Faster SDRAM Controller with Inbuilt Memory Using AHB Interface

S. Gopala Krishna[#] M. Nagarani[$]
[#$]Electronics and Communication Department
[#$]Vivekananda Institute of Technology and Science, Karimnagar, AP, INDIA

Abstract- ***Microprocessor performance has improved rapidly these years. In contrast, memory latencies have improved little. The result is that the memory access time and its associated speed has been a bottleneck which limits the system performance. A high performance memory controller (MC) which is SDRAM controller is designed to attack this problem. The memory controller is the part of the system that controls the memory. The memory controller is normally integrated into the system chipset. A high performance bus slave interface is used which is compatible with that of the low speed SDRAM memory. The SDRAM controller designed controls all the operations associated with the SDRAM from and to the memory. Using the FIFOing and pingpong technique the performance of the SDRAM controller is increased. In this paper decreased delay produced when comparing the SDRAM controller without FIFOing and with FIFOing. Hence latency is been reduced by 92% with the proposed SDRAM controller.***

Keywords- SDRAM, AHB Interface, FIFOing, pingpong

I. Introduction

The memory controller is designed which compatible with Advanced High-performance Bus (AHB) which is a new generation of AMBA bus. The AHB is for high-performance, high clock frequency system modules. The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripherals.

The design two way cable networked SoC, that is SDRAM controller connected by AMBA (Advanced Microcontroller Bus Architecture).The AMBA AHB is for high-performance, high clock frequency system modules. The AHB acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripherals.

It has their own bandwidth requirements and responding speed requirements for SDRAM. By analyzing the multiple accesses from the modules and the SDRAM specifications such as its accessing delay, we take both side 1 and side 2 into consideration respectively. On side 1, we use bank closing control. On side 2, the controller employs two data write FIFO to reduce the data access awaiting time, and uses 2 read FIFO to decrease the CAS delay time when reading data from SDRAM.

Due to the complexity of implementing the interleaving technique, we haven't introduced that technique to our design yet. However our design is proved to be functionally correct and high-performance. A SDRAM must be initialized before starting to access it. The configurable timing analysis scheme considers that the timing process might have tiny differences between different SDRAMs from different corporations

II. Related Work

On-chip bus organized is among the top challenges in SoC technology due to rapidly increasing operation frequencies and growing chip size. Usually, IP cores, as constituents of SoCs, are designed with many different interfaces and communication protocols. Integrating such cores in a SoC often requires insertion of suboptimal glue logic. Standards of on-chip bus structures were developed to avoid this problem. Currently there are a few publicly available bus architectures from leading manufacturers, such as Core Connect from IBM, AMBA from ARM, Silicon Backplane from Sonics, and others. This paper focuses on SoC providing a survey of three popular buses called AMBA, Core Connect and Wishbone from an industrial and research viewpoint.

2.1 Advanced High-performance Bus (AHB)

AHB is a new generation of AMBA bus which is intended to address the requirements of high-performance synthesizable designs. It is a high-performance system bus that supports multiple bus masters and provides high-bandwidth operation. AHB is for high clock frequency system modules. It acts as the high-performance system backbone bus. AHB supports the efficient connection of processors, on-chip memories and off-chip external memory interfaces with low-power peripherals.

AMBA AHB implements the features required for high-performance, high clock frequency systems including: which are burst transfers, Pipelined operation, multiple bus masters, split transactions, wider data bus configurations (64/128 bits)

Bridging between this higher level of bus and the current ASB/APB can be done efficiently to ensure that any existing designs can be easily integrated. An AMBA AHB design may contain one or more bus masters, typically a system would contain at least the processor and test interface. However, it would also be common for a Direct Memory Access (DMA) or Digital Signal Processor (DSP) to be included as bus masters.

The external memory interface, APB Bridge and any internal memory are the most common AHB slaves. Any other peripheral in the system could also be included as an AHB slave. However, low-bandwidth peripherals typically reside on the APB. A typical AMBA AHB system design contains the following components. A bus master which is able to initiate read and write operations by providing an address and control information. Only one bus master is allowed to actively use the bus at any one time.

Even though the arbitration protocol is fixed, any arbitration algorithm, such as highest priority or fair access can be implemented depending on the application requirements and the AHB decoder is used to decode the address of each transfer and provide a select signal for the slave that is involved in the transfer.

AHB signal prefixes: H indicates an AHB signal. For example, HREADY is the signal used to indicate that the data portion of an AHB transfer can complete. It is active HIGH.

2.2 Advanced System Bus (ASB)

An AMBA-ASB based microcontroller typically consists of a high-performance system backbone bus, able to sustain the external memory bandwidth, on which the CPU and other Direct Memory Access (DMA) devices reside, plus a bridge to a narrower APB bus on which the lower bandwidth peripheral devices are located. ASB signal prefixes B is an ASB signal for example, BnRES is the ASB reset signal. It is active LOW.

The ASB is a high-performance pipelined bus, which supports multiple bus masters. The basic flow of the bus operation is that the arbiter determines which master is granted access to the bus. When granted, a master initiates transfers on the bus.

The decoder uses the high order address lines to select a bus slave. The slave provides a transfer response back to the bus master and data is transferred between the master and slave.

2.3 **Advanced Peripheral Bus (APB)**

The AMBA ArPB should be used to interface to any peripherals which are low bandwidth and do not require the high performance of a pipelined bus interface. The latest revision of the APB ensures that all signal transitions are only related to the rising edge of the clock. The changes to the APB also make it simpler to interface it to the new Advanced High-performance Bus (AHB).

III. AHB COMPLIANT SDRAM CONTROLLER

We used FIFO to store the Read/Write commands coming from processors/user side along with corresponding write data and included a search engine to search recently read/write data inside the FIFO in order reduce the clock cycles of fetching data from SDRAM.

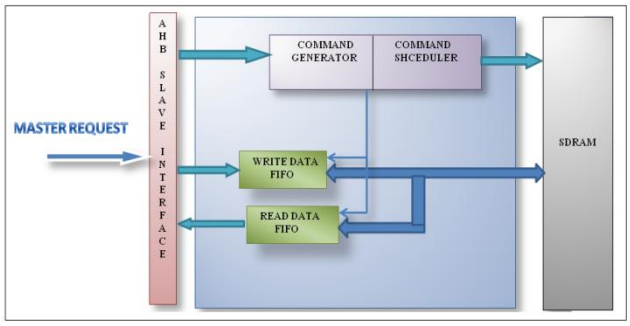

Figure:.3.1 System Module

Synchronous dynamic random access memory (SDRAM) is dynamic random access memory (DRAM) that is synchronized with the system bus. Classic DRAM has an asynchronous interface, which means that it responds as quickly as possible to changes in control inputs. SDRAM has a synchronous interface, meaning that it waits for a clock signal before responding to control inputs and is therefore synchronized with the computer's system bus enabling higher speed. The SDRAM controller is capable of either 16-bit or 32-bit data path, and supports byte, half-word and word access. Bursts can be used for both write and read access.

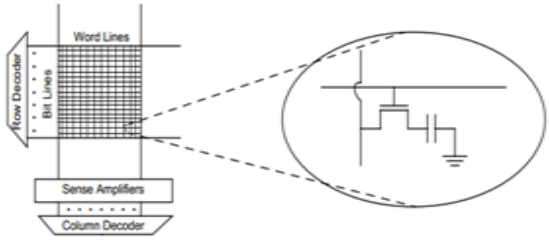

Figure:.3.2 SDRAM Array structure

In DRAM devices, large numbers of DRAM cells are grouped together to form DRAM array structures. Above figure illustrates a single bank of DRAM storage cells where a row address is sent to the row decoder, and the row decoder selects one row of cells.

A row of cells is formed from one or more word lines that are driven concurrently to activate one cell on each one of thousands of bit lines. There may be hundreds of cells connected to the same bit line, but only one cell will place its stored charge from its storage capacitor on the bit line at any one time. The resulting voltage on the bit line is then resolved into a digital value by a sense amplifier.

3.1 **Functional Description of SDRAM**

The SDRAM achieve high-speed operation. A single read or write access for the SDRAM effectively consists of a single n-bit-wide, one-clock cycle data transfer at the internal DRAM core and one corresponding n-bit-wide, one- clock-cycle data transfers at the I/O pins.

A bidirectional data strobe (DQS) is transmitted externally, along with data, for use in data capture at the receiver. DQS is a strobe transmitted by the SDRAM during READs and by the memory controller during WRITEs.

DQS is edge-aligned with data for READs and center-aligned with data for WRITEs. The SDRAM operates from a different clock (CK and CKE); the crossing of CK going HIGH and CKE going LOW will be referred to as the positive edge of CK. Commands (address and control signals) are registered at every positive edge of CK. Input data is registered on both edges of DQS, and output data is referenced to both edges of DQS, as well as to both edges of CK. Read and write accesses to the SDRAM are burst oriented; accesses start at a selected location and continue for a programmed number of locations in a programmed sequence. Accesses begin with the ACTIVE command, which may then be followed by a READ or WRITE command. The address bits issued coincident with the ACTIVE command are used to select row to be accessed. The address bits registered coincident with the READ or WRITE command are used to start the column location for the burst access. The SDRAM provides for programmable READ or WRITE burst lengths of 2, 4, or 8 locations. As with standard SDRAMs, the pipelined architecture of SDRAMs allows for concurrent operation, thereby providing high effective bandwidth by hiding row recharge and activation time.

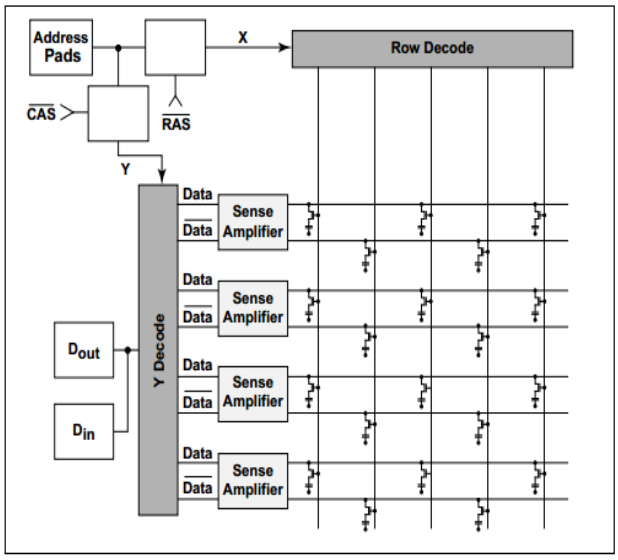

Figure: 3.1 Simplified SDRAM Diagram

Row addresses are present on address pads and are internally validated by the RAS (Row Address Access) clock. A bar on top of the signal name means this signal is active when it is at a low level. The X addresses select one row through the row decode, while all the other non-selected rows remain at 0V.

Column addresses are present on the address pads and are internally validated by the Column Address Access (CAS) clock. Each selected memory cell has its data validated in a sense amplifier. In modern DRAM devices, the capacitance of a storage capacitor is far smaller than the capacitance of the bit line. Typically, the capacitance of a storage capacitor is one-tenth of the capacitance of the long bit line that is connected to hundreds of other cells [2].

The relative capacitance values create the scenario that when the small charge contained in a cell is placed on the bit line, the resulting voltage on the bit line is small and difficult to measure in an absolute sense. In DRAM devices, the voltage sensing problem is resolved through the use of a differential sense amplifier that compares the voltage of the bit line to a reference voltage.

Operation of SDRAM controller

Only one open row in an active bank can be accessed. An ACTIVE command can open a row and make active the particular row in the memory. A PRECHARGE command issued to memory can set the SDRAM to idle state, i.e. closing the open row in this memory. If every time after accessing a row AUTOPRECHARGE command is performed, and the next access is actually involving the same row of the same bank, an ACTIVE command needs to be applied again in order to access last open row.

As it is known that time needed between ACTIVE and READ/WRITE commands has to be fulfilled. In a successive period of time a program probably only accesses a small part of continuous address space. So that it is not necessary to issue an AUTO-PRECHARGE command after every read/write access without judging if there's any need. If the current access is for an open row in an active bank, the READ or WRITE command is directly issued to the SDRAM, the ACTIVE and READ or WRITE commands.

When the auto-refresh is required, all the active banks will be inactivated by applying PRECHARGE ALL command as auto-refresh can only be issued when the whole SDRAM is in idle state. In this way, the overhead caused by frequently opening and closing the SDRAM banks can be decreased. The figure below is the whole architecture for our SDRAM controller which has the data path relating to read and write FIFO, AMBA interface and SDRAM.

The memory controller basically consist of three main sub parts, they are

A. 2 Read and 2 write FIFOs
B. Command generator
C. Command scheduler

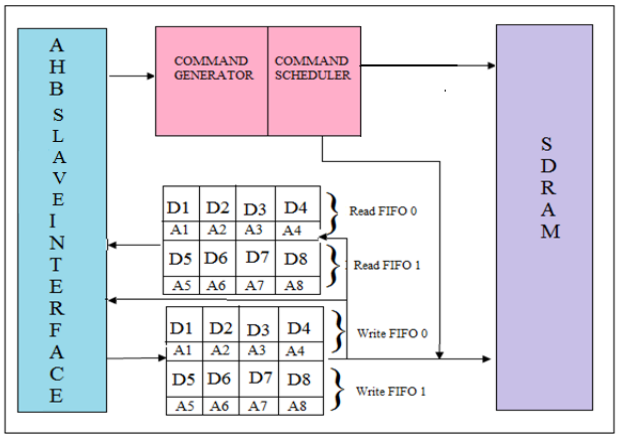

Figure: 3.1 Architecture of SDRAM controllers

3.2 Read FIFO

An AHB (Advanced High-performance Bus) transfer consists of two distinct sections, the address phase and the data phase. The address phase lasts only a single cycle. The address and some control signals are transferred from a master to a slave during this phase. During the data phase, data and responding signals are delivered. If a slave can't finish the data phase in one cycle, it can use HREADY to make the master hold the address and control signals. During the data phase, slave will send the corresponding response signals to the master to notify if the transfer is successfully finished or if it needs to retry the transfer.

It is known that, SDRAM cannot finish the data access in a single cycle. So we need some strategies to decrease the responding time. We analyzed our application and found that if the SDRAM is accessed in the single mode, the needing data is available at least after 2 cycles Plus time consumed by the latching in the bus interface part and the PRECHARGE command, in fact, it is more than 2 cycles. To support the fast response time, burst mode access and read FIFOing techniques need to be used whenever possible.

When AHB bus needs to read data, see if data is already in read FIFO by only checking if the current accessing address is in the range of either of the read FIFO and if the corresponding bit in the corresponding Valid Vector is valid. The Valid Vector is used to mark if the data stored in data FIFO is valid. Valid Vector works like tags for a cache. Every read FIFO is only as big as the capacity of a SDRAM burst, so that Valid Vector is only a several bit vector. If the needed data is in read FIFO, data can be directly read from them, otherwise, a READ command is needed issuing to SDRAM.

After a preset number of clock cycles, the data is available on the output latches of the SDRAM for reading, and data is delivered to AHB bus and written to one of the read FIFO at the same time. The whole burst access data will be loaded in read FIFO.

In this way, we implemented perfecting. According to the local principles of programs, the next read access is possibly a successive address to the current one. So the next data can be read from read FIFO, which can fasten the responding speed. In order to reduce the complexity of implementing read FIFO, data from SDRAM are stored in read FIFO 0 and read FIFO1 in turn.

3.3 Writing or ping-ponging

As it is described about the AHB, in order to reduce the responding time to write access, write FIFO are used to pack and align the data when the AHB bus data transfer size does not match the SDRAM data bus width. Based on our previous research on FPGA designs, we didn't use only one write FIFO, instead, we use 2 FIFO. It is well known that ping-ponging can reduce or eliminate the mismatch effects between 2 different modules which are operating at different speeds.

By utilizing ping-ponging between the two write FIFO, part of the time writing one FIFO and part of the time moving data from another FIFO to off chip SDRAM can be overlapped.

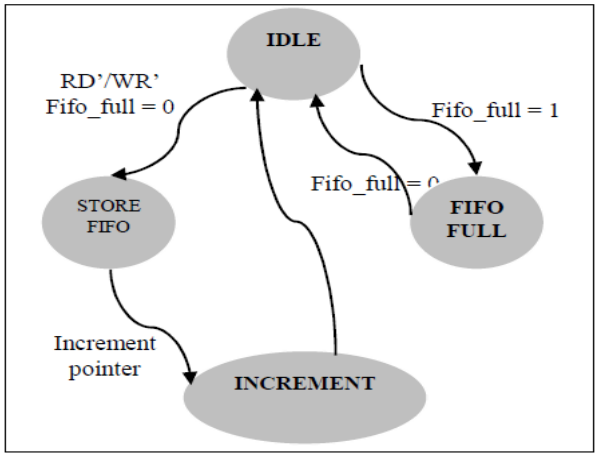

Figure: 3.2 FIFO state machines

If write FIFO 0 is not empty and write FIFO 1 is being written by AHB bus, move FIFO 0's data to SDRAM and this FIFO will be in the progress of moving until all data is moved to SDRAM, vice versa. This is the ping-ponging technique we used between write FIFO 0 and write FIFO 1 [1].

The flow of moving data from FIFO to SDRAM is as follows

- If write FIFO 0 is empty, data will be written to write FIFO 0. Else if write FIFO 1 is in the progress of AHB writing and write If write FIFO 0 is being written by AHB bus and write FIFO 1 is not empty, move FIFO 1's data to SDRAM and if FIFO 0 is not empty, move FIFO 0's data to SDRAM.

This flow explains how the ping-ponging can make the time writing FIFO and writing SDRAM overlap. The description of operations about AHB bus writing FIFO is as follows:

- If write FIFO 0 is empty, data will be written to write FIFO 0. Else if write FIFO 0 is not empty or in the progress of moving its data to RAM and if write FIFO 1 is empty, data will be written to write FIFO 1. Else make the AHB bus hold the bus signals until one of the FIFO is empty.

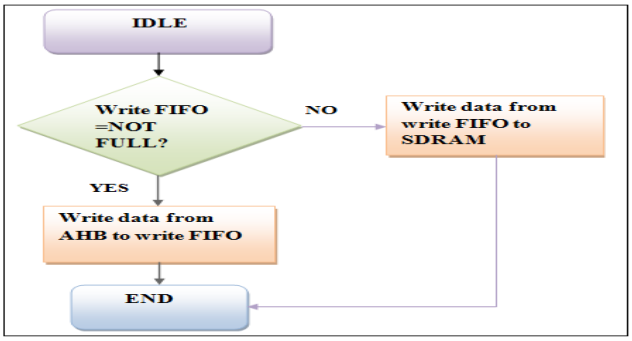

Figure:3.3 Single Data flow to and from the FIFOs

3.4 Read Update Logic

One of the novel features of this controller is the internal search which is carries out before giving the request to the DRAM. When the controller encounters a read command at the head of the FIFO queue, it takes this command and searches the FIFO to see if the required data is already in the FIFO.

This search is carried out in parallel and all the FIFO locations are searched in one clock cycle. Hence the searching time is not dependent on the depth of the FIFO but at the cost of hardware overhead. The flag bit (executed / not executed) aids in the search.

IV. IV SIMULATION RESULTS AND SYNTHESIS REPORT

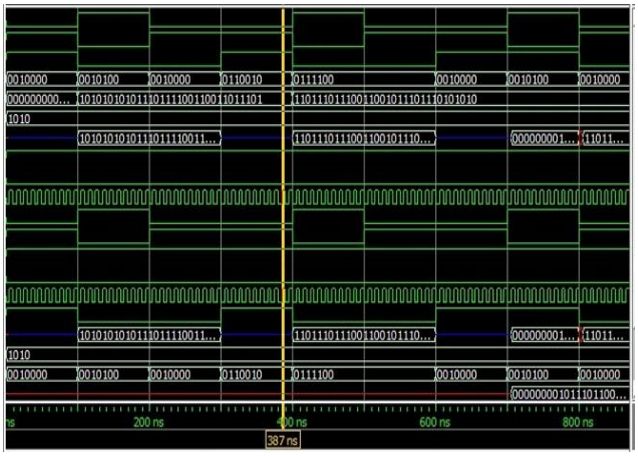

Figure: 4.1 SDRAM Memory

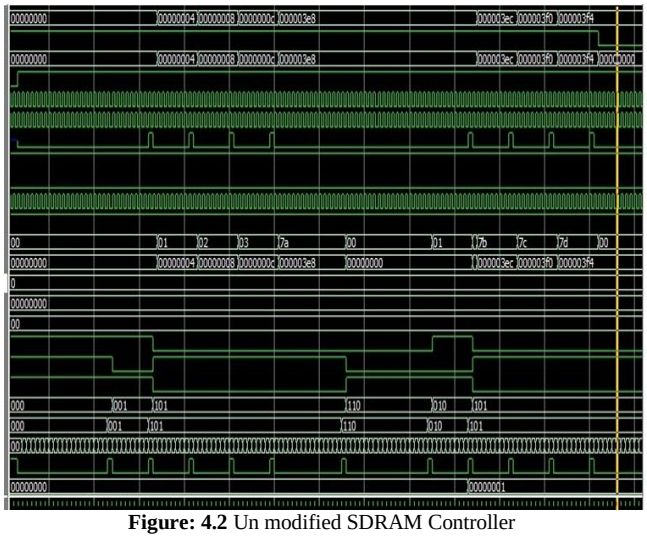

Figure: 4.2 Un modified SDRAM Controller

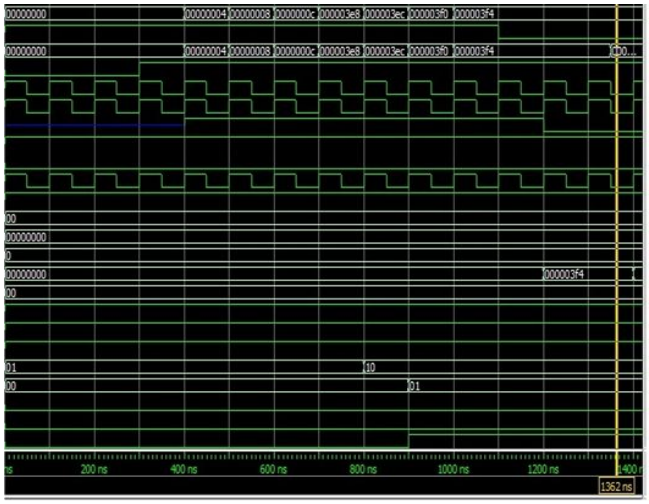

Figure: 4.3 Modified SDRAM Controller

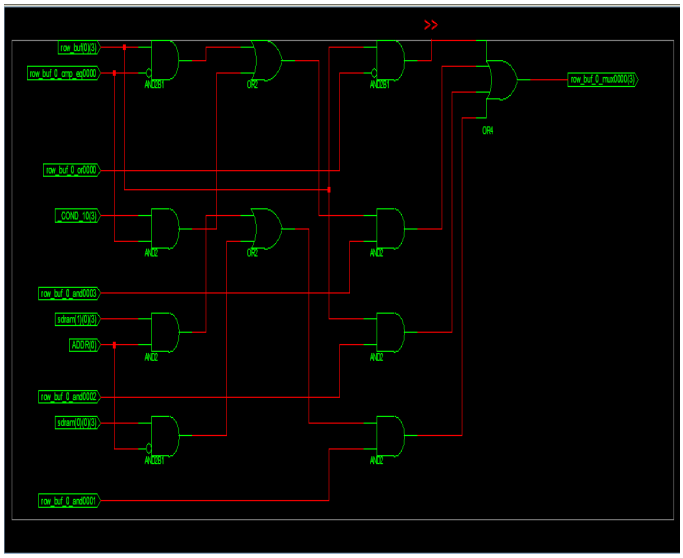

Figure: 4.4 SDRAM RTL schematic

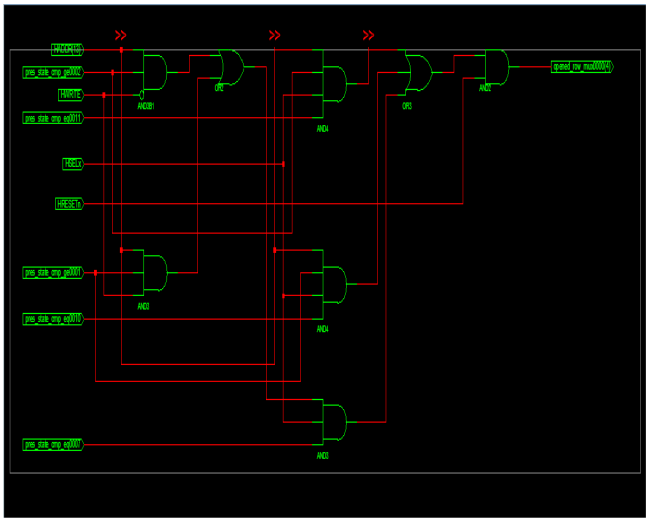

Figure: 4.4 SDRAM RTL schematic

Testing results between our controller and the traditional controller, the speedup rate difference is 2.1ns. Moreover, for the two SDRAM controllers, we compared the times of issuing READ and WRITE commands to off chip SDRAM while running the application program. It is shown in Table 17, running the same application program, modified SDRAM controller can reduce the times of accessing off chip SDRAM, which has a heuristic meaning for low power designs as well.

The results in table 1 are normalized according to those from the SDRAM controller described in this paper. The frequency of the whole SoC got from ISE is 92 MHZ for the existing SDRAM controller and 114MHZ for the modified SDRAM controller we described in this paper. We can see that our SDRAM controller has made a significant decrease of the total execution time of the application program.

Table: 1

Parameters	**Existing SDRAM**	**Modified SDRAM**
Minimum period	10.832ns	8.743ns
Maximum Frequency	92.319MHz	114.383MHz
Minimum input arrival time before clock	9.902ns	6.914ns
Maximum output required time after clock	12.479ns	10.075ns
Maximum combinational path delay	11.460ns	No path found

Conclusion

An AHB interfaced high-performance SDRAM Controller has been proposed, verified and evaluated we find this SDRAM controller has high performance by taking good use of the features of SDRAM architecture and utilizing the well-known techniques such as data FIFOing and ping-ponging and burst-mode-data transfer. In this paper testing results shows 91.66% of reduced delay with FIFOing when compared to the latency produced with FIFOing, which is nothing but in-built memory used within the SDRAM controller to improve its performance.

References

[1] Mohd Wajid, Shahank SB, ”Architecture for Faster RAM Controller Design with Inbuilt Memory”, IEEE ,2010

[2] Ching- SDRAM Controller Applications” .IEEE J. Solid-State Circuits, Vol..39, Nov. 2004. Che Chung, Pao-Lung Chen, and Chen-Yi Lee.

[3] Micron Technology Inc., Synchronous DRAM Data Sheet, 2001.

[4] ARM, AMBA Specification Rev.2.0, 1999.

[5] “Memory Controllers for Real-Time Embedded systems” Benny Akesson Kees Goossens vol. 3, no. 3, pp. 75–77, Mar1999.

[6] Hynix Semiconductor Inc., SDRAM Device operationRev.1.1, Sep. 2003.