Master Thesis

DESIGN AND IMPLEMENTATION OF A DDR SDRAM CONTROLLER FOR SYSTEM ON CHIP

Magnus Själander

Computer Science and Engineering Program

Department of Computer Science and Electrical Engineering

Division of Embedded Internet Systems

2002-12-15

Abstract

The aim of this study was to investigate the different problems associated with the design and implementation of a DDR SDRAM Controller for CMOS technology.

This study has lead to a working implementation of a DDR SDRAM Controller that is meant to be used as a reference for future implementations. The report highlight design issues and propose solutions to problems like data resynchronization and how to phase shift the data strobe.

The result of this study highlights design issues common to any Double Data Rate interface that can be used when designing and implementing applications in need of the higher performance given by a Double Data Rate interface and is not limited to the design and implementation of a DDR SDRAM Controller.

Acknowledgement

I would like to thank the people at the unit of ASIC Technology & System on Chip at Ericsson AB in Mölndal for all the support I received. Especially I want to thank my supervisor Fredrik Johansson for all the given support. Further I want to thank Martin Johansson and Tomas Sand for the help with the backend work and discussions about various timing problems.

I also would like to thank Sten Gunnarsson for all the interesting conversations about everything from analog filters and mixers to rock climbing and hiking that together with our weekly climbing made the time in Göteborg that much more interesting.

Finally I would like to thank Per Lindgren acting as my examiner at Luleå University of Technology.

Göteborg December 2002

Magnus Själander

Index

1 INTRODUCTION 9

PART I

2 DOUBLE DATA RATE SOURCE-SYNCHRONOUS INTERFACES 13

3 DDR SDRAM 15

3.1 ARCHITECTURE 15
3.1.1 A closer look on a read and write operation 15
3.1.2 Why is the memory dynamic and needs to be refreshed 17
3.1.3 Frequency considerations of SDR and DDR SDRAM 17
3.2 COMMANDS 20
3.2.1 Activation and Precharge 20
3.2.2 Read 20
3.2.3 Write 20
3.2.4 Refresh 21
3.2.5 Mode Register Set 22
3.2.6 Extended Mode Register Set 23
3.3 INITIALIZATION 23
3.4 ADDRESSING 24

4 HIGH SPEED LOW VOLTAGE SWING BUS 25

4.1 REFLECTIONS 25
4.2 INPUT BUFFER 26
4.3 DIFFERENTIAL SIGNALS 26
4.4 TRISTATE SIGNALS 28

PART II

5 DDR SDRAM CONTROLLER 31

5.1 CORE MEMORY CONTROLLER 32
5.1.1 Current and Next Address 32
5.1.2 Open Banks 32
5.1.3 Read Write Command 33
5.1.4 Command Timing 33
5.1.5 Address Handling 34
5.1.6 Crossing a Row Boundary 35
5.2 AHB INTERFACE 36
5.2.1 AHB Core 36
5.2.2 Data Buffer 37
5.2.3 x2 37
5.3 APB 39
5.4 ARBITER 40

6 SOURCE RESYNCHRONIZATION 41

6.1 PHASE SHIFT THE DATA STROBE 41
6.1.1 Delay Lock Loop 41
6.1.2 Inverter Delay 41
6.1.3 PCB Line Delay 42
6.1.4 Programmable Delay Line with Temperature sensing 42
6.2 SYNCHRONIES THE DATA 43
6.2.1 Simplified Phase Detector 44
6.2.2 Alternative method 46

7 FUNCTIONAL SIMULATION 49

7.1 PERFORMANCE 49
7.2 DDR SDRAM SIMULATION MODELS 50

8 BACKEND 51

8.1 SYNTHESIS 51
8.2 FLOORPLAN 51
8.3 PLACEMENT 53
8.4 TIMING PROBLEMS 54

PART III

9 NEW DRAM INTERFACES 57

9.1 DDR II SDRAM 57
9.2 ESDRAM 57
9.3 QDR SDRAM 57
9.4 DIRECT RAMBUS 58
9.5 FCDRAM 58

10 FUTURE WORK 59

11 CONCLUSION 61

12 REFERENCES 63

APPENDIX 65

A FIGURES 67
B TERMINOLOGIES 69
C DDR SDRAM COMMANDS 71
D PLACE AND ROUTE STATISTICS 73
E VHDL CODE 75

1 Introduction

The demand for ever higher performance with highly integrated System on Chips that can consist of a number of processing units like DSP and CPU cores has lead to an increased demand in memory bandwidth to of chip memory.

This demand has been recognized by the unit of ASIC Technology & System on Chip within Ericsson AB and has in cooperation with EISLAB at Luleå University of Technology conducted an investigation in the form of a master thesis in computer engineering to explore the possibility for the DDR SDRAM standard [1] to meet up to Ericsson AB needs of memory bandwidth for their future system platforms to be used in the telecom industry. The goal was to investigate if it is possible to implement a DDR SDRAM Controller as an Intellectual Property core (IP core). For the DDR SDRAM Controller to be general and possible to use as an IP core for System on Chip production it was decided that the DDR SDRAM Controller would be based on an AMBA Advanced High-speed Bus interface [2] for on chip communication.

The reason to investigate the possibilities of the DDR SDRAM standard to be able to meet Ericsson AB's need for increased bandwidth is that DDR SDRAM has become a commodity used for every day desktop computers which have lead to an attractive price performance relation. Further the DDR SDRAM has a significant impact on bandwidth performance compared to the standard SDRAM solutions that Ericsson AB's system platforms have been based on up to date. The reason to base the DDR SDRAM Controller on an AMBA Advanced High-speed Bus interface is because it is an acclaimed industry standard that is widely used as within Ericsson AB.

I

2 DOUBLE DATA RATE SOURCE-SYNCHRONOUS INTERFACES 13

3 DDR SDRAM 15

3.1 ARCHITECTURE 15
3.1.1 A closer look on a read and write operation 15
3.1.2 Why is the memory dynamic and needs to be refreshed 17
3.1.3 Frequency considerations of SDR and DDR SDRAM 17
3.2 COMMANDS 20
3.2.1 Activation and Precharge 20
3.2.2 Read 20
3.2.3 Write 20
3.2.4 Refresh 21
3.2.5 Mode Register Set 22
3.2.6 Extended Mode Register Set 23
3.3 INITIALIZATION 23
3.4 ADDRESSING 24

4 HIGH SPEED LOW VOLTAGE SWING BUS 25

4.1 REFLECTIONS 25
4.2 INPUT BUFFER 26
4.3 DIFFERENTIAL SIGNALS 26
4.4 TRISTATE SIGNALS 28

2 Double Data Rate Source-synchronous Interfaces

Double Data Rate (DDR) interfaces are becoming increasingly common in the ASIC world. A DDR interface is a type of source-synchronous interface meaning that the clock is sent along with the data from the transmitting device.

With traditional interfaces data will switch value either at rising edge or falling edge of the clock but not at both, this is called single data rate (SDR) and the data lasts for one full clock cycle. DDR interfaces allow data to be transferred at both rising edge and falling edge, thus providing twice the bandwidth of a SDR interface. Since data is transferred at both rising and falling edge the data only lasts for one half of a clock cycle, therefore DDR interfaces demands stricter timing requirements compared to SDR interfaces using the same clock frequency.

The data strobe give a number of benefits compared to using a standard synchronous interface [3]. A standard synchronous interface is limited to a time of flight between to ICs of one clock period while a source-synchronous interface has no time of flight limits. A standard synchronous interface also has to deal with clock skews between the transceiver and receiver ICs but since a source-synchronous interface transmits the data strobe together with the transmitted data there is no skew. With a source-synchronous interface there is a synchronization challenge when communicating on the same bus with more than one IC. Every IC will transmit its own data strobe and data strobes from different ICs do not have to be in synchronization with each other. The receiving IC has to be able to sample data using the different data strobes making it challenging to design a resynchronization input buffer. The resynchronization buffer is used for sampling the data bus and synchronizes the data to the internal clock of the transceiver.

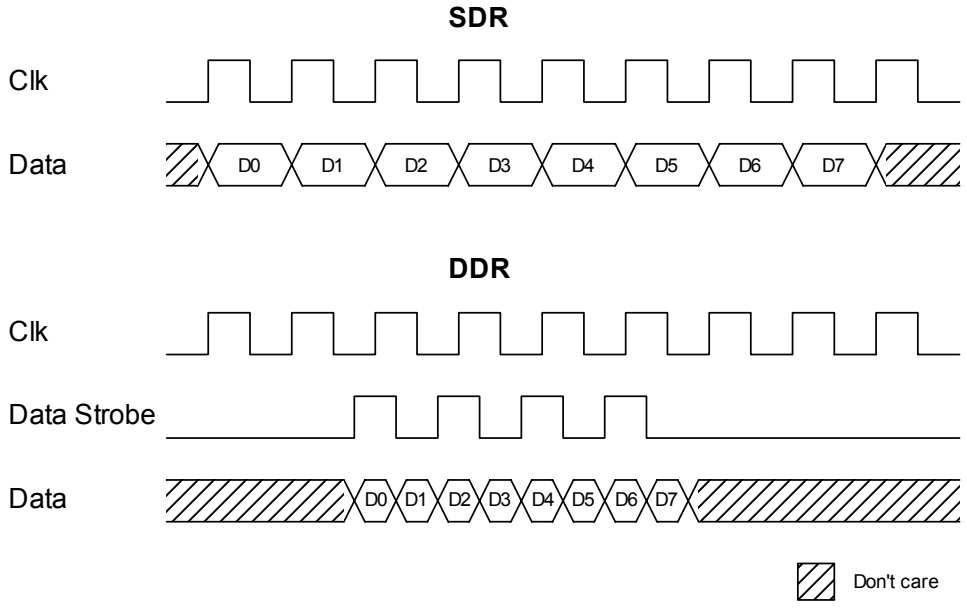

Figure 1 Comparison of an eight beat burst with SDR and DDR

3 DDR SDRAM

As with standard SDRAM the architecture is pipelined and consists of multiple banks allowing concurrent operation and thereby providing high effective bandwidth [4], [5] and [6]. All reads and writes are burst oriented and are programmable to burst lengths of 2, 4 or 8 beats (other vendor specific burst lengths might exist).

A DDR SDRAM is divided into four banks where each bank consists of a number of rows. A row is then divided into columns which each contains 32 bits of data. The number of rows and columns are dependent on the size of the DDR SDRAM and the internal organization. The organization with four banks makes it possible to issue one command to every bank. It is not possible though to make more than one read or write operation at a single time since the data bus only can handle data for one operation. It is possible to while a read or write burst is in progress in one bank to precharge and activate a row in one of the other banks.

3.1 Architecture

The memory array of the DDR SDRAM is divided into four equally large memory banks. Each memory bank consists of a number of row and column select lines [7]. At each crossing of a row and column select line there exist a transistor and a capacitor (M_1 and C_s Figure 2) called a 1T memory cell since there is only one transistor for every bit of information stored. Each capacitor is capable of storing one bit of information and the transistor is used for accessing the data stored in the capacitor. Before any read or write operations can be performed a row has to be activated within one of the banks. The activation is done by first decoding the row address into a word line which then turns on all the transistors on that row and the data of those memory cells are read out to the sense amplifiers. When the data have been read out to the sense amplifiers the first read or write operation can be done by connecting the sense amplifiers to the data lines by decoding the column address. Before an access can be done to another row the bank first has to be precharged.

3.1.1 A closer look on a read and write operation

To be able to read out the data from a row the bit line, same as the column line, has to be precharged to half the VDD voltage. When the bit line have been precharged the word line is activated which turns the transistor M_1 on and the charge in the capacitor C_S is redistributed between C_S and C_{BL}. C_{BL} is the parasitic capacitance of the bit line. The redistribution of the charge gives a change of the voltage on the bit line, which can be detected by the sense amplifier [8]. The sense amplifier is a voltage differential amplifier with positive feedback that uses an inactive bit line BL^* that also have been precharged to half the VDD that is used as a reference voltage which the voltage on the bit line with the memory cell is compared to. If the memory cell contains a zero, that means that the capacitor C_S does not contain any charges, then when the transistor M_1 is turned on charges from the bit line represented as C_{BL} will be redistributed between the C_S capacitor and C_{BL} capacitor. This will lower the voltage on the bit line and the sense amplifier will notice that the bit line voltage is lower than the reference voltage and drives the voltage on the bit line towards ground level because of the positive feed back of the sense amplifier. If the memory cell on the other hand contains a one meaning that the capacitor C_S is fully charged then when the transistor M_1 is turned on the charges will be redistributed resulting in a net increase of the charges on the bit line and a decrease in the charges of the C_s capacitor. This will raise the voltage on the bit line and the sense amplifier will drive the bit line towards VDD.

During a write operation the sense amplifier are forced to change its state by either driving the bit line to ground or VDD depending on if the data to be written is a zero or one. The data held by the sense amplifiers are propagated along the bit lines to the capacitors of the memory cells that will be charged or discharged according to the data to be stored. When the row is being precharged the transistor M_1 are turned of which stores the charge in the capacitors.

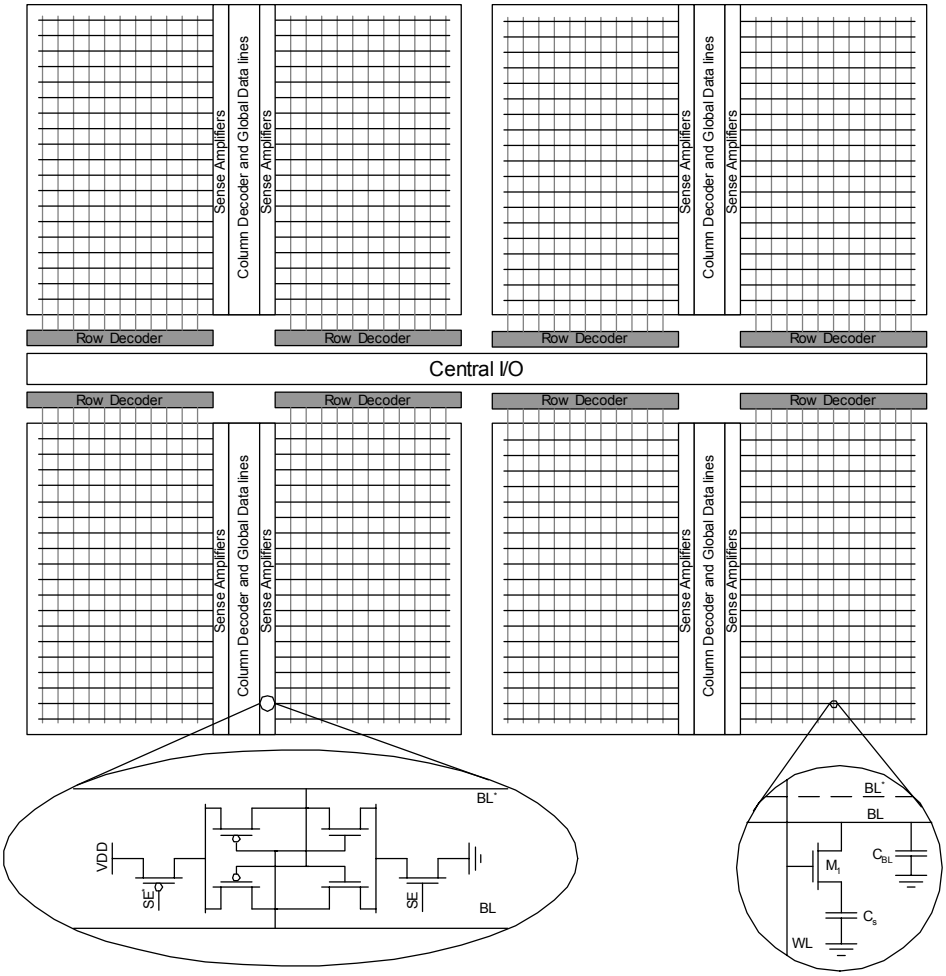

Figure 2 Internal architecture of a SDRAM with a sense amplifier and memory cell

The architecture and type of sense amplifier shown in Figure 2 is only one example of how a DDR SDRAM can be constructed. In an actual DDR SDRAM there exists sense amplifiers interleaved with the memory array between the row select lines, which is not being shown.

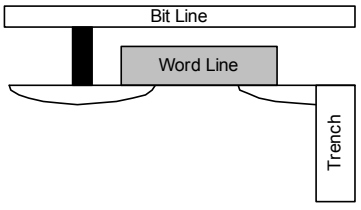

Figure 3 Memory cell in cross section

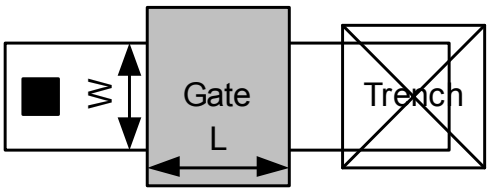

Figure 4 Memory cell in top view

Figure 3 and Figure 4 shows an example on how a memory cell can be constructed in silicon. The trench works as the capacitor C_s and are capable of storing charges. The word line crosses over the gate of the transistor and the bit line is connected through a VIA, a connection between the different layers of the Die, to the source. The drain of the transistor is in direct contact with the trench.

3.1.2 Why is the memory dynamic and needs to be refreshed

The transistor M_1 can never be made ideal and will always have a current leakage. The current leakage will eventually corrupt the data stored in the capacitor C_S if nothing is being made to prevent loss of data. To prevent loss of data the DDR SDRAM has to be refreshed within a certain period of time which is defined by how fast the voltage level of the memory cells capacitor alters because of the current leakage through the access transistor. The refresh cycle is usual within a couple of microseconds. When the DDR SDRAM is being refreshed the information of the memory cells is read out and at the same time written back thus restoring the voltage level of the memory cells capacitor to its previous value of VDD or ground.

3.1.3 Frequency considerations of SDR and DDR SDRAM

In a Single Data Rate SDRAM the data is read out and then clocked on to the data bus on the next coming rising edge of the clock. This limits the clock period to a minimum of about 7.5 ns for practical performance since the time for the data to be read out is about 7 ns for a standard SDRAM.

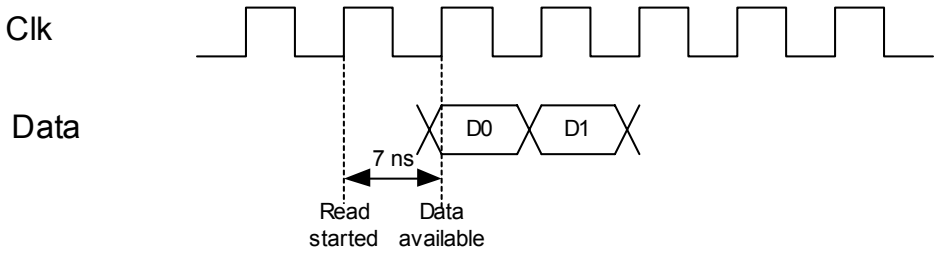

Figure 5 Single Data Rate SDRAM clock period limit

The time to read out the data is limited by the time it takes to decode the column address and propagate the values from the sense amplifiers to the output buffers. This time is not easily reduced and thus is a limiting factor on the clock frequency for standard SDR SDRAM. In the case of the DDR SDRAM this limitation is solved by using a Delay Lock Loop, DLL, to delay the clock signal which is used for clocking the data at the output buffers [9], Figure 6. With the use of the DLL it is possible to clock the DDR SDRAM in a higher frequency than what is possible with the regular SDR SDRAM as shown in Figure 7.

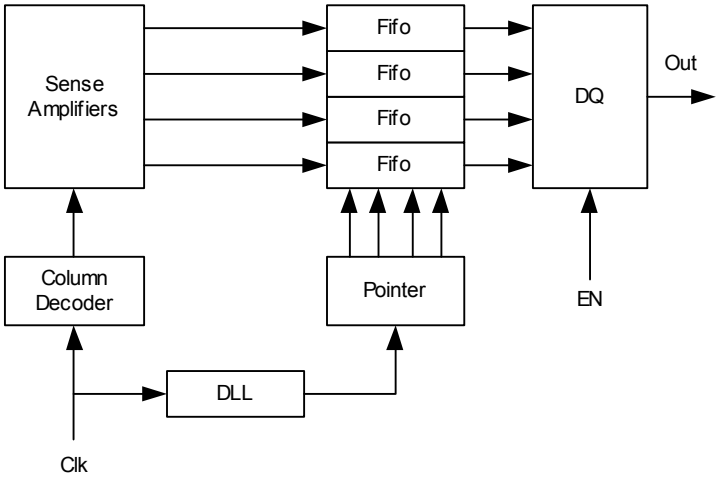

Figure 6 Schematic view of the clock delay in a DDR SDRAM

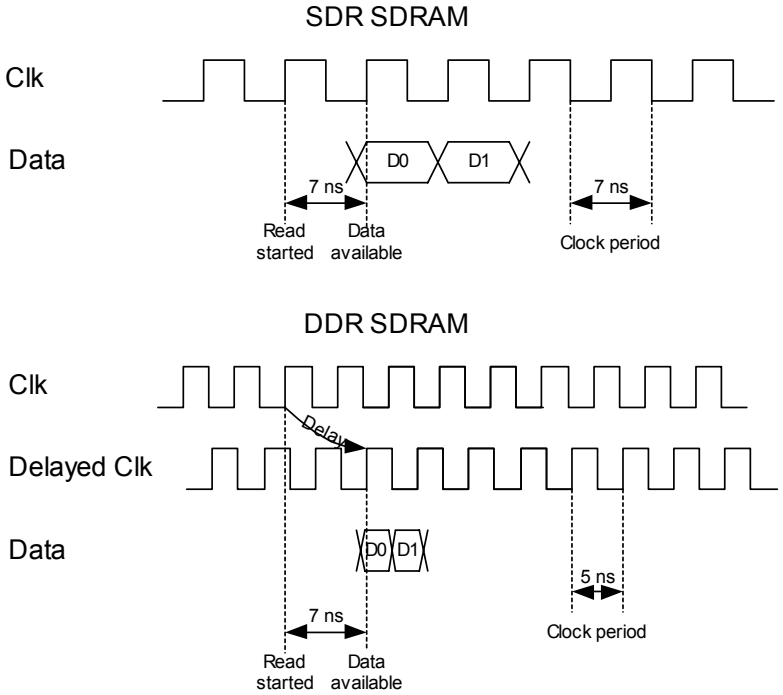

Figure 7 Comparisons of a SDR and a DDR with DLL

For the DDR SDRAM to be able to deliver data at the rapid pace that is needed a method called 2n-prefetch is used [10]. The method is quite simple and is done so that for every rising edge of the clock data is being read out from the current address and the following address. In Figure 8 this can be seen by the 64 bit bus from the sense amplifiers that go to the 2n-prefetch block. The data on the 64 bit bus is then muxed out on the output bus with the first 32 bits on the rising edge and the next 32 bits on the negative edge. As can be seen in Figure 9 the two consecutive operations are overlapping which results in data from more than one operation propagates along the internal bus lines making the timing requirements significant thus demanding the need of the DLL which is capable of delaying the clock with only 100

picoseconds of jitter. A write is done in the same way. Data from two beats are sampled by the data input registers and then written back simultaneously. This technique limits the burst length to a minimum of two beats and the burst length can only be in multiples of two.

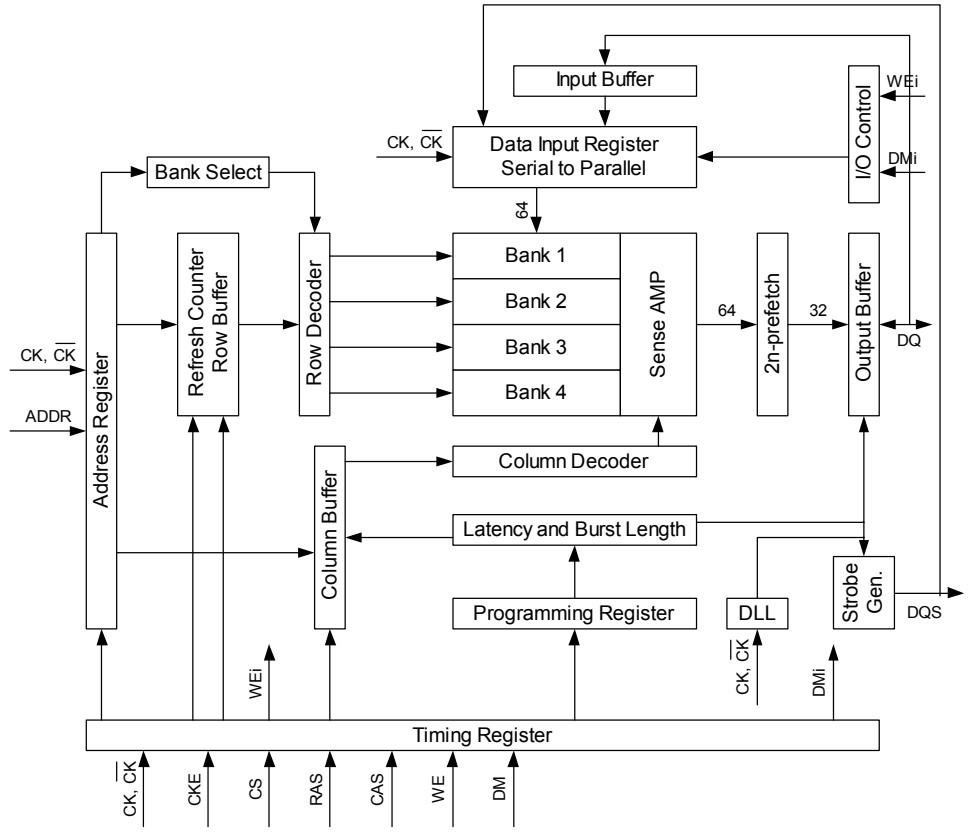

Figure 8 Schematic view of DDR SDRAM architecture

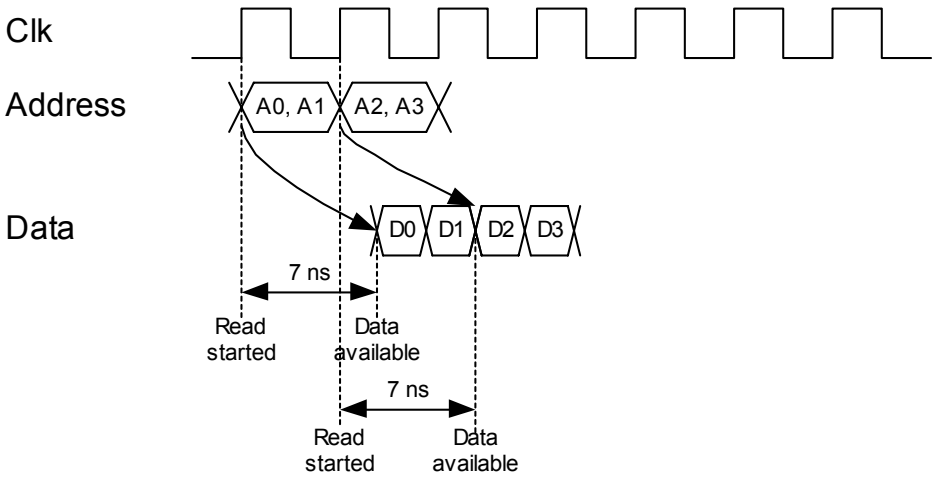

Figure 9 Wave chart illustrating 2n-prefetch

3.2 Commands

The set of commands for the DDR SDRAM is almost identical to its predecessor SDRAM. The Extended Mode Register Set command is the only new command, which is used for controlling the Delay Lock Loop of the DDR SDRAM. All commands are not stated in this report. For a full set of available commands see JESD 79.

3.2.1 Activation and Precharge

Before any read or write operations can be issued the row in the bank to be accessed has to be activated. Activation is done by placing the command on the command bus along with the bank and the row to be activated. After a specific row in a bank has been activated only read and write operations to that row within that bank can be issued. To make a read or write operation to another row within that same bank the currently activated row have to be closed which is done by issuing a precharge on that bank and then activate the row to be accessed. Since there are four banks it is possible to have four activated rows at one time, one active row in each bank.

3.2.2 Read

A read command to an activated row is done by placing the command on the command bus along with the starting column address of the first location to be read. After a specified setup time (called CAS latency) the first location of data is presented on the data bus along with the DDR SDRAM generated data strobe signal. Unless the read burst is terminated or interrupted by another read command then the specified number of locations in the MRS will be presented on the data bus. The data strobe is edge-aligned with the data on the data bus. The DDR SDRAM will drive the data strobe and bus into High-Z if there is no read burst in progress.

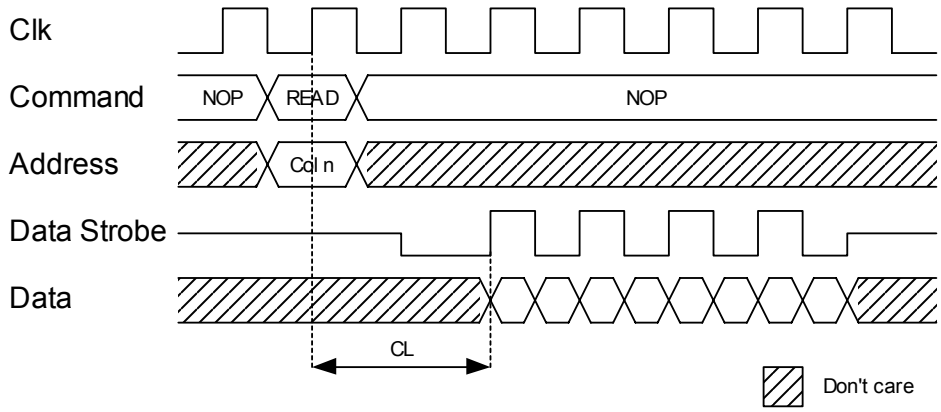

Figure 10 Wave chart of a read burst with eight beats

3.2.3 Write

A write command to an activated row is done by placing the command on the command bus along with the starting column address of the first location to be written. Within a certain time from that the command have been issued the data strobe generated by the memory controller have to make a switch from LOW to HIGH marking the first presented data on the data bus along with the masking signal. The data strobe is center-aligned with the data on the data bus making it possible for the DDR SDRAM to sample the data on the edges of the data strobe.

The data mask is for masking out which byte(s) of the presented data that is to be stored in the memory. The memory controller will drive the data strobe and bus into High-Z if there is not a write burst in progress.

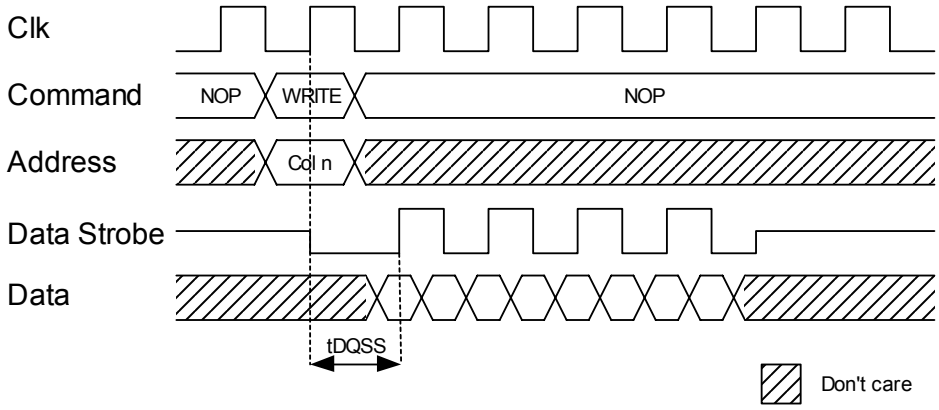

Figure 11 Wave chart of a write burst with eight beats

3.2.4 Refresh

The DDR SDRAM is a dynamic memory and has to be refreshed with regular intervals to recharge the capacitances that are representing the stored data. The longest interval between two refreshes is defined by the size of the memory but is never shorter than 7.8 us. To improve the efficiency in scheduling and switching between tasks some flexibility in the absolute refresh interval is provided. A maximum of eight refresh commands can be posted to any given DDR SDRAM and the maximum absolute interval between two refresh commands is eight times the defined refresh period for the DDR SDRAM used. Before a refresh command can be applied all rows have to be closed which can be done by applying a Precharge All command before the Refresh command is issued.

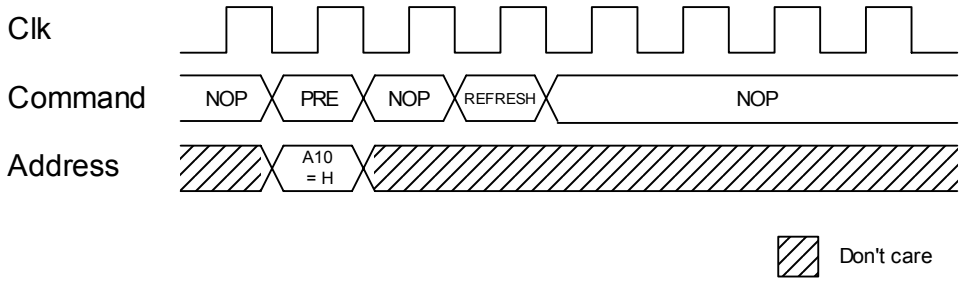

Figure 12 Wave chart of a refresh

3.2.5 Mode Register Set

The Mode Register Set command is for writing data to the Mode Register in the DDR SDRAM which defines the specific mode of operation of the DDR SDRAM. For a full description of the Mode Register see Figure 13.

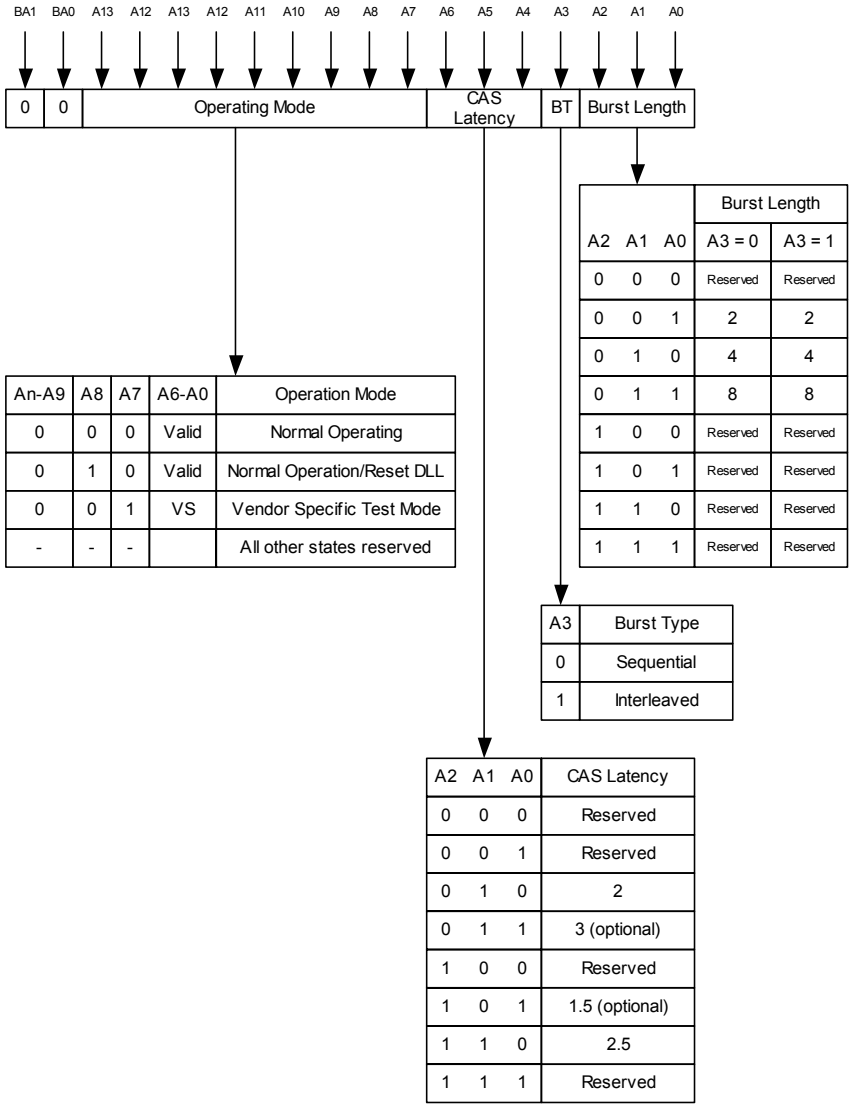

Figure 13 Mode Register definition

3.2.6 Extended Mode Register Set

The Extended Mode Register Set command is for writing data to the Extended Mode Register in the DDR SDRAM which control functions such as the Delay Lock Loop. For a full description of the Extended Mode Register see Figure 14.

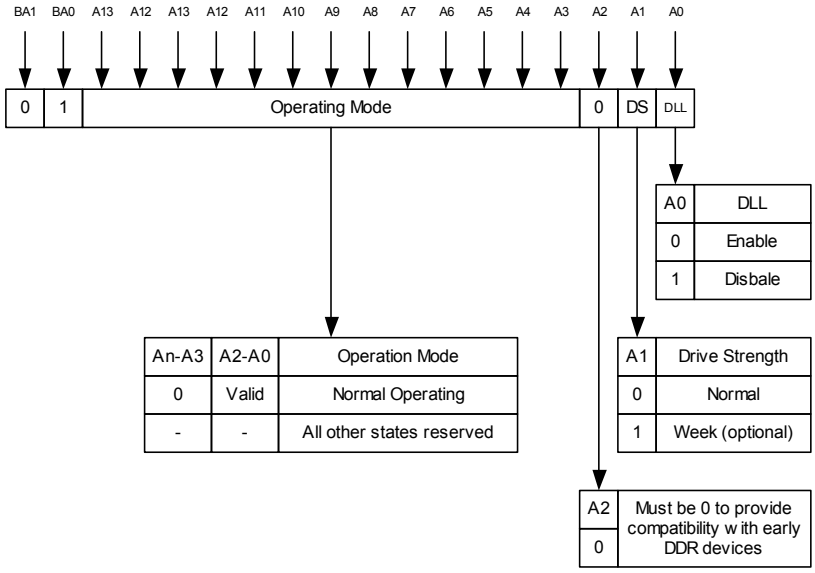

Figure 14 Extended Mode Register definition

3.3 Initialization

The DDR SDRAM has to be powered up and initialized in a predefined manner to assure accurate performance of the memory. The powering up consists of a predefined order to assert the different voltages to the memory (exact way can be found on page 7 of the JESD 79 specification). During the power up sequence CKE must be set to LOW to guarantee that the DQ and DQS outputs will be in High-Z state. After all power supply and reference voltages are stable as well as the clock the DDR SDRAM requires a 200 μs delay prior to applying any executable commands.

Once the 200 μs delay has been satisfied the initialization of the DDR SDRAM can start. The initialization starts with applying a NOP or DSELECT command and raising CKE. Following the NOP is a PRECHARGE ALL command and next an EXTENDED MODE REGISTER SET (EMRS) command to enable the internal DLL. Then a MODE REGISTER SET (MRS) command is applied to reset the DLL and to program the operating parameters of the DDR SDRAM. From that the DLL is reset it has to pass 200 clock cycles before any READ commands are applied (there exists vendor specific requirements there a bank can not be activated within these 200 clock cycles). When the DLL have been reset a PRECHARGE ALL command should be applied to place the DDR SDRAM in the "all banks idle state". Two AUTO refresh cycles have to be performed followed by a MRS where the bit for DLL reset is deactivated. Following this the DDR SDRAM will be ready for normal operation.

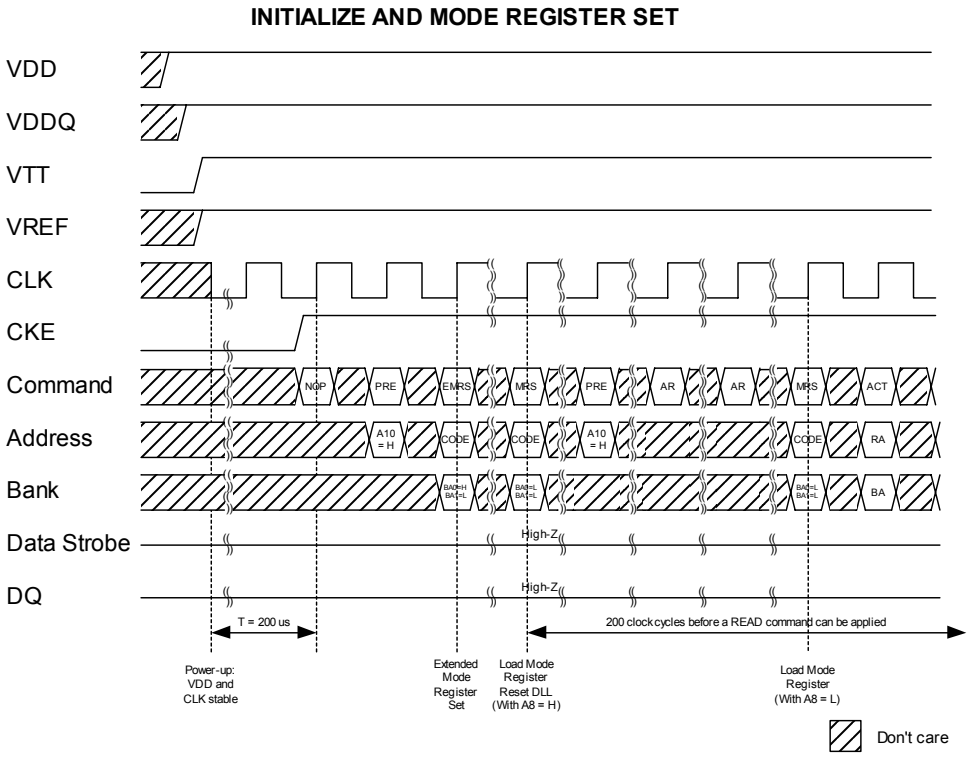

Figure 15 Wave chart of the Initialization of the DDR SDRAM

3.4 Addressing

The addressing of the DDR SDRAM can be either sequential or interleaved. The sequential addressing wraps within the default burst length. So if the default burst length is four the two least significant bits of the address will wrap. So when the two least significant bits are "11" the address wraps and the two least significant bits become "00" without the rest of the address changing.

4 High Speed Low Voltage Swing Bus

All I/O buffers of a DDR SDRAM are compatible with the SSTL_2 standard (Stub Series Terminated Logic for 2.5 V) [11]. SSTL_2 is designed to improve the signal integrity and to provide means to transmit data at a rate of more than 400 Mbps per I/O buffer. The standard is particularly intended to improve operation in situations where busses must be isolated from relatively large stubs. The stubs are isolated by resistors which also reduce the on-chip power dissipation of the drivers.

4.1 Reflections

For all signal transmissions where the wave length of the transmitted signal is shorter or close to the length of the wire, which the signal is transmitted on, reflections to some extent will occur. Any reflection will contribute to the distortion of the signal and if the reflection is not minimized it can come to corrupt the whole signal. To minimize the reflection the designer always strives to make the signal path to be matched. A matched signal path means that at all cross sections of the signal path the resistance in both directions is equal, Figure 16.

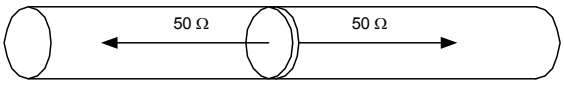

Figure 16 Matched signal path

A common method to match a wire to the resistance of the output and input buffer is to use stubs which is a wire with the length equal to a quarter of the wavelength of the signal to be transmitted. The stub has to be placed a certain distance from the input and output buffer. The exact workings of a stub and how to design it is beyond the scope of this report [12]. In the ideal case a stub will cancel out all reflections for the particular frequency it is designed for. In any practical case there will always be reflections to some extent. To make the situation even worse the square wave used for digital transmission is built up of a numerous number of sinus waves where only one frequency can be matched with the use of stubs. The ground sinus wave is the most dominant and therefore matched with the use of stubs. To reduce the distortion on the signal by the reflections the signal path with the stubs can be isolated by using a resistor in series with the wire, R_S in Figure 17. The resistor will dampen the signal and thus making the reflections dissipate faster.

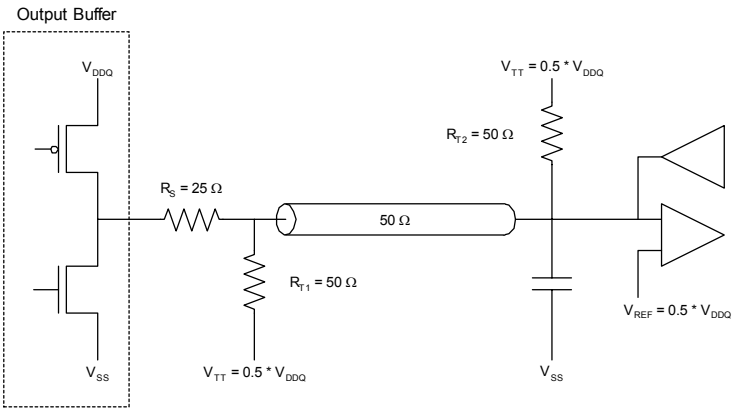

Figure 17 Example of a SSTL_2, Class II, output environment

4.2 Input Buffer

The input buffer is constructed of a differential amplifier with a reference voltage to which the input signal is compared. The differential amplifier works such that the difference between the reference voltage and the input voltage is amplified. By choosing a large amplification the amplifier will be driven into saturation even at small differences between the input voltage and the reference voltage. If the input voltage is lower than the reference voltage the difference is negative and the output will be driven to ground, if the amplification is large enough. In the other case where the input voltage is higher than the reference voltage the difference will be positive and the output will be driven to V_{DDQ}, if the amplification is large enough. The use of a reference voltage and an amplifier makes it possible to with even small voltage swings detect a low respectively high level. A small voltage swing demands strict conditions on the reference voltage and its variations. The reference voltage is expected to be within 49 to 51 percent of V_{DDQ} and to be able to track variations in V_{DDQ}. The minimum voltage swing at the input buffer is V_{REF} +/- 0.31 V as can be seen in Figure 18.

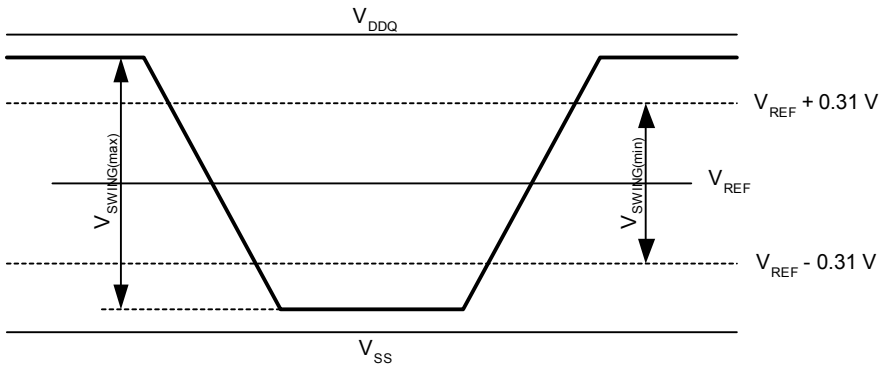

Figure 18 Voltage swing

4.3 Differential Signals

A common problem when dealing with digital signals is that the positive flank of the signal has a lower slew rate than the negative flank. This causes a distortion on the signal with the time when the signal is high being shorter than the time when it is low as can be seen in Figure 19.

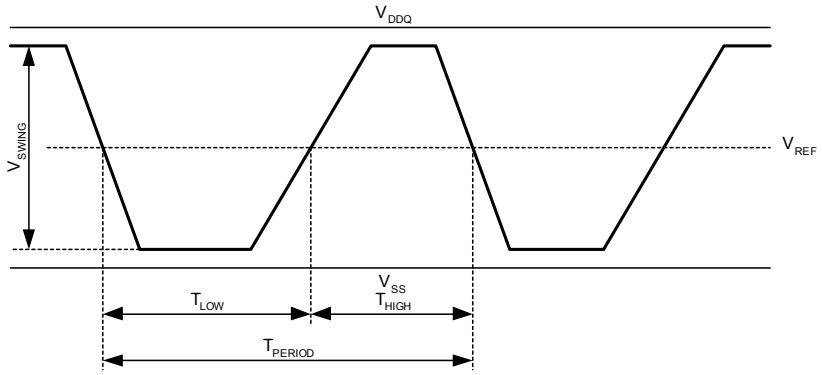

Figure 19 Illustration of the effect of different slew rate for positive and negative flank

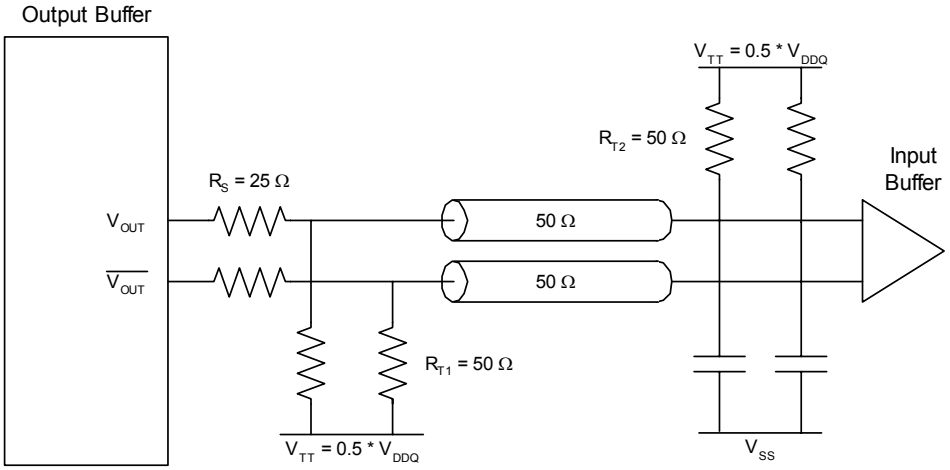

Figure 20 Example of a SSTL_2, Class II, Differential signals

To come to terms with this problem it is possible to use differential signals where together with the transmission of the original signal is also a negated version of the signal, Figure 20. The negated signal is then used as the reference voltage for the differential amplifier giving an equally long high as low period, Figure 21.

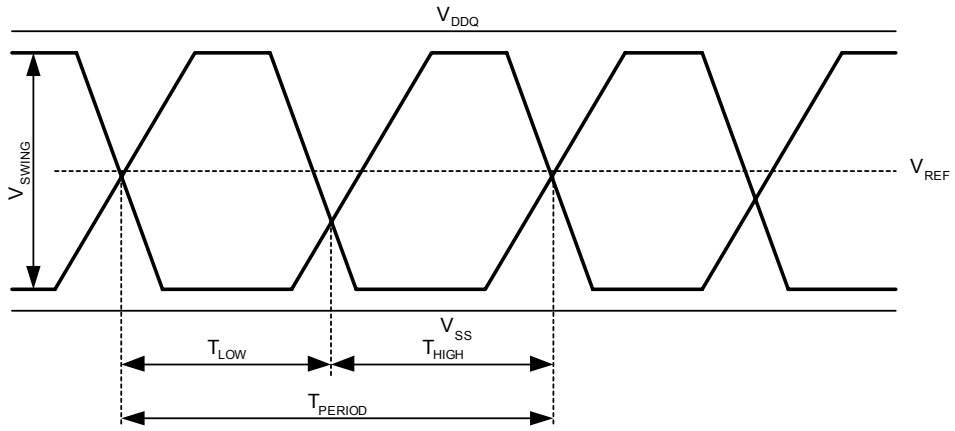

Figure 21 Differential signal wave chart

The resistor values in Figure 17 and Figure 20 are only an example. Actual values are a system design decision.

4.4 Tristate Signals

Signals that are in tristate, High-Z, leaves the voltage level on the bus equal to that of V_{REF} making the bus sensitive to noise since even small amounts of noise changes the voltage level on the bus which can drive the Input Buffer to either a high or low level for shorter periods of time. To avoid mixing the interference of noise with a real signal the DDR SDRAM uses a preamble and post amble on the data strobe indicating that a valid signal is transmitted. The preamble exists of a low voltage level for the duration of one full clock cycle and the post amble is a low voltage level for the duration of a half clock cycle. By looking for a preamble it is then possible to know when a valid signal is being received. If a low level is detected on the data strobe the receive logic is turned on. To further improve the possibility of not reacting to a false signal is to start looking for a preamble only when it is known that there will come one. In the DDR SDRAM case it is known that a preamble will arrive a certain time after a read command has been issued. The time is dependent on the CAS-latency and the time of flight for the signals back and forth between the DDR SDRAM Controller and the DDR SDRAM.

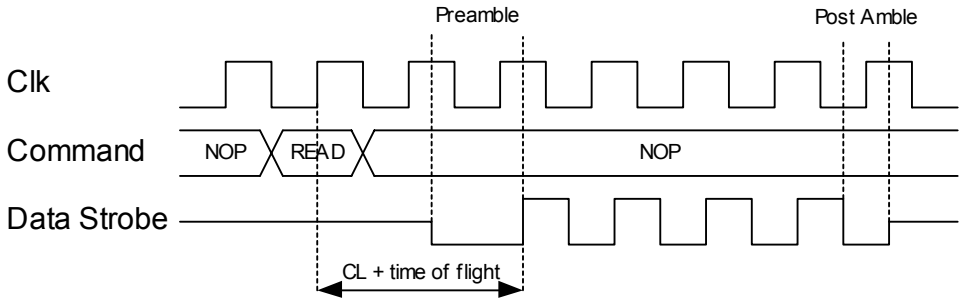

Figure 22 Pre and Post amble

II

5 DDR SDRAM CONTROLLER 31

5.1 CORE MEMORY CONTROLLER 32
5.1.1 Current and Next Address 32
5.1.2 Open Banks 32
5.1.3 Read Write Command 33
5.1.4 Command Timing 33
5.1.5 Address Handling 34
5.1.6 Crossing a Row Boundary 35
5.2 AHB INTERFACE 36
5.2.1 AHB Core 36
5.2.2 Data Buffer 37
5.2.3 x2 37
5.3 APB 39
5.4 ARBITER 40

6 SOURCE RESYNCHRONIZATION 41

6.1 PHASE SHIFT THE DATA STROBE 41
6.1.1 Delay Lock Loop 41
6.1.2 Inverter Delay 41
6.1.3 PCB Line Delay 42
6.1.4 Programmable Delay Line with Temperature sensing 42
6.2 SYNCHRONIES THE DATA 43
6.2.1 Simplified Phase Detector 44
6.2.2 Alternative method 46

7 FUNCTIONAL SIMULATION 49

7.1 PERFORMANCE 49
7.2 DDR SDRAM SIMULATION MODELS 50

8 BACKEND 51

8.1 SYNTHESIS 51
8.2 FLOORPLAN 51
8.3 PLACEMENT 53
8.4 TIMING PROBLEMS 54

5 DDR SDRAM Controller

The DDR SDRAM Controller (referred to as the memory controller) is designed to be able to support a various number of AHB slave interfaces. To achieve this, the control path and data path has been separated. The core memory controller handles the control path and the data path is incorporated into each AHB interface of the memory controller. An internal arbitrator decides which AHB interface that has access to the data path and the core memory controller. For the initialization of the memory controller there is also an APB interface. Through the APB interface EMRS and MRS registers are set up as well as the size and organization of the DDR SDRAM being used. The memory controller can handle a various number of DDR SDRAM chips and the maximum number of supported chips are decided upon synthesis.

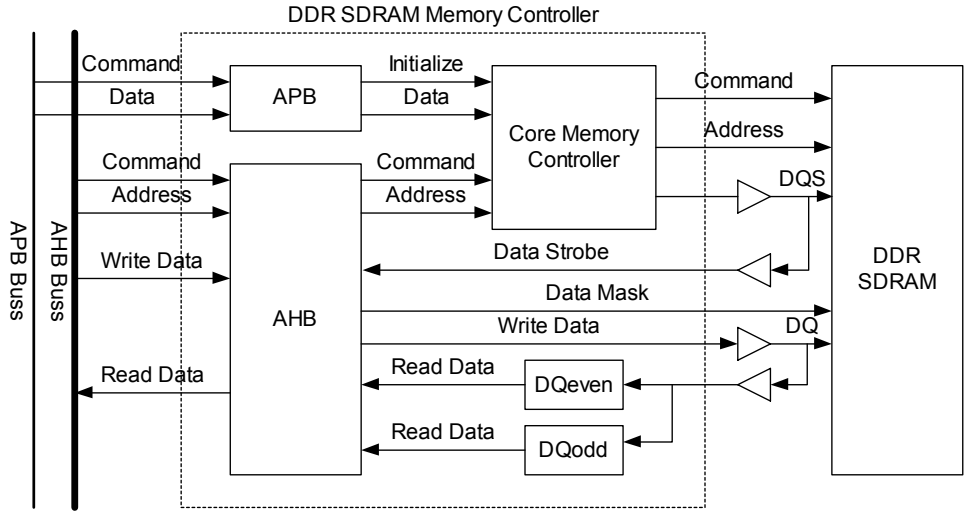

Figure 23 Schematic view of the memory controller with one AHB Interface

When an AHB interface has been granted access to the data bus and the core memory controller the AHB interface can tell the core memory controller to do either a read or write operation. The AHB interface also tells how long the burst is going to be. The core memory controller then handles the activation of rows and if necessary splits the burst into more than one command and issues them to the DDR SDRAM. The core memory controller then signals to the AHB interface when data has to be sampled from or presented on the data bus depending on if it is a read or write operation that have been requested. The core memory controller handles all the timings that are involved when a command can or has to be issued to the DDR SDRAM.

5.1 Core Memory Controller

The two main tasks for the core memory controller are to handle all the timings between different commands and to keep track of which rows that are currently activated. The activation of rows are time consuming and therefore the core memory controller has a look ahead functionality where the arbitrator can notify which command that is in turn to be executed after the current one has finished. This makes it possible to activate the row in advance if the next command is not accessing the same bank or chip as the current command.

The core memory controller divides the burst into as many commands that are needed to perform the whole burst. The core memory controller also handles the activation of the next row if the burst would cross the boundary between two rows.

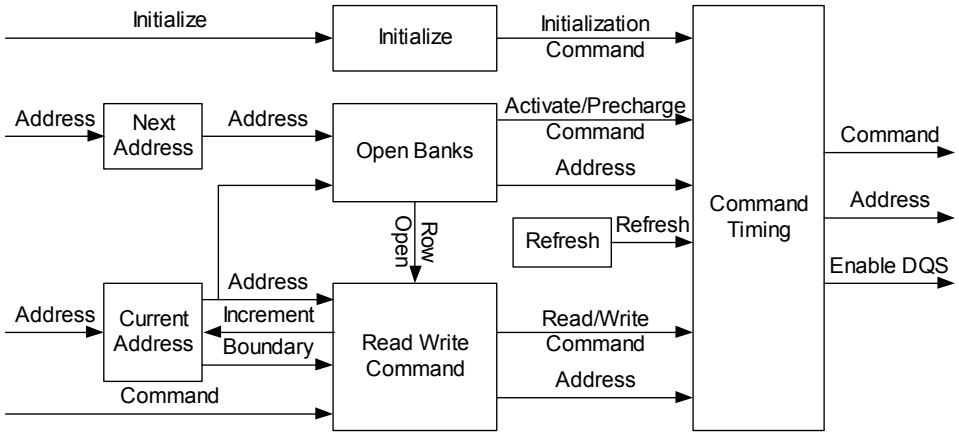

Figure 24 Schematic view of the Core memory Controller

5.1.1 Current and Next Address

The Address decoder divides the 32 bit address into a Column Address, Row Address, Bank Address and which Chip the burst is meant for. Since the width of the Column and Row address changes with the size and internal organization of the DDR SDRAM the Address decoder has to know what DDR SDRAM that is in use. This information is set up by the APB interface before the initialization of the DDR SDRAM is started.

The Address decoder for the current address also notifies if the next burst will end at the boundary to the next row. When the Address decoder is asked to increment the address it increments it in a predefined manner which makes it possible to do incremental burst even though the address of the DDR SDRAM is of wrapping nature. How this is achieved is described in Chapter 5.1.5.

5.1.2 Open Banks

A bank that has a row that is activated is called to be an open bank, therefore the name of this module. Since the activation of a command is time consuming it is vital to keep track of if a bank is open and in that case which row that are activated so that if a consecutive command is to the same bank and row it does not have to be activated. If the memory controller has more than one AHB interface and an arbitrator that tells which AHB interface that will be granted access to the core memory controller after the current one then the Open Banks module are

also able to look at that AHB interface Address, (Next Address), and in advance activate the row.

The Open Bank module is also responsible to make sure that the timing limits between PRECHARGE to ACTIVATE and ACTIVATE to ACTIVATE as well as ACTIVATE to READ or WRITE commands are fulfilled. To accomplish this, the Open Banks module actually consists of an Open Bank module for each DDR SDRAM chip. The Open Bank modules are within a top module that by looking at the addresses decides which chip that is currently being accessed and which will be accessed next. From the Open Bank modules that are handling the chips that are accessed their commands are propagated to the Command Timing module. Commands from the current Open Bank has of course higher priority than commands issued by the next Open Bank. The next address does not have to be an access to another chip than the current one for the row to be activated in advance. It is enough that the two addresses are not to the same bank in the same chip.

To keep track of all the timings that are involved each module have a precharge timer and activate timer for each bank which is set to zero whenever a PRECHARGE or ACTIVATE command is issued to that bank. A command can not be made to that bank unless the timers are above certain values. There is also a general timer for PRECHARGE commands to handle the minimum time necessary for precharges between different banks.

The open bank module that is handling the current command is also responsible for notifying when the row is activated and are ready for READ and WRITE commands.

5.1.3 Read Write Command

The Read Write Command module, (RW module) is responsible for taking the command from the AHB interface and divides the burst into as many necessary commands as needed to complete the burst. The RW module waits until it gets notified by the Open Blocks module that the row is activated before it issues the first command to the Command Timing module. Read more about how the burst is divided up into more than one command in Chapter 5.1.5.

Since a burst is to consecutive locations of the DDR SDRAM the RW module only checks that the row is active the first time it issues a command of a new burst or if the Address decoder notifies that a row boundary have been reached. If a row boundary has been reached it means that the bank will be precharged and a new row will be activated before the next READ or WRITE command can be issued.

5.1.4 Command Timing

The actually issuing of commands to the DDR SDRAM chips is done by the Command Timing module. The timing module keeps track of when a command can be issued to the DDR SDRAM so for example if the default burst length of the DDR SDRAM is eight beats then it is the timing module that controls that a new READ or WRITE command is issued every fourth clock cycle as long as the RW module issues new commands (for each clock cycle two beats of data is transferred one on positive edge and one on negative edge). The different commands are prioritized such that a REFRESH command is issued before a READ or WRITE command which in turn is issued before a PRECHARGE or ACTIVATE command. If there is not a pending command with a higher priority or the command with higher priority is not due to be issued a command with lower priority can be issued. For example if there is a pending READ command and the default burst length is eight beats and the previous command were a READ then there are two clock cycles before its time to issue the READ command so a pending PRECHARGE or ACTIVATE command can then be issued.

The timing module also indicates to the AHB interface when it is time to present data on the DDR SDRAM data bus on a WRITE burst. The timing module also indicates how many beats the AHB interface is to sample during a READ burst by raising the SampleData signal for one clock cycle for every beat to be read. The indication for when to present data is needed since the AHB interface does not have any other way to know when a WRITE command have been issued to the DDR SDRAM and therefore no way to know when its time to present the data on the data bus. During a READ the AHB interface knows when to sample the data by observing the activity on the data strobe signal and only have to know how many beats it should sample. The Command Timing module also enables the data strobe during writes.

5.1.5 Address Handling

The address of the DDR SDRAM wraps within the default burst length, as described in Chapter 3.4, while the burst issued by the AHB interface is sequential and incremental. The first observation to be done is that a wrapping address starting at location zero never wraps so if the address always would start at zero there would not be any problems with wrapping addresses. Always starting at location zero would introduce long latencies though since if the burst actually start at location six the first six locations would have to be masked out for the burst. This is not acceptable so in order to be able to start at a location not being zero and still get an incremental burst the fact that a burst can be interrupted by another burst has to be used. For the example where the burst start at location six a burst starting at location six can be issued lasting for two beats and then issue a new burst from location zero of the following address block. By cleverly addressing the memory with interrupting bursts it is possible to achieve incremental bursts.

Since it is not possible to make a burst that is shorter than two beats or beats that are of an odd number of beats to the DDR SDRAM the first thing is to make the address aligned to two beat bursts by checking the least significant bit. If the least significant bit is a one a burst of length one is issued to the command timing module that issues a burst to the DDR SDRAM with the least significant bit of the column address set to zero. The first beat of the burst is then masked out either by the AHB interface during READ, by ignoring to sample the first beat or by the command timing module by masking out the data with the data mask. Once the least significant bit is zero it is possible to issue bursts that are of even length. The second thing that is checked is the second least significant bit. If this bit is a one a burst of length two is issued without and changes of the column address. After that burst the third least significant bit is checked and if it is a one a burst of length four is issued. When all these steps have been taken the three least significant bits will all be zero and a burst length of eight can be issued without the address wrapping. If the default burst length is shorter than eight all steps are not necessary to take before burst of the default length can be issued without the address wrapping.

The Current Address module is aware of the steps and increments the address according to it as well as the RW module that issues the burst lengths to be made to the Command Timing module. The one thing the Command Timing module does is to look at the least significant bit and always sets it to zero and if it is a one it masks out the first beat during WRITE. For READ the masking is handled by the AHB interface. The algorithm for calculating the size of the next burst can be found in Appendix E.

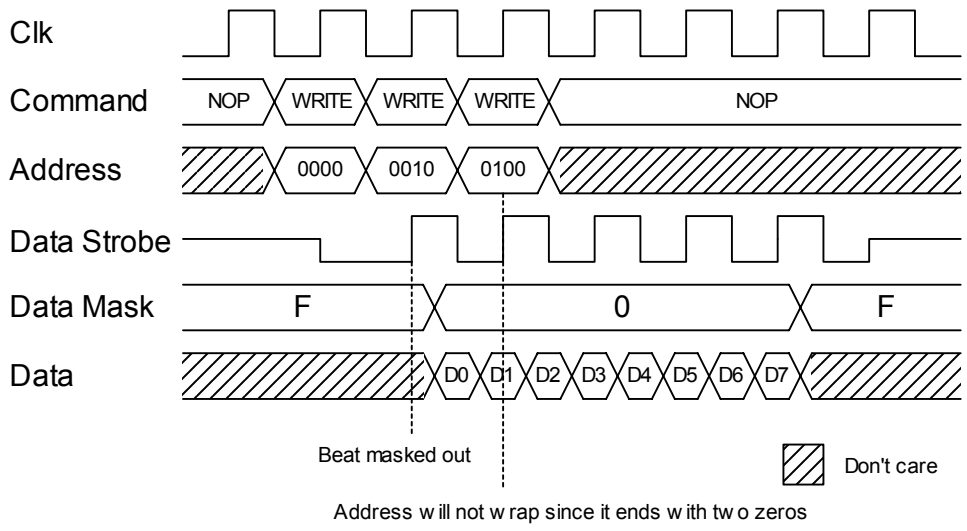

Figure 25 Eight beat burst starting at address 0001 with default burst length of four

5.1.6 Crossing a Row Boundary

When the current burst will end at the boundary between two rows, that is the column address will consist of only ones, the Current Address module indicates this by raising the Boundary signal. The RW module uses this information so that it knows that the row will be closed and the consecutive row will be opened. While waiting for the row to be opened the RW module places itself in closed state and waits for the Row Open signal to go HIGH.

When reaching a row boundary it can be necessary to make out unwanted data. Take for example if the row boundary would have been reached when address 0010 is issued in Figure 25, then the following WRITE command would not have been issued since a row boundary has been reached and the current row has to be closed. The consecutive row has to be activated before the next WRITE command can be issued thus making the address to wrap and two beats of unwanted data would have been presented by the AHB interface corrupting the whole burst. During a WRITE the masking of the two unwanted beats are done by lowering the PresentData signal generated by the timing module. For a READ the SampleData signal have only been raised for as many beats as it should sample so when it reaches the state that the address have wrapped it already has read all the data and it will ignore the following two beats.

Instead of masking out the unwanted data it would be possible to issue a precharge command to terminate the read burst making the need for keeping track of how many beats to read by the AHB interface obsolete. However all memory vendors does not allow a read burst to be terminated by a precharge even though the JEDEC specification states that a read burst can be terminated by a precharge.

5.2 AHB Interface

The AHB Interface is a slave interface of the AMBA AHB bus. The AHB interface is intended to be the access port to the DDR SDRAM(s) for other IP cores that exists on the same silicon chip as the Memory Controller. When a read request comes on the AHB bus the AHB Interface tells the Core Memory Controller the starting location of the data and how much data to retrieve. The data is buffered and as soon as the interface has started to receive data from the DDR SDRAM it presents the data on the AHB bus. For a write request the AHB Interface first buffers the data to be written before it tells the Core Memory Controller to write the data to the DDR SDRAM. The reason for first buffering the data is that the AHB bus works on a much lower clock frequency than the DDR SDRAM so to avoid running out of data during a write burst all data is buffered before the write burst is started.

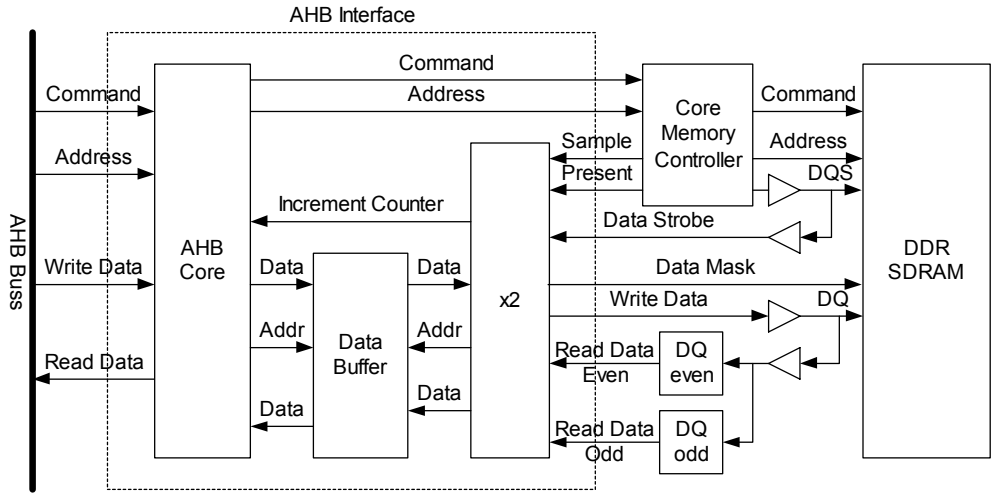

Figure 26 Schematic view of the AHB Interface

5.2.1 AHB Core

The AHB Core is the only module that has direct access to the AHB bus and is responsible for moving data between the AHB bus and the data buffer as well as handle read and write requests from the AHB bus. It also tells the Core Memory Controller the location and length of the burst. The AHB Core is capable of handling all types of bursts that are handled by the AHB bus like incrementing bursts where the data is written to consecutively higher addresses as well as bursts where the address wraps. The wrapping bursts are handled such that the data being stored in the Data Buffer is stored incremental. For example assume that there is a request for an eight beat write wrapping burst and the three least significant bits of the address is set to three. The first data is then stored to the Data Buffer starting at location three of the Data Buffer and the next four consecutive data is stored on address four to seven. The address is then wrapping so the rest of the data is stored on locations zero to six in the Data Buffer. This makes the wrapping burst into an incremental burst to the DDR SDRAM since the data now is stored incremental in the Data Buffer starting from location zero. Similar is done during a read wrapping burst where the lowest address of the wrapping read is calculated, by setting the right amount of least significant bits to zero. The data is then read from that location and stored in the Data Buffer. The data is then read from the Data Buffer in the wrapping order.

By handling wrapping writes in this manner the AHB interface is not capable of handling a loss of the bus during the write since then all data is not available for making an incremental write to the DDR SDRAM.

Read increment, that does not have a predefined length, is handled by filling the buffer with data from the DDR SDRAM and then present it on the AHB bus as long as the burst continues. If the burst is longer than the data stored in the Data Buffer then HReady is lowered, to suspend the burst, and the Data Buffer is filled once more with data and as soon as the buffer starts to fill up HReady is raised, which resumes the burst, and data is presented on the AHB bus. This continues until the burst is terminated by the AHB master. For a write increment with no predefined length the scenario is similar. The data buffer is filled until the AHB master terminates the burst or the buffer gets full upon HReady is lowered and the data stored in the buffer is written to the DDR SDRAM. Once the buffer is empty HReady is raised and the buffer starts to fill up again. This continues until the AHB master terminates the burst and the last data stored in the buffer have been written to the DDR SDRAM.

5.2.2 Data Buffer

The Data Buffer is a generated quad port on chip memory with an independent clock for each port that can store 16 words of 32 bits. The Data Buffer is used for buffering data and to move data between the AHB clock domain and the clock domain used for communicating with the DDR SDRAM(s). The depth of the Data Buffer has been chosen to the longest predefined burst of the AHB bus, which is 16 words.

5.2.3 x2

The x2 module is the most timing critical module of them all since it is responsible for presenting or sampling data in the rate handled by the DDR SDRAM. To achieve this, the x2 module run at the same clock frequency as the data is switched with on the data bus. During a write, data is read out from the Data Buffer and presented on the data bus as long as the Present Data signal from the Command Timing module is high. Along with the data presented on the data bus is also the corresponding Data Mask signal. The Data Mask signal is to indicate for the DDR SDRAM which bytes of the word to be written. In order to write a byte to the DDR SDRAM the data is presented on the data bus along with some other random data, since a byte is only 8 bits and the data bus is 32 bits wide. The data mask is used to tell which one of the bytes that is to be written. It is only possible to mask out data in groups of one byte and therefore restrains the data to be byte aligned. To simplify the handling of data for the memory controller the access has been restricted to also be halfword and word aligned depending on the requested size on the AHB bus. Since most modern CPU's also follows this restriction in practice it does not have any effect on efficiency or flexibility.

The AHB address has the byte as the smallest addressable block while the DDR SDRAM address has the word as the smallest addressable block. This is used for creating the Data Mask by looking at the two least significant bits of the AHB address this tells which byte within the first word to start reading from. If the AHB address would end with “01” it means that the first byte of the word has to be masked else the data at that location will be corrupted. Since all burst are converted into incremental bursts to consecutive addresses to the DDR SDRAM the rest of the data mask is generated by not masking out any of the data until the end of the burst. At the end of the burst the two least significant bits of the sum from the burst length and the starting address is used. This tells which byte is the last byte to be written to the DDR SDRAM and the rest will be masked out by the Mask Data signal. For reads it is much simpler since the data does not get corrupted when it is read from the DDR SDRAM so the first word containing the first byte to be read is read into the Data Buffer and then the

AHB core looks at the AHB address, using the two least significant bits, in much the same way as describe above, to present the data on the AHB bus.

When a write is made the data is aligned with the data strobe which is also generated by the memory controller so the timing is relatively easy to achieve. For a read on the other hand the data is aligned with the data strobe generated by the DDR SDRAM. The data strobe therefore needs to be delayed so that the data strobe becomes center aligned with the data. Different methods for this can be read in Chapter 6.1. The data strobe is then used to sample the data with the use of two Flip-Flops. One of the Flip-Flops sample data presented on the positive edge of the data strobe while the other Flip-Flop sample data presented on the negative edge. This makes the data available for twice the duration than it is on the original data bus creating more time to synchronize the data to the internal clock, Figure 27.

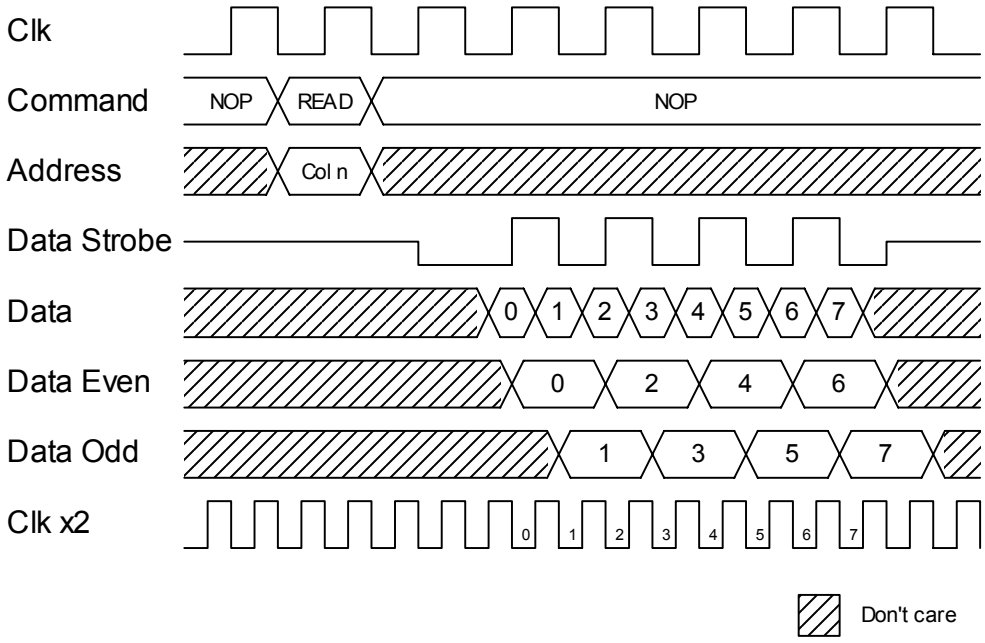

Figure 27 Wave chart of a read using odd and even data

The data strobe is not synchronized with the clock of the receiving IC and therefore a method for telling when the data on the Data Even and Data Odd bus is stable and can be sampled by the internal Clk x2 clock is needed. Such a method is being discussed in Chapter 6.2.

5.3 APB

The APB interface is used to initialize the memory controller so that it knows the size and organization of the DDR SDRAM that it is to work with. The initialization is done by writing data to two registers in the APB interface. The first register is for setting the size and organization of the memory as well as which refresh period that is to be used and is called the Variable Register. The second register is for setting the data to be used during Extended Mode Register Set and Mode Register Set and is called the MRS register. For the memory controller to work properly the Variable Register has to be set before any data is written to the MRS register. As soon as any data has been written to the MRS register the APB interface tells the Core Memory Controller to start the initialization of the DDR SDRAM chip(s).

Once the initialization of the DDR SDRAM chip(s) have started it is not possible to change the values of the two registers without resetting the Memory Controller first.

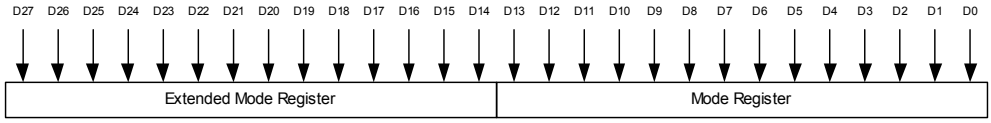

Figure 28 MRS Register see Chapter 3.2.5 and 3.2.6 for closer details

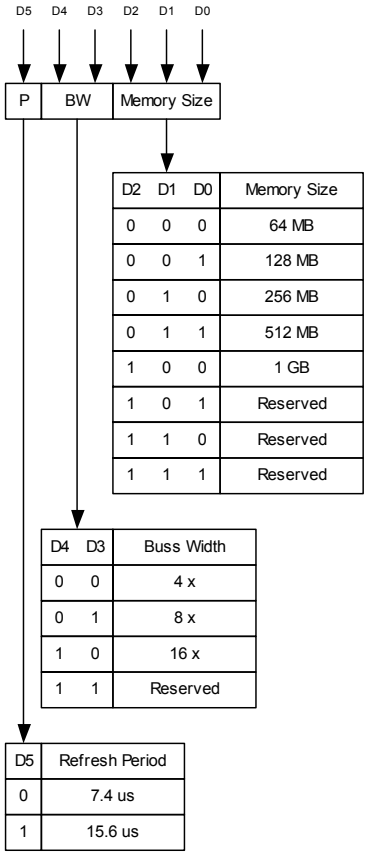

Figure 29 Variable Register

5.4 Arbiter

With more than one AHB Interface the utilization of the memory is increased making it possible to transfer more data to and from the DDR SDRAM(s) than if only one AHB Interface would be used. The arbiter can be implemented with any kind of arbitration protocol, as for example round robin or prioritized. The arbiter is also the module that limits how many AHB Interfaces the memory controller can service. The arbiter that has been implemented uses round robin and supports two AHB Interfaces. This makes the Arbiter simple since there is no need to implement any functionality to decide which AHB Interface that will be granted access when the current AHB Interface has finished its burst since there is only one more AHB Interface and it will be granted access if it has a pending burst.

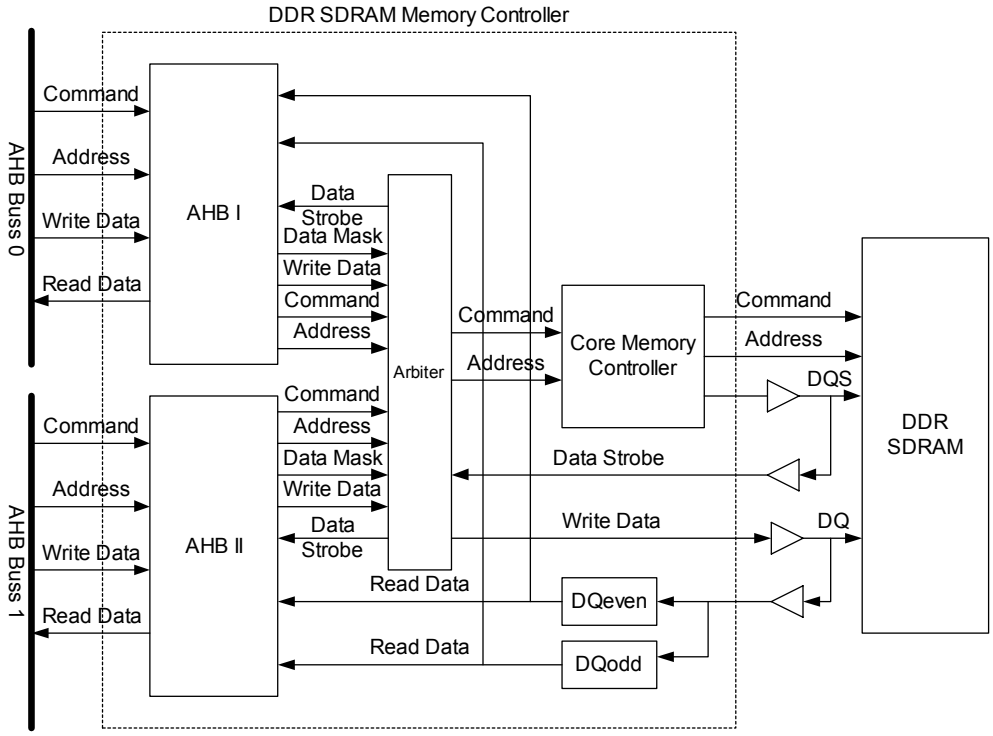

Figure 30 Schematic view of a DDR SDRAM Controller with two AHB interfaces

6 Source Resynchronization

The source-synchronous interface has several advantages compared to a standard synchronous interface as told in Chapter 2. One new problem though with the source-synchronous interface is that the data presented on the data bus is in synchronization with the data strobe and not with the internal clock of the receiving IC. This makes it necessary to resynchronize the incoming data to the internal clock. To make it even more difficult the data from a DDR SDRAM is edged aligned with the data strobe making it impossible to directly use the data strobe to sample the data without altering the data strobe first.

6.1 Phase Shift the Data Strobe

To be able to use the data strobe to sample received data the data strobe is usually phase shifted 90 degrees, in other words delayed a quarter of a clock period. This can be done in a number of different ways where the most common method is to use a Delay Lock Loop (DLL). Other methods that can be used are an inverter delay or PCB line delay [13]. A possible method based on a temperature sensor and a programmable delay line is also discussed.

6.1.1 Delay Lock Loop

The most common method to phase shift the data strobe is with the use of a DLL. The DLL is constructed of a phase detector and a digital delay line.

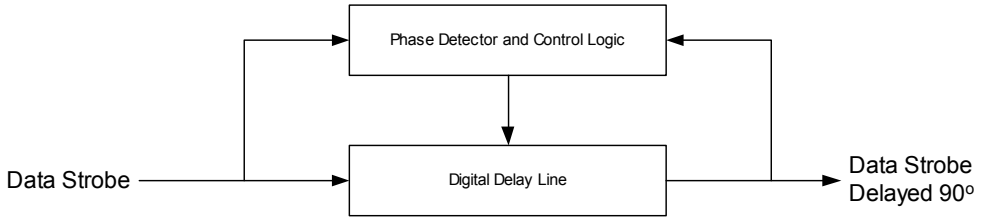

Figure 31 Schematic view of a Delay Lock Loop

The advantage with using a DLL is that regardless of variations of the data strobe the phase detector will always make sure that the delay is 90 degrees. The design difficulty when constructing a DLL is to create a Digital Delay Line that does not cause jitter problems and that have a fast initial locking. The time for the DDL to lock is the time it takes from the first transition of the data strobe until the Digital Delay Line is set up correctly so that the output signal is delayed with 90 degrees.

When constructing a DDR SDRAM controller with the use of a DLL the designer has to have in mind that initial read bursts is needed to give the DLL time to make its initial lock to the data strobe. During these read bursts the data should not be attempted to be sampled since the DLL has not had time to delay the data strobe which can result in meta-stability in the sampling Flip-Flops. The data strobe used during write bursts also has to be masked from the DLL.

6.1.2 Inverter Delay

An inverter delay chain is a number of inverters connected in series and for each inverter the signal passes, the signal is delayed by the time it takes to propagate through the inverter. To

create a 90 degrees phase shifted signal it is enough to pass the signal through the right amount of inverters. The problem with this method is that the delay through an inverter is dependent on the process voltage temperature so the total delay can vary significantly. To achieve a more stable phase shift, inverters that are voltage temperature independent should be used.

6.1.3 PCB Line Delay

The PCB Line Delay is a simple and robust way to phase shift the data strobe. To create the phase shift a delay line on the PCB is inserted in the data strobes path. The delay line is of a fixed value and does not track variations of the data strobe. The PCB has to be redesigned if a data strobe with another frequency would be used. This is most likely the case even though a PCB Line Delay would not have been used since the stubs of the data bus are constructed to reduce the reflections for a specific frequency. Making it necessary to redesign the PCB regardless of the PCB Line Delay being used or not.

6.1.4 Programmable Delay Line with Temperature Sensing

The major problem with the inverter delay chain is that the delay of the inverters can change with the process voltage temperature and process variations in the construction of the IC can also give variations in the inverter delay. This problem does not exist when using a DLL but on the other hand a DLL is large and expensive. A possible alternative could be to use a programmable delay line that is controlled by a temperature sensitive logic, Figure 32.

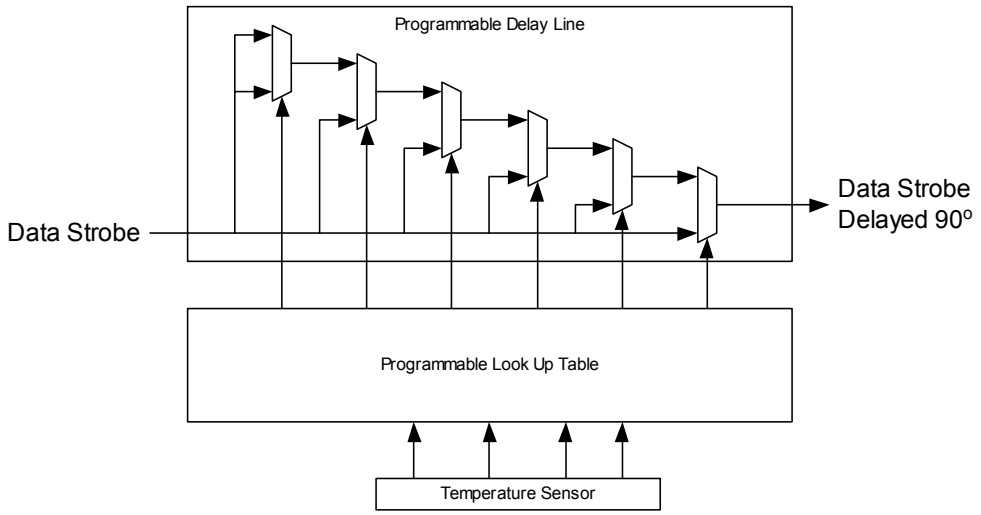

Figure 32 Programmable Delay Line with Temperature sensing

The advantage with this method is that most of it components are regular digital components that can be dealt with by the regular ASIC flow used for the creation of the entire chip. The only sophisticated component is the temperature sensor that is capable of giving a digital representation of the temperature. For the case in Figure 32 the temperature sensor has a representation of four bits giving 16 temperature levels which should be enough to create a stable delay of the data strobe. The temperature level is looked up in the Look Up Table, (LUT) and the stored information is used for setting up the muxes in the delay line. Depending on how much the data strobe has to be delayed the signal is propagated through a different number of muxes. The use of muxes is only one example on how the delay line could be constructed. The LUT is programmable and by running the IC core at different

temperature levels it is possible to test how much the data strobe has to be delayed to work properly and can be programmed into the LUT.

One problem with this method is to create reliable tests for the different temperature levels and it might not be enough to make the test with one IC and program the LUT for all other IC's according to that result. It might be necessary to make the temperature test for every IC.

6.2 Synchronies the Data

With the phase shifted data strobe it is possible to sample the data but the data still has to be synchronized with the internal clock of receiving IC. The data is first sampled by two Flip-Flops where one samples data on the rising edge and the other Flip-Flop samples data on the falling edge of the data strobe, Figure 33. This makes the data held by each Flip-Flop to be stable for twice the duration it is stable on the data bus making it easier to synchronize the data to the internal clock., Figure 34.

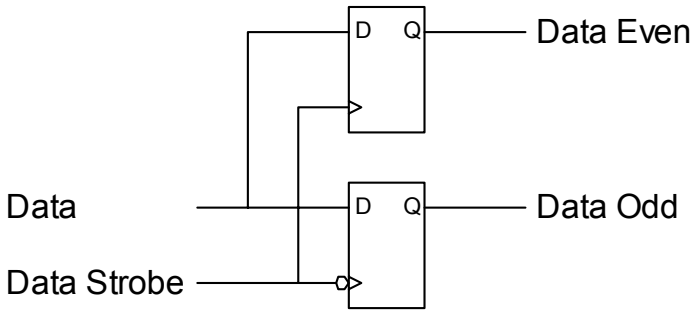

Figure 33 Schematic view of the sampling Flip-Flops

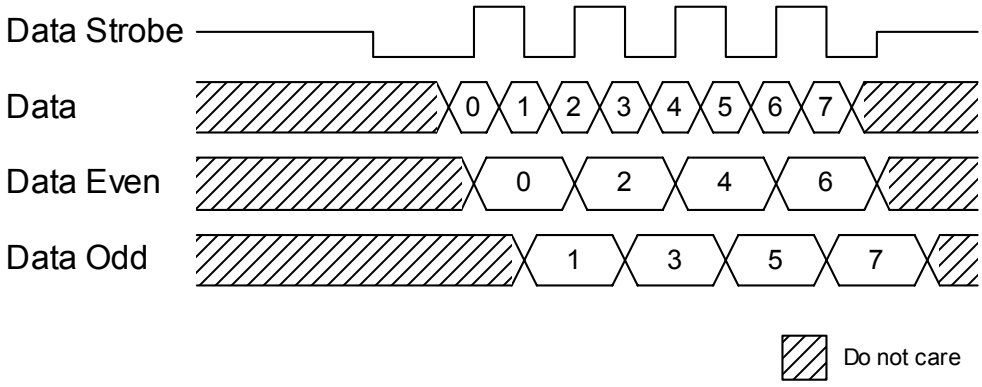

Figure 34 Wave chart of the sampling Flip-Flops

Using a clock frequency equal to the frequency which the data changes that is twice the frequency of the data strobe there will always be at least one rising edge of the clock where the data of Data Even and Data Odd will be stable regardless of the phase of the data strobe. To decide when the data is stable a reference clock is used with equal frequency as the data strobe. The relationship between the faster clock and the reference clock is that the rising edge of the faster clock is 90 degrees after the rising edge the slower clock. The phase of the data strobe is compared against the reference clock. If the rising edge of the data strobe is when the reference clock is low the data will be stable when the reference clock is high and if the rising edge of the data strobe is when the reference clock is low the data will be stable

when the reference clock is low, Figure 35. So if the relative phase between the reference clock and the data strobe is know it is also known when the data will be stable and can be sampled.

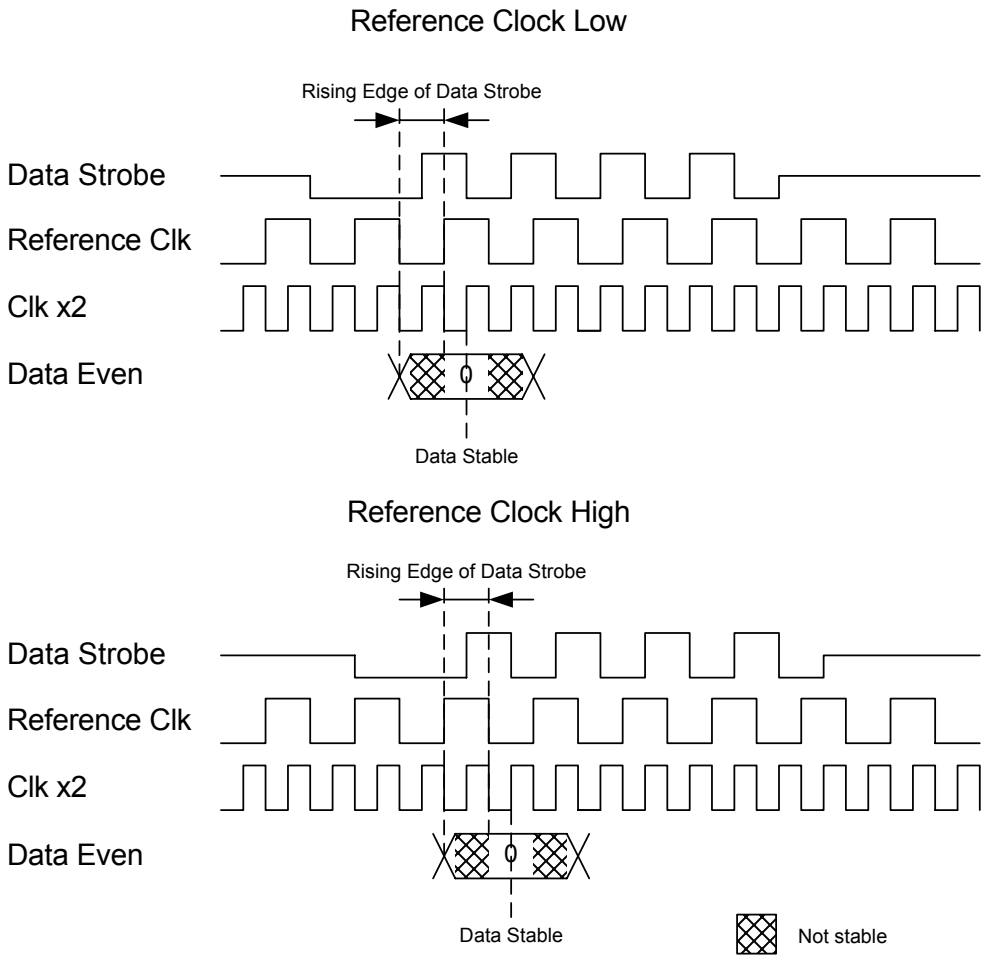

Figure 35 Wave chart illustrating when data is stable

6.2.1 Simplified Phase Detector

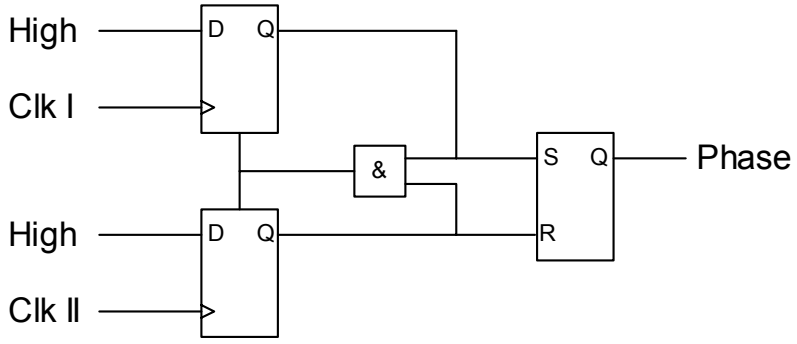

Figure 36 Phase detector

Figure 36 is a simplified phase detector and is constructed from a regular phase detector used for example in Phase Lock Loops (PLLs). The input port of the Flip-Flops are held at a constant high level and will drive the Out-port high as soon as a rising edge is detected at the clock input. When both Flip-Flops have had a rising edge on the clock input both Out-ports will be at a high level driving the AND-gate to high resetting the Flip-Flops driving the Out-ports to a low level. A regular phase detector uses the difference in high level on the Out-ports to decide the phase difference. The larger difference in the outputs from the Flip-Flops the large difference in phase. In this simplified version the Out-ports are connected to a SR Flip-Flop. A SR Flip-Flop works such that the Out-port is set to high when the Set-port is high and is reset to low when the Reset-port is high. When both the Set-port and Reset-port is low the Out-port is unchanged. The Out-port is undefined if both the Set and Reset-port is high. When Clk I has its rising edge before Clk II it will get periods of high levels on it Out-port while the Clk II Out-port will only have a sharp spike long enough to drive the and-gate high resetting the Flip-Flops. If the wires are long enough or decoupled by capacitors, inserted before the SR Flip-Flop, the spike will be seen as a low level thus not being registered by the SR Flip-Flop. This will set the SR Flip-Flop Out-port to high, Figure 37. In the other case when the Clk II rising edge appears before Clk I the Out-port of Clk II will have periods of high levels while the Clk I Out-port will get the spikes and this will reset the SR Flip-Flop to a low level.

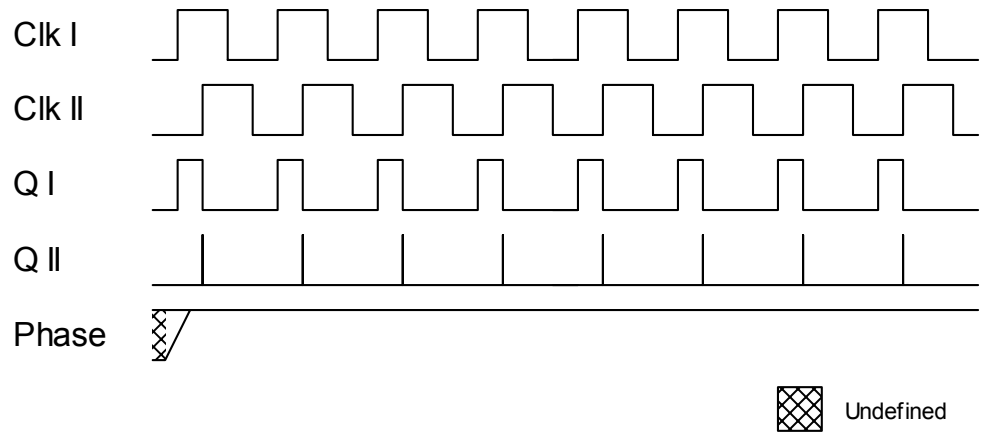

Figure 37 Wave chart describing the Phase Detector

If the rising edge of both clocks are simultaneous or almost simultaneous both Out-ports will have short spikes which both will be seen at the SR Flip-Flop as low levels not changing the Out-port of the SR Flip-Flop. This however is not a problem because if the data strobe is in phase with the reference clock the data will be stable for both sampling occurrences, Figure 38.

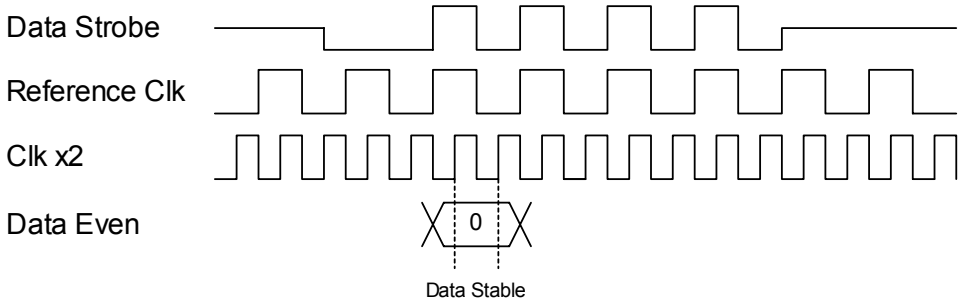

Figure 38 Wave chart of Data Strobe and Reference Clk in phase

Assuming that the data strobe is connected to Clk I on the phase detector the sampling of data will start after the first rising edge of the data strobe have been detected and at the first occurrence when the phase is of the opposite level of the reference clock, Figure 39. In the DDR SDRAM Controller the clock of the Core Memory Controller is used as the reference clock. The resynchronized data can easily be moved to another clock domain by the use of an asynchronous FIFO or a memory with independent clocks for the data ports.

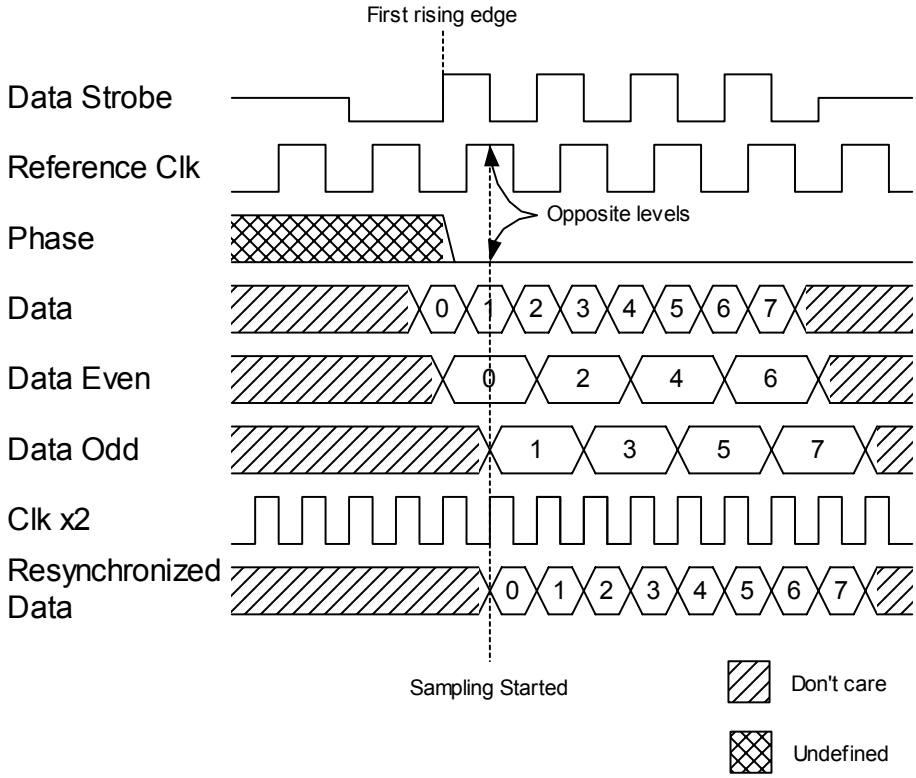

Figure 39 Wave chart of the resynchronization of data

6.2.2 Alternative method

It can seem strange to use the Clk x2 clock to resynchronize the data just to use an asynchronous FIFO or a memory to move the data to another clock domain. This is done because the data strobe is not oscillating continuously which makes it impossible to be used as a clock input to a FIFO or memory. However if the data strobe is phase shifted using a DLL

an internal version of the data strobe can be kept oscillating making it possible to use it directly to drive the clock input of a FIFO or memory. The data strobe is not switching fast enough since the data changes on both rising and falling edge making it necessary to use two FIFOs or memories, one storing the data for the rising edge and one for the falling edge. It would also be possible to use a PLL possibly together with a DLL to create a clock with twice the clock frequency and in phase with the data strobe making it possible to use only one FIFO or memory.

7 Functional Simulation

To verify the functionality of the DDR SDRAM Controller a test environment has been implemented. The test environment consists of one simulation test bench for the core memory controller where the test bench acts as an AHB Interface. The core memory controller is totally separated from the data bus so with this test bench no actual data is written or read from the simulation module of the DDR SDRAM that is used. The purpose of this test bench is to verify that the control signals generated by the core memory controller is correct and have been used mainly in an early stage of the implementation of the Memory Controller.

For a more complete verification of the Memory Controller's functionality a test bench that generates random traffic on the AHB interface has been implemented. The generated traffic is based on a 32 bit word being generated by a pseudo random generator. The generated word is used as the starting address of the burst and parts of the bits is also used as the representation of the control signals for the AHB interface. By looking at parts of the random word the test bench sets up a burst on the AHB interface and acts as a fully compliant AHB master. During a write the data that the test bench provides to be written is the same as the address. This makes the test bench simple since it does not have to keep track of what data have been written to the memory since during a read all it has to do is to compare if the data that the memory controller presents on the data bus is the same as the address. This however limits the size of the burst to be 32 bits else the data written to the memory and the address will not be equal. To check that the memory controller also works in these cases the test bench has been used to create a set of different burst of different sizes and then manually checking that the memory controller works correctly. Since this had to been done manually sizes of other than 32 bits have not been tested excessively. The reason for not verifying the functionality more thorough for other sizes than 32 bits is that it is enough to verify that the memory controller works for one size to show that the design is sound and functional. This implementation is meant to be a prototype and used as a reference for future implementations and is meant to highlight issues and give solutions and proposals on how to design and implement an effective DDR SDRAM controller.

7.1 Performance

The performance observed during simulation is that there is a latency of two clock periods between two consecutive write burst accessing already activated rows. The reason for this delay is that the arbiter for the two AHB interfaces is simple and is not capable of keeping track of which AHB interface have access to the Core Memory Controller and the Data bus when bursts from two different AHB interfaces start to overlap combined with the information the arbiter has to make its decisions. The latency between a write to read burst and read to read burst is two clock cycle plus the CAS latency of the read burst. To be able to reduce the impact of the CAS latency the Core Memory Controller would have to start the read burst earlier which could be done by looking at the command to follow and start to execute it in advance. The other two clock cycles of delay is because the simulation modules for the DDR SDRAM used for simulation did not support write bursts being interrupted by read burst. The JESD 79 specification states that this is supported so this might be a vendor specific latency.

An observation on how much the utilization of the bandwidth on the data bus can be increased by using more than one interface was also made. By adding a second AHB interface with the same load as the first AHB interface the time to complete all bursts only increased with 25 percent. That means that the data bus bandwidth utilization increased with as much as 60 percent by using two AHB interfaces. Even though this was just a random example being

observed it shows that there is a significant performance gain achieved by using more than one AHB interface. The same performance gain can not be expected to be achieved by adding a third AHB interface but it would possible make a difference. A Memory Controller with more than four AHB interfaces would probably not perform any better than a Memory Controller with fewer AHB interfaces since the AHB interfaces would spend most of their time waiting for the Core Memory Controller or data bus to be free. These observations were made with an AHB bandwidth of half the maximum bandwidth of the data bus to the DDR SDRAM and with full load of the AHB bus. For a system with equal bandwidth on the AHB bus as the data bus to the DDR SDRAM would not gain any by using two AHB interfaces since one interface is enough to sustain the Memory Controller with data.

7.2 DDR SDRAM Simulation Models

For the simulations a various number of DDR SDRAM simulation modules from two different memory vendors have been used to verify the functionality of the DDR SDRAM Controller. The first simulation model was a 64 Mb module from Samsung electronics for DDR 333. With this simulation module it was possible to verify that the initialization of the DDR SDRAM was performed correctly. This model reported errors on the timing of the data strobe and data mask during writes even though the timing given by the JESD 79 specification was followed. A few others simulation modules from Samsung and Micron were tested and all of them reported errors on something but after checking the JESD 79 specification none of them were accurate. To observe is that the same error was never reported by two different simulation modules. The final simulation module used was a 64 Mb module from Micron that was used to simulate DDR 400 timings.

8 Backend

The technology used for the backend is LSI's Gflx technology which is a cell based CMOS ASIC process technology for the 0.13-micron generation offering copper and low-K interconnects. To better meet the timing requirements the performance version has been used (Gflx-p). For the RTL netlist generation Cadence's Ambit BuildGates have been used and for floorplanning and placement Avant!'s ApolloII together with LSI's tools for place and route.

8.1 Synthesis

The minimum target for the implementation was to meet the timing requirements for DDR 200. Early in the development it stood clear that the design would be able to meet the timing requirements for higher specifications and a decision to try to meet the timing requirements for DDR 400 was taken. The higher target frequency led to that the design had to be improved and long combinatorial paths had to be split into shorter stages without trying to introduce any latency. The biggest challenge lied within the memory core where the most critical path have been when a new address is introduced that has to be compared to see if the current row is already open and if so issue a command to the Command timing module and increment the address for the next burst. To improve the timing the Read Write module only checks that the row is activated when the address is introduced and that it get notified by the Current Address module that the row boundary will be crossed in advance which tells the Read Write module when the row will get closed and that it need to back off. With an improved implementation it was possible to meet the timing requirements for DDR 400. The quad port on chip memories with independent clocks used as Data Buffers was created by a LSI memory compiler for the Gflx technology.

8.2 Floorplan

For the floorplan it has been assumed that the memory controller will be placed at the edge of a larger chip giving fast access to the pads of the chip and short busses to the DDR SDRAM chips. This assumption makes it convenient to gather all pins that are used for accessing the DDR SDRAM(s) on one side. Further it has been assumed that the pins of the Memory Controller are not restricted to a particular side of the core. The synthesized implementation of the Memory Controller contains two AHB interfaces and the corresponding pins for each interface have been placed on either side of the core. Pins for clocks and APB interface have been placed on the opposite side of the pins for access to the DDR SDRAM(s), Figure 40.

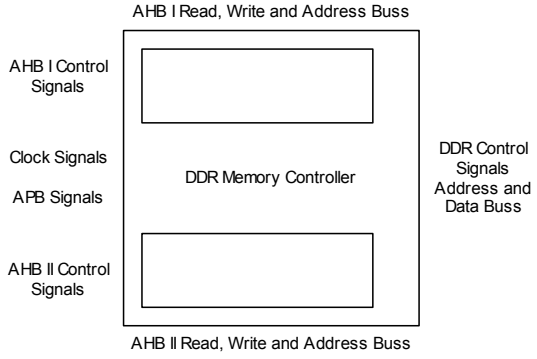

Figure 40 Pin placement

The memories take up a large part of the entire core area and needs to be placed manually since they are hard macros. The memories have one read port and one write port on each side. This made it ideal to place each memory along a side with the ports for the AHB read, write and address ports to face out from the chip and the read, write and address ports for the internal data path for the DDR SDRAM to face inwards. To guide the placement of the modules a region has been constructed around each memory wherein each AHB interface is contained. The size of the core and placement of the memories and regions can be seen in Figure 41. The figure does not show the two power rings with VDD and ground that exists around the core and together uses 12 μm of space on each side.

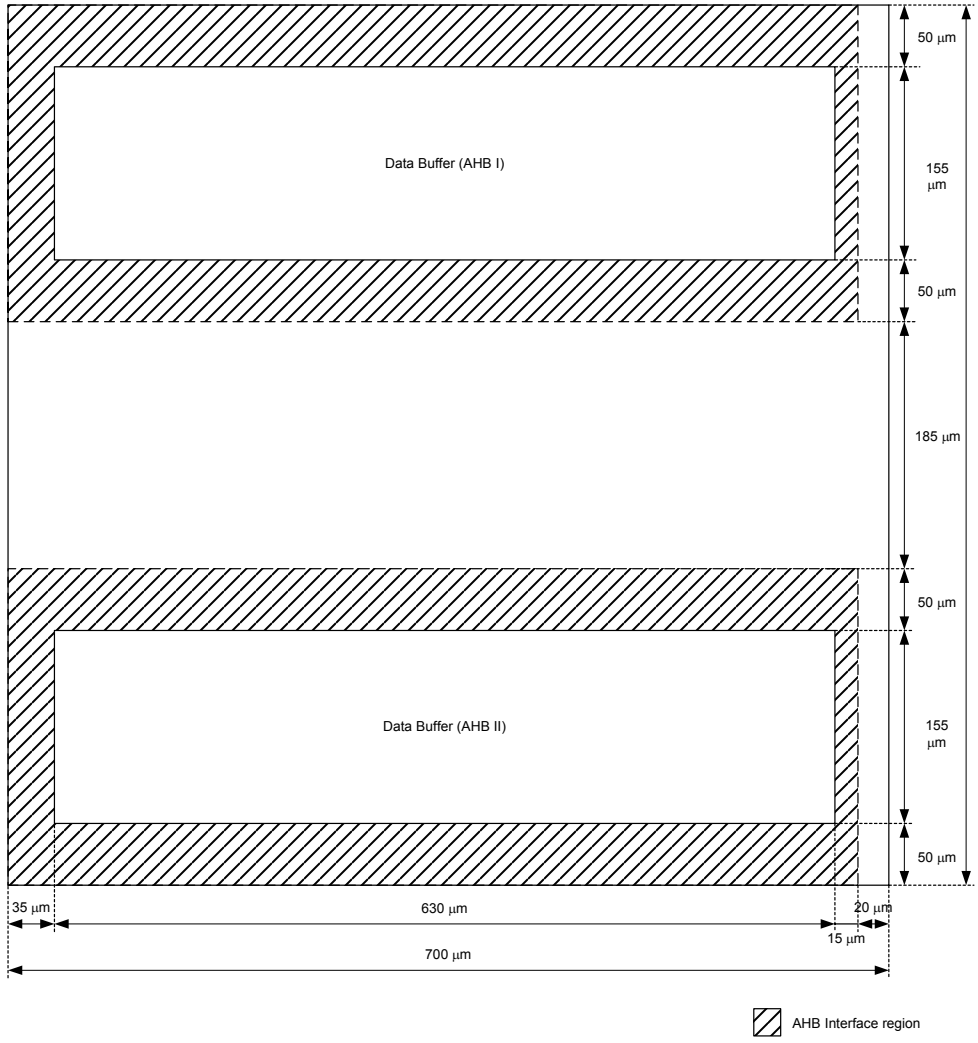

Figure 41 Floorplan

8.3 Placement

The place and route was done by LSI own tools for the Gflx technology and with the described floorplan there was no problems with either congestion or timing. The only problems during place and route was that Apollo core dumped when inserting clock trees making it impossible to create proper clock trees with little skew. Instead a tree used for reset signals had to be created, to avoid that Apollo core dumped. This problem is not related to the design of the DDR SDRAM Controller rather it is a problem within the Apollo program. Another problem with Apollo is that the hierarchical netlist generated after final place and route did not correspond to the spef-file containing timing information which is also generated by Apollo. The netlist and spef-file is used to create an sdf-file which is used for back annotated timing simulations. This problem could be solved by using one of LSI's tools together with the ECO information from the place and route tools to generate a functional netlist.

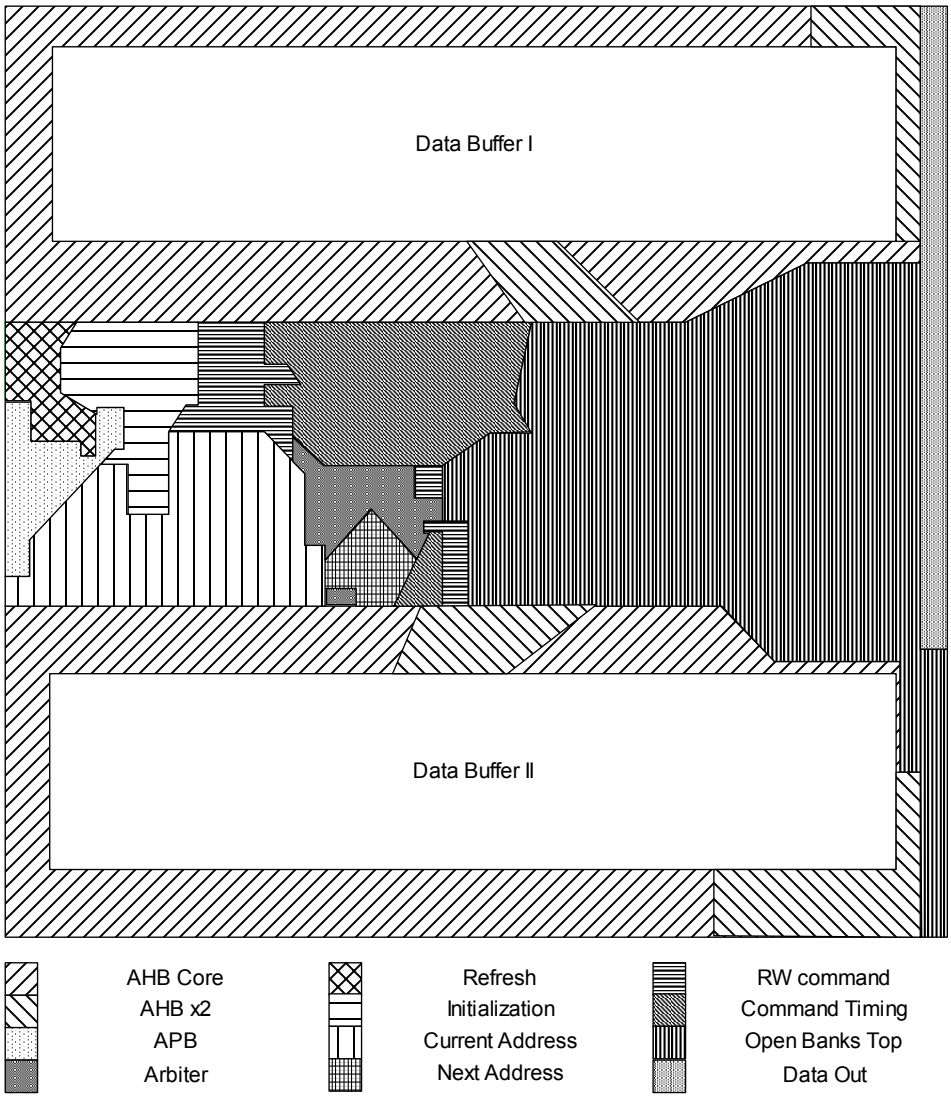

Figure 42 Placement

8.4 Timing Problems

Within the design there is a few timing problems where one of them is directly linked to the problem with the clock tree generation. The clock tree generated for the data strobe to be used to sample the data from the data bus has a skew of 700 picoseconds. The clock tree is connected to 64 Flip-Flops with half of the Flip-Flops being triggered on rising edge and the other half on falling edge. This made the data unstable for various bits of the data upon the sampling which resulted in that the timing checks for these Flip-Flops had to be turned off during simulation. With a working clock tree generation this would not be a problem.

A more serious problem is that the increment signal used by the AHB x2 module to indicate to the AHB Core how much data that have been retrieved can create meta-stability in the AHB Core's counter. The clock for AHB x2 is phase shifted compared to the clock for AHB Core and this together with when the sampling of data is started can lead to that the Increment signal only has 1.25 nanoseconds to travel from the AHB x2 module to the AHB Core module that updates the AHB Core counter, Figure 43. Along the path the increment counter passes through three logic gates and gets delayed 800 picoseconds and together with the transition time of the Flip-Flop which the Increment signal comes from the setup time for the Flip-Flops in the counter are violated. A solution to the problem would be to check if the clock of the AHB Core is low and in that case wait until the next rising edge of the Clk x2 giving the Increment signal 3.75 nanoseconds, Figure 44.

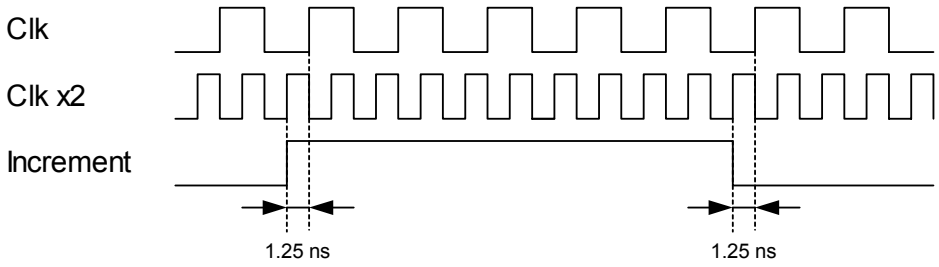

Figure 43 Increment signal creating setup problem

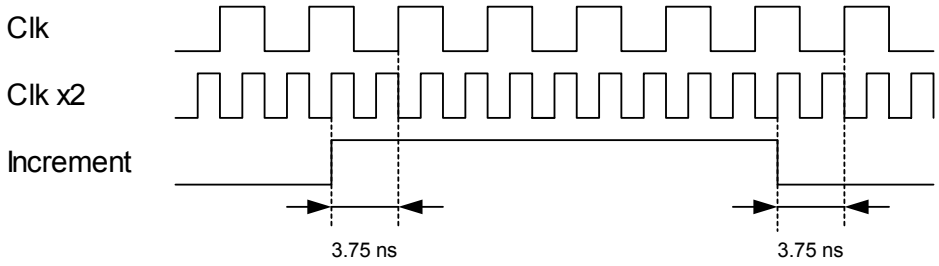

Figure 44 Increment signal not creating a setup problem

This has not been implemented because the increment counter is dependent on the relation between the two clocks which can vary dependent on which clock frequency is chosen on the AHB bus. The implemented increment counter works when the Clk x2 clock is a multiple of the clock frequency of the Clk clock. This is not always the case and therefore a change of the increment counter would be necessary making any changes done now irrelevant.

III

9 NEW DRAM INTERFACES 57

9.1 DDR II SDRAM 57
9.2 ESDRAM 57
9.3 QDR SDRAM 57
9.4 DIRECT RAMBUS 58
9.5 FCDRAM 58

10 FUTURE WORK 59

9 New DRAM Interfaces

The DDR SDRAM have become a standard in almost every desktop computer of today which has lead to large manufacturing quantities and a low and affordable price making the DDR SDRAM an interesting choice when choosing memory architecture to boost an architectures performance. The natural step to go to from using DDR would then be to upgrade the system to the coming DDR II standard that will increase the bandwidth to the memory even more than of today's DDR memories.

There has been a lot of talking about the Direct Rambus DRAM but considering the high licensing costs involved for manufacturing the Direct Rambus DRAM leading to high market costs for the chips and the improvement gains made with the DDR and in the future with the DDR II Direct Rambus DRAM will be used mostly for specific applications resulting in small manufacturing quantities leading to even higher prices.

9.1 DDR II SDRAM

The DDR II standard is an improved version of the DDR SDRAM with the same functionality except for that the data is transferred at even higher speeds [14]. To achieve the higher speeds a 4n-prefetch method is used working in the same way as the 2n-prefetch method described in Chapter 3.1.3 with the change that instead of for every clock cycle retrieve the data for two addresses the architecture retrieves data for four addresses. The data is also intended to switch four times for every clock cycle. The DDR II standard does not imply any dramatic changes to the functionality of a memory controller except that the smallest possible burst length is four and all burst lengths is a multiple of four instead of two as in regular DDR. The biggest design challenge is to implement the sampling of data being read from the DDR II since the data will switch even faster than for the DDR case.

9.2 ESDRAM

ESDRAM is an enhancement of the DRAM made by a caching structure consisting of a direct-mapped SRAM cache line, the same size as the DRAM row, associated with each bank. This makes it possible to make accesses to the most recently activated row regardless if the bank have been refreshed or precharged making it possible to make these service operations in the background while read and write operations are made to the cache line. This enhancement can be made to any SDRAM and comes down to a question of performance gain compared to cost if it is going to be manufactured in any larger scale.

9.3 QDR SDRAM

QDR SDRAM has two ports where accesses can be made simultaneously to different parts of the memory. QDR has two data busses making it possible to make both a read and a write burst at the same time to the same memory chip. The functionality of the busses and commands are the same as for the DDR. The success for the QDR is questionable except for specific high demanding applications since the added data bus add pins to the IC needed by the memory controller. Today pin count is a limiting factor when designing ICs so an increase in pin count is not preferred.

9.4 Direct Rambus

The Direct Rambus DRAM uses that same signaling as the a DDR but they have more banks than a regular DDR SDRAM making it more likely that a consecutive access is to another bank than the current access making it possible to activate the row in the next bank and thus increasing the possibility to pipeline accesses due to the reduced probability of consecutive accesses being mapped into the same bank. To reduce the manufacturing cost sense amplifiers are shared between two banks.

9.5 FCDRAM

Fast Cycle DRAM, developed by Fujitsu, is an enhancement to SDRAM which allows faster repetitive access to a single bank. This is done by dividing each bank into smaller blocks reducing the access time due to reduced capacitance of the word and bit line and enables pipelining of consecutive access to the same bank. Multistage pipelining of the core array hides precharge and allows it to occur simultaneously with input-signal latching and data transfer to the output latch. A disadvantage of this technique is the extra space needed on the Die for the dividing of the banks leading to high manufacturing costs. The FCDRAM can be found for both SDR and DDR SDRAM interfaces.

10 Future work

The continued work for the DDR SDRAM Controller would be to simulate it together with a real system and analyze the traffic to find out the characteristics of the AHB transactions to try to optimize the latency for read and write bursts. One part in the work of latency reduction is to optimize the arbiter to improve its knowledge of the state of the Memory Controller in order to be able to reduce the impact of the CAS latency in the beginning of a read burst. Another way to improve the efficiency of the AHB transactions would be to improve the buffer handling during writes. The implemented buffer handling work such that a write to the DDR SDRAM is not started until the AHB transaction is finished or the Data Buffer is full. This introduces latency since the stored data has to be written to the DDR SDRAM before the next AHB transaction can be started. To improve the efficiency there are two possible solutions. One is that when the first location of the Data Buffer has been written to the DDR SDRAM the next AHB transaction can be started if it is a write transaction. It can start writing data to the locations of the Data Buffer that already has been written to the DDR SDRAM. This does not improve the efficiency for read transaction since all data have to be written to the DDR SDRAM before the data bus will be available for reading data to the Data Buffer. To improve the efficiency of the read transactions it would be possible to start writing before the current transaction is finished or the Data Buffer is full. This could be done by splitting the burst that the AHB Interface module tells the Core Memory Controller to make. By doing this it would be possible to start writing data to the DDR SDRAM earlier and with that the time to finish the write would be reduced. The scheduling of these split up bursts would have to be studied so not to impact the overall performance of the Memory Controller. The overall utilization would be reduced if the scheduling would grant other AHB Interface modules access to the DDR SDRAM which can cause unnecessary precharge and activates in between the split bursts.

To allow for improved efficiency the DDR SDRAM has the ability to receive a maximum of eight refresh commands and to function without a refresh command for a period of up to eight times the standard refresh period. This gives the possibility to schedule the refresh commands such that it limits the impact on the efficiency of other commands. Such a scheduling scheme has not been implemented and the behavior of a complete system would be needed to be studied to find an effective scheduling routine.

For systems where the accesses to the memory are short bursts to consecutive locations a caching functionality should be implemented. The implemented Memory Controller reads the exact number of beats that is stated by the AHB transaction even if more data is presented on the data bus. The reason for this is that the AHB x2 module is not aware of if the data presented on the data bus is consecutive or if the data presented is because of that a row boundary have been reached, making the address to wrap, and needs to be discarded. The largest gain with a caching system would be when the bursts are short and to consecutive locations in conjunction with using a default burst length of eight. In a system with longer bursts the Data Buffer would have a higher utilization leaving little space to be used for caching making a caching system inefficient. To improve the efficiency of such a system the Data Buffers would need to be larger but that would increase the core size of the Memory Controller. A caching system for such a system has to be evaluated with respect of performance gain compared to the extra cost for the larger memories.

11 Conclusion

It has been shown that it would be possible to design and implement a DDR SDRAM Controller in house by Ericsson AB to be able to meet their increased demand for higher bandwidth to of chip memory for their future system platforms.

The crucial design issues lies within the phase shift of the data strobe and resynchronization of retrieved data during read bursts. This report gives several solutions to these issues and how to come to terms with them. The presented solutions are applicable when designing any Double Data Rate interface and not only DDR SDRAM interfaces.

The preferable solution would be the PCB line delay for shifting the data strobe since it is a simple and robust solution not demanding any advanced and costly circuitry as compared to the use of a DLL. For sampling the data it is necessary to know when to start the sampling, which is done by observing the preamble and the first rising edge of the data strobe. For this the simplified phase detector could be used to detect the phase of the data strobe to find out when the data will be stable and it is possible to sample without creating any meta-stability.

The study has lead to a working implementation of a DDR SDRAM Controller that can be used as a reference design for future implementations.

12 References

[1] JEDEC, "Double Data Rate (DDR) SDRAM Specification", JESD 79 Release 2, May 2002, JEDEC Solid State Technology Association

[2] ARM, "AMBA Specification" (Rev 2.0), 1999, ARM IHI 0011A

[3] Hansel A. Collins, Ronald E. Nikel, "DDR-SDRAM, High-speed, Source-synchronous Interfaces Create Design Challenges", September 2 1999, EDN www.ednmag.com

[4] Tegze P. Haraszti, "CMOS Memory Circuits", 2000, Kluwer Academic Publishers, ISBN 0-7923-7950-0

[5] Jan M. Rabaey, "Digital Integrated Circuits a Design Perspective", 1996, Prentice Hall Inc, ISBN 0-13-394271-6

[6] Elpida, "How to use DDR SDRAM", April 2002, Elpida Memory Inc., www.elpida.com/pdfs/E0234E30.pdf

[7] Lostcircuits, "Inside the EDDR Chip, Combining DRAM storage and SRAM speed", November 27 2000, Lostcircuits, www.lostcircuits.com/memory/eddr/

[8] Joerg Vollrath, "Tutorial: Characterizing SDRAMS", 1999, IEEE International Workshop on Memory Technology, Design and Testing

[9] Valerie Lines, Mammoun Abou-Seido, Cynthia Mar, Arun Achyuthan, Sampei Miyamoto, Yoshihiro Murashima, Shinzo Sakuma, "High Speed Circuit Techniques in a 150MHz 64M SDRAM", 1997, IEEE International Workshop on Memory Technology, Design and Testing

[10] Yasuhiro Konishi, Hisashi Iwamoto, Seiji Sawada, Yasumistu Muria, Takashi Araki, Masaki Kumanoya, "Dual Clock Scheme for over 200 MHz Synchronous DRAM System", September 17-19 1996, 22nd European Solid State Circuits Conference

[11] JEDEC, "Stub Series Terminated Logic for 2.5 V (SSTL_2)", JESD8-9B, May 2002, JEDEC Solid State Technology Association

[12] Reinhold Ludwig, Pavel Bretchko, "RF Circuit design", 2000, Prentice Hall ISBN 0-13-095323-7

[13] Samsung Electronics, "Key Points for Controller Design", 1998, DDR SDRAM/ SGRAM Application Note Samsung Electronics MPP-JLEE-Q4-98, www.samsungelectronics.com/semiconductors/dram/technical_data/application_notes

[14] Brian Davis, Bruce Jacob, Trevor Mudge, "The New DRAM Interfaces: SDRAM, RDRAM and Variants", 2000, Vol. 1940 of Lecture Notes in Computer Science pp.26-31

Appendix

A Figures

FIGURE 1 COMPARISON OF AN EIGHT BEAT BURST WITH SDR AND DDR 13
FIGURE 2 INTERNAL ARCHITECTURE OF A SDRAM WITH A SENSE AMPLIFIER AND MEMORY CELL....16
FIGURE 3 MEMORY CELL IN CROSS SECTION 16
FIGURE 4 MEMORY CELL IN TOP VIEW....17
FIGURE 5 SINGLE DATA RATE SDRAM CLOCK PERIOD LIMIT....17
FIGURE 6 SCHEMATIC VIEW OF THE CLOCK DELAY IN A DDR SDRAM 18
FIGURE 7 COMPARISONS OF A SDR AND A DDR WITH DLL....18
FIGURE 8 SCHEMATIC VIEW OF DDR SDRAM ARCHITECTURE 19
FIGURE 9 WAVE CHART ILLUSTRATING 2N-PREFETCH....19
FIGURE 10 WAVE CHART OF A READ BURST WITH EIGHT BEATS....20
FIGURE 11 WAVE CHART OF A WRITE BURST WITH EIGHT BEATS 21
FIGURE 12 WAVE CHART OF A REFRESH....21
FIGURE 13 MODE REGISTER DEFINITION....22
FIGURE 14 EXTENDED MODE REGISTER DEFINITION 23
FIGURE 15 WAVE CHART OF THE INITIALIZATION OF THE DDR SDRAM 24
FIGURE 16 MATCHED SIGNAL PATH 25
FIGURE 17 EXAMPLE OF A SSTL_2, CLASS II, OUTPUT ENVIRONMENT....25
FIGURE 18 VOLTAGE SWING 26
FIGURE 19 ILLUSTRATION OF THE EFFECT OF DIFFERENT SLEW RATE FOR POSITIVE AND NEGATIVE FLANK 26
FIGURE 20 EXAMPLE OF A SSTL_2, CLASS II, DIFFERENTIAL SIGNALS....27
FIGURE 21 DIFFERENTIAL SIGNAL WAVE CHART 27
FIGURE 22 PRE AND POST AMBLE....28
FIGURE 23 SCHEMATIC VIEW OF THE MEMORY CONTROLLER WITH ONE AHB INTERFACE 31
FIGURE 24 SCHEMATIC VIEW OF THE CORE MEMORY CONTROLLER....32
FIGURE 25 EIGHT BEAT BURST STARTING AT ADDRESS 0001 WITH DEFAULT BURST LENGTH OF FOUR....35
FIGURE 26 SCHEMATIC VIEW OF THE AHB INTERFACE....36
FIGURE 27 WAVE CHART OF A READ USING ODD AND EVEN DATA 38
FIGURE 28 MRS REGISTER SEE CHAPTER 3.2.5 AND 3.2.6 FOR CLOSER DETAILS 39
FIGURE 29 VARIABLE REGISTER 39
FIGURE 30 SCHEMATIC VIEW OF A DDR SDRAM CONTROLLER WITH TWO AHB INTERFACES 40
FIGURE 31 SCHEMATIC VIEW OF A DELAY LOCK LOOP....41
FIGURE 32 PROGRAMMABLE DELAY LINE WITH TEMPERATURE SENSING....42
FIGURE 33 SCHEMATIC VIEW OF THE SAMPLING FLIP-FLOPS 43
FIGURE 34 WAVE CHART OF THE SAMPLING FLIP-FLOPS 43
FIGURE 35 WAVE CHART ILLUSTRATING WHEN DATA IS STABLE....44
FIGURE 36 PHASE DETECTOR 44
FIGURE 37 WAVE CHART DESCRIBING THE PHASE DETECTOR....45
FIGURE 38 WAVE CHART OF DATA STROBE AND REFERENCE CLK IN PHASE 46
FIGURE 39 WAVE CHART OF THE RESYNCHRONIZATION OF DATA 46
FIGURE 40 PIN PLACEMENT 51
FIGURE 41 FLOORPLAN 52
FIGURE 42 PLACEMENT....53
FIGURE 43 INCREMENT SIGNAL CREATING SETUP PROBLEM....54
FIGURE 44 INCREMENT SIGNAL NOT CREATING A SETUP PROBLEM 54

B Terminologies

AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architechture
APB	Advanced Peripheral Bus
ASIC	Application Specific Integrated Circuit
Beat	The transaction of one set of data
Burst	A set of consecutive beats
Core	Core is usually used as the name of a larger functional block like a CPU core, DSP or memory controller core in an IC. Core is also used in this document as a name of modules with vital functionality.
CMOS	Complementary Metal-Oxide Semiconductor
DDR	Double Data Rate
Default Burst Length	The burst length set by the Mode Register Set command
Die	The silicon upon the IC is made
DLL	Delay Look Loop
ECO	Engineering Change Order
FIFO	First In First Out
Flip-Flop	Flank triggered digital curcuit that on either positive or negative flank of its clock signal samples the input port and presents its value on the output port untill next flank on the clock
Gflx	LSI's name of there 0.13 μm CMOS technology
IC	Integrated Circuit
IP	Intellectual Property
Jitter	Variations in the oscillation of the clock
LSI	ASIC vendor
Mb	Mega bit
Mbps	Mega bit per second

Module	Hirarchy level, logic with close relation is gathered in one module and a module can consist of several modules. The Command Timing is an example of a module and the Core Memory Controller is an example of a module consisting of other modules
Netlist	The representation of the implemented logic as a file describing the interconnects between standard cells.
PCB	Printed Curcuit Board
RTL	Register Transfer Level
SDR	Single Data Rate
SDRAM	Synchronous Dynamic Random Access Memory
Slew Rate	The maximum possible change of a magnitude of units per unit of time (Voltage/second or Amper/second)
Time of flight	The time it takes for a signal to travel between two Integrated Curicuits
VDD	Voltage source
VHDL	VHSIC Hardvare Description Language
VHSIC	Very High Speed Integrated Circuit
VIA	Connection from one metal layer to another

C DDR SDRAM Commands

NOP	No Operation
ACTIVATE	Activates a row in a bank. The bank has to be in idle state
PRECHARGE	Close the currently active row in a single bank or in all banks
READ	Make a read burst
READ with Autoprecharge	Make a read burst and automatically precharged the bank after the burst has finished
WRITE	Make a write burst
WRITE with Autoprecharge	Make a write burst and automatically precharged the bank after the burst has finished
REFRESH	Refresh the memory. Needs to be done periodically
MRS	Mode Register Set
EMRS	Extended Mode Register Set

D Place and Route Statistics

```
Libraries Information

Library Name                    Technology
gflxp.techlib                   gflxp
rr_ddr_rwbuffer.tech            gflxd

Section One: Design Summary
===========================

     Top Module                :ddr_top
     Technology                :gflxp
     Design Quoted Dimension   :(X = 0.76, Y = 0.76)

Section Two: Design Statistics Summary
======================================

(2.1). I/O Statistics

     Input Pins Used     : 227
     Output Pins Used    : 165
     Bidirect Pins Used  : 0

     Total Pins Used     : 392
     Total Pads Used     : 0
     Total Slots Used    : 0

(2.2). Design Statistics

     Logic Units Used(lu)           :   98648.00
     Logic Gates Used(lg)           :   28998.00
     Mix-n-match Logic Gates(hmg)   :       0.00
     ClockOverhead Logic Gates(cog):     225.50
     Megacell Units Used(mu)        :  121401.20
     Cell Units Area                :     2.0172 um^2
     Chip Raw Units                 :   82004.00
     Chip Usage                     :    2.68340
     Total Cells                    :    9737.00
     Total Cell Types               :     200.00
     Total Units (lu + mu)          :  220049.20

     Power Leakage

         Power Type            Leakage Power
         vdd2                  177.47107 uW

         Total Chip Leakage Power : 177.47107 uW

     Total Signal Nets         : 10218
     Total NC Nets             : 2
     Average Pins/Nets         : 3.77720
```

E VHDL Code

```
-- Address                     Address location of next burst
-- Size unsigned(3 dowonto 0) The size of the next burst
-- DefaultBurstLength         The burst length set with MRS

if Address(0) = '1' then
      Size <= x"1";
elsif Address(1) = '1' then
      Size <= x"2";
elsif Address(2) = '1' and DefaultBurstLength >= 4 then
      Size <= x"4";
else
      Size <= DefaultBurstLength;
end if;
```