycle on both edges of the clock signal, actualizing burst mode information exchange. The DDR SDRAM Controller is a parameterized center giving user the adaptability for altering the information widths, burst exchange rates, and CAS dormancy settings of the design. What's more, the DDR center backings shrewd bank administration, which is finished by keeping up a database of "all banks actuated" and the "lines enacted" in every bank. With this data, the DDR SDRAM Controller chooses if a dynamic or pre-charge command is required. This successfully diminishes the inactivity of read/write commands issued to the DDRSDRAM. Since the DDR SDRAM Controller deals with initiating/pre-charging the banks, user just needs to issue straightforward read/write commands.

5.2.1 Read operation in DDR SDRAM

The flow chart shown in fig 5.2 explains the read operation in DDR SDRAM. When read operation is initialized, data is read from the memory at 2X clock cycle, the data is read at both the half clock cycles, which shows data is read at higher speed from the memory.

![Fig 5.2 Flow chart for read operation in DDR SDRAM](image-url)
5.2.2 Write operation in DDR SDRAM

The flow chart shown in fig 5.3 explains the operation of write by DDR SDRAM. When the write command is initialized, first the data is read and stored in the temporary register when a clock occurs, the data from the input is sent to the memory in 2 half clock cycles which gets stored in the memory, the data is not directly written to the memory but it is first stored in the register and then transferred to the memory.

Fig 5.3 Flow chart for write operation in DDR SDRAM

Necessary command set for DDR SDRAM is mentioned in table 5.1 that provides the quick overview of the command which are used for controlling of DDR SDRAM. All the signal work on the rising edge of the clock and are synchronous to the clock. DDR SDRAM is much the same as SDRAM aside from that is has higher data transmission, which means more prominent speed. DDR SDRAM is clock multiplied adaptation of SDRAM, which is supplanting SDRAM. DDR permits data transfer on rising edge and falling edges of the clock cycle.
5.3 Basic Commands of DDR SDRAM Controller:
Controller main operation is to bridge the gap between the device in use and the memory element. It has got functionality to generate signals such as initializing signals, power ON signal, read/write signal, address signals and data signals, etc. as per the requirement of the memory.

<table>
<thead>
<tr>
<th>Command</th>
<th>Sign</th>
<th>CS_N</th>
<th>WE_N</th>
<th>RAS_N</th>
<th>CAS_N</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Operation</td>
<td>NOP</td>
<td>L</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Load Mode Register</td>
<td>LMR</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Precharge</td>
<td>PRE</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>Read</td>
<td>READ</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>H</td>
</tr>
<tr>
<td>Write</td>
<td>WRITE</td>
<td>L</td>
<td>H</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>Active</td>
<td>ACT</td>
<td>L</td>
<td>L</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Auto Refresh</td>
<td>REF</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>H</td>
</tr>
</tbody>
</table>

5.4 Architecture of DDR SDRAM Controller:
The design of DDR SDRAM controller mainly comprises of 3 main modules: Dataflow Module, Control Module and Finite State Machine. They are used for generation of read and write related signals. The modules are shown in fig 5.4

5.4.1 Data Flow Module:
The functionality of the data flow module is to act as a bus interface between the device in operation and SDRAM. The data flow module is shown as one of the block in fig 5.4. It has got some set of registers which are used to take the data from the user interface and transfer it to the memory at 2X clock domain. It is also responsible for taking the data from the controller and transfer to the use at 1X clock cycle. This block is responsible for generation of DQS signal from the controller for the SDRAM for the specific operation. The input for is 16 bit for one clock cycle where as the output is of 32 bits for two clock cycles so, we understand that the output is read at double the rate of input.
5.4.2 Control Module

The main functionality of this module is to take the commands from CMD input and decode it to generate the necessary signals such as READA, REFRESH, PRECHARGE, LOAD, etc. It is also responsible for taking the data from SADDRS and distributing over a set of required registers available in the control module for the necessary operations. The control module is as shown in figure 5.4.

5.4.3 Finite State Machine:

Figure 5.5 shows the FSM developed for the complete design. State Machine is designed to work for the whole architecture it comprises of various states such as read state, Write state, Load mode, refresh and precharge, etc. The FSM takes the decision of flow depending upon the status and commands generated, the necessary commands are explained in 5.4.3.1.
5.4.3.1 Distinctive FSM states:

**Idle:** Activation of reset compels the system to idle without considering the function device is performing at that point of time.

**NOP:** The DDR SDRAM does not perform any operations, so the held operations are stopped form further operation.

**LMR:** When inputs are given, LMR command is generated so execution does not occur.

**Precharge:** The columns that are open in the bank is deactivated in a specific bank.

**Read:** It initiates to start the read access to row or column.

**Write:** It initiates a burst write access to a row or column.

**Active:** The column of a specific bank starts to open for read and write to be performed.
5.4.3.2 Refresh in DDRSDRAM

Periodic refreshing is required for a DDR memory so that information can be hold. Once the Auto Refresh command is initiated, all the banks will be out of state. A SDRAM refresh solicitation is created by enacting sdr_REF_REQ sign of the controller. The sdr_REF_ACK sign will recognize the acknowledgment of sdr_REF_REQ and will be dynamic all through the entire refresh cycle. The sdr_REF_REQ signal must be kept up until the sdr_REF_ACK goes dynamic so as to be perceived as a refresh cycle. None of the system read/write access cycles are permitted when sdr_REF_ACK is dynamic.

All system interface cycles will be disregarded amid this period. The sdr_REF_REQ signal statement should be endless supply of sdr_REF_ACK acknowledge, otherwise another invigorate cycle will again be performed. Endless supply of sdr_REF_REQ attestation, the state machine CMD_FSM enters the c_AR state to issue an AUTO REFRESH charge to the SDRAM. After tRFC time postponement is fulfilled, CMD_FSM comes back to c_idle.

The state machine enters in the NOP mode if no command is given to the design and initializes the design and wait for the next instruction. On getting command it switches to load mode via. Load reg1 or load reg2. Based on the data either coming from memory or data coming from the device.

Depending on read and write operation it enters to the read state and write state and generates the required signal to perform the specified task. Once the task is completed the acknowledgement signal is generated and device is either refreshed or precharged and returned back to the load register state and wait for the next operation to be given as an input.

5.5 Implementation Results of DDR SDRAM controller

The design developed is actualized on FPGA working on 200MHz clock recurrence and coded utilizing Verilog. Xilinx ISE 14.1 is utilized as an advancement domain and Modelsim6.3 is utilized to chip away at the practical confirmation of the design. The snapshots obtained for the developed architecture is shown further.
The design was coded in VERILOG HDL and was synthesized on XILINX ISE 14.1 to obtain the results. Figure 5.6 shows the design summary, we can understand that the design occupy very less numbers of slices and LUTs and hence it was possible implement on XILINX Spartan III FPGA.

From figure 5.7 it is understood that the total device utilization for the DDRSDRAM Controller is only 22% with very less utilization of LUTs just 4%. This is very small indeed. Figure 5 shows the top level RTL schematic generated for the developed code.

All the necessary signals are generated to achieve a very user friendly interface between the device and SDRAM. Behavioral functionality in the next section shows the clear picture of the simulation results obtain for the design.

Fig 5.6 Design summary for the designed DDR SDRAM controller
The designed architecture is verified using behavioral simulation of each and every module based on MODELSIM Simulator 6.3c. All the functional behavioral such as read and write operation has been tested and verified.
Fig 5.9 Timing Summary of SDRAM controller

Fig 5.10 SDRAM command sequence
Fig 5.11  SDRAM write operation

Fig 5.12  SDRAM read operation
5.5.1 Read and Write sequences in DDR SDRAM controller

SDRAM write sequence

Figure 5.11 shows the simulation results for the write operation mode of the controller from the simulation a clear picture is obtained that the designed is working functionally correct by generation the necessary signals for the operation to be done it can also be observed that the double data rate has been achieved by transferring and receiving data at both the edges of the clock achieving the functionality of double data rate SDRAM.

5.6 Conclusion on DDR SDRAM Controller

During the development of the DDRSDRAM controller it is analyzed that the design is bit tricky and complex due to enormous logic and timing constrained involved in it. It is very necessary that the design is developed with all attention. Based on the implementation I can conclude the designed architecture has all the necessary functionality and meets the requirement of DDRSDRAM controller and is also compatible to high speed bus such as PCI Express as per the requirements. The whole design was developed based on XILINX Spartan 3 FPGA family and was tested and implemented on the same for the hardware verification with seamlessly writing and reading operation performed. The design has been verified based on the functional verification and hardware implementation.