REFERENCES

[1] Veena H K and Dr. A H MAs than Ali, Design and Implementation of High-Speed DDR SDRAM Controller on FPGA, International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 IJERTV4IS070428, Vol. 4 Issue 07, July-2015.

[2] Zude Zhou, Songlin Cheng, and Quan Liu, Application of DDR Controller for High-speed Data Acquisition Board, Innovative Computing Information and Control International Conference on 2006, pp. 611-614.

[3] Hadley M. Siqueira, Ivan S. Silva, Marcio E. Kreutz, Edgard F. Correa, DDR SDRAM Memory Controller for Digital TV Decoders, Brazilian Symposium on Computing System Engineering (SBESC) 2011, pp. 78-82.

[4] Micron Technology Inc. Double _Data Rate controller.

[5] Anders Nordstrom, Sub-cycle Functional Timing Verification Using SystemVerilog Assertions, Verilab Inc.

[6] CHRIS SPEAR, System Verilog for verification, ‖ Synopsys, Inc.

[7] www.testbench.in

[8] DDR SDRAM Wikipedia.

[9] Sonali, AshokKumar Patel, SantoshKrishna M., Design of DDR SDRAM controller for embedded system, Journal of Information, Knowledge and Research in Electronics and Communication Engineering, Nov 12 to oct13, volume-02, Issue-02.

[10] Deepali Sharma, Shruti Bhargava, Mahendra Vucha, Design and VLSI Implementation of DDR SDRAM Controller for high speed applications, International Journal of computer Science and Information Technologies, Vol.2 (4), 2011

[11] Double Data Rate (DDR) SDRAM Specification, JEDEC STANDARD JESD79E, May 2005.

[12] J.Bhaskar “A verilog HDL primer – 2 nd Edition.”

[13] John Wakerly “Digital systems design – 8 th Edition.”

[15] DDR SDRAM Controller white paper, Lattice Semiconductor Corporation, Reference Design: RD1020, April 2004.

[16] Michael John Sebastian Smith, “Application Specific Integrated Circuits”, 2004.

[17] Z. Peyton Peebles, Jr. "Digital Communication Systems," PrenticeHall, INC

[18] J. Shtargot, Design Guidelines for High-Performance, Multichannel, Simultaneous Sampling ADCs in DataAcquisition Systems‖, Maxim semiconductors.

[19] Anders Nordstrom, ―Sub-cycle Functional Timing Verification Using SystemVerilog Assertions, ‖ Verilab Inc.

[20] Using SystemVerilog Assertions in Gate-Level Verification Environments, Mark Litterick, DVCon 2006.

[21] CHRIS SPEAR, ―System Verilog for verification, ‖ Synopsys, Inc.

[22] SystemVerilog Assertions Design Tricks and SVA Bind Files, Cliff Cummings, SNUG 2009.

[23] Harry Foster, "Maturing a Project's ABV Process Capabilities."

Available at https://verificationacademy.com/sessions/maturing - abv - process - capabilities

[24] Harry Foster, Adam Krolnik, David Lacey, "Assertion Based Design, 2nd Edition," Springer, www.springeronline.com

[25] Clifford E. Cummings, SystemVerilog Assertions Design Tricks & SVA Bindfiles, ‖ SNUG 2009 (San Jose).

Available at www.sunburst - design.com/papers

[26] Clifford E. Cummings, SystemVerilog Implicit Port Enhancements Accelerate System Design & Verification, ‖ SNUG 2007 (Boston). Available at

www.sunburst - design.com/papers

PAPER PUBLICATION DETAILS

Published Two papers

1. Based on the project work the paper has been published in the **International Research Journal of Engineering and Technology (IRJET)** and the paper publication details have been given below.

Pavan Kumar M P[1], Dr. Subodh Kumar Panda[2] "**Design and VLSI Verification of DDR SDRAM Controller Using VHDL"** has been published in **International Research Journal of Engineering and Technology (IRJET)** Volume 6, Issue 3, March 2019 S.NO:942.

2. Based on the project work the paper has been published in the **IEEE ICICCS (3rd International Conference on Intelligent Computing and Control Systems) 2019.** and the paper publication details have been given below.

Paper entitled "**Design and verification of DDR SDRAM memory controller using SystemVerilog for higher coverage**" was successfully submitted to the **IEEE ICICCS (3rd International Conference on Intelligent Computing and Control Systems) 2019**. Paper got accepted and already presented technical seminar on 16th may 2019. Publication preceding is attached.

e-ISSN: 2395-0056 p-ISSN: 2395-0072

International Research Journal of Engineering and Technology (IRJET)

(An ISO 9001 : 2008 Certified Journal)

It is hereby awarding this certificate to

Pavan Kumar M P

In recognition the publication of the manuscript entitled

Design and VLSI Verification of DDR SDRAM Controller using VHDL

published in our Journal Volume 6 Issue 3 March 2019

Impact Factor : 7.211

www.irjet.net

Editor in Chief

E-mail : editor@irjet.net

VAIGAI

COLLEGE OF ENGINEERING

(Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai)

Vinayagapuram, Therkutheru, Madurai - 625122

Web: www.vaigai.org, E-mail: vaigaioffice@gmail.com

IEEE

Third International Conference on Intelligent Computing and Control Systems (ICICCS 2019)

Technical Sponsor: IEEE ED Coimbatore Chapter; ISBN: 978-1-5386-8113-8

CERTIFICATE OF PRESENTATION

This is to certify

PAVAN KUMAR. M. P

has presented the paper entitled

Design and Verification of DDR SDRAM Memory Controller using Systemverilog for Higher Coverage

in the Third International Conference on Intelligent Computing and Control Systems (ICICCS-2019), organized by Vaigai College of Engineering during May 15-17, 2019 at Madurai, India.

Session Chair

Conference Chair
Prof. P. Sugumaran

Principal
Dr. K. Maniysundar

International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 03 | Mar 2019 www.irjet.net p-ISSN: 2395-0072

Design and VLSI Verification of DDR SDRAM Controller Using VHDL

Pavan Kumar M P[1], Dr. Subodh Kumar Panda[2]

[1]*M. Tech (VLSI Design and Embedded systems), ECE Department, BNMIT, Karnataka, INDIA*
[2]*Associate Professor, ECE Department, BNM Institute of Technology, Karnataka, INDIA.*

Abstract - *Present days, DDR SDRAM (Double Data Rate Synchronous Dynamic Random-Access Memory) has become the most popular class of memory used in computers due to its high speed, burst access and pipeline feature. For high speed applications like image/video processing, signal processing, networking etc. DDR SDRAM is widely used. The basic operations of DDR SDRAM controller are similar to that of SDR (Single Data Rate) SDRAM; however, there is a difference in the circuit design; DDR simply use sophisticated circuit techniques to achieve high speed. To perform more operations per clock cycle DDR SDRAM uses double data rate architecture. DDR SDRAM (also known as DDR) transfers data on both the rising and falling edge of the clock. The DDR controller is designed with objective of proper commands for SDRAM initialization, read/write accesses, regular refresh operation, proper active and pre-charge command etc. DDR SDRAM controller is implemented using Verilog HDL and simulation and synthesis is done by using Modalism 6.6d and Xilinx ISE 9.2i accordingly.*

Key Words: *DDR, SDRAM, burst access, pipeline, Verilog HDL.*

1.INTRODUCTION

Double data rate synchronous dynamic random-access memory (DDR SDRAM) is a class of memory integrated circuits used in computers. Nowadays, Memory devices are almost found in all systems, high speed and high-performance memories are in great demand. For better throughput and speed, the controllers are to be designed with clock frequency in the range of megahertz. As the clock speed of the controller is increasing, the design challenges are also becoming complex. Therefore, the next generation memory devices require very high-speed controllers like double data rate and quad data rate memory controllers.

With the rapid development in the processor's family, speed and capacity of a memory device is a major concern. The DDR is an enhancement to the traditional synchronous DRAM. The DDR is able to transfer the data on both the edges of each clock cycle. Thus, doubling the data transfer rate of the memory device. The DDR is available in a very low cost that's why it is widely used in personal computers where they are basically used to provide the functions of storage and buffers. The DDR SDRAM supports the data widths of 16, 32 and 64 bits. Its automatic refresh during the normal and power down modes. The DDR is a complete synchronous implementation of controller. It increases the throughput using command pipelining and bank management. This improvement allows the DDR module to transfer data twice as fast as SDRAM. As an example, instead of a data rate of 133MHz, DDR memory transfers data at 266MHz.DDR modules, like their SDRAM predecessors, arrive in there. Although motherboards designed to implement DDR are similar to those that use SDRAM, they are not backward compatible with motherboards that support SDRAM. You cannot use DDR in earlier SDRAM based motherboards, nor can you use SDRAM on motherboards that are designed for DDR.

1.1 Random Access Memory

Random access memory (RAM) is the best-known form of computer memory. RAM is considered "random access" because you can access any memory cell directly if you know the row and column that intersect at that cell. (RAM) data, on the other hand, can be accessed in any order. All the data that the PC uses and works with during operation are stored here. Data are stored on drives, typically the hard drives. However, for the CPU to work with those data, they must be read into the working memory storage (RAM).

1.2 Types of Random-Access Memory

1.2.1. Static Random-Access Memory

Static Random-Access Memory uses a completely different technology. In static RAM, a form of flip-flop holds each bit of memory. A flip-flop for a memory cell takes four or six transistors along with some wiring, but never has to be refreshed. This makes static RAM significantly faster than dynamic RAM. However, because it has more parts, a static memory cell takes up a lot more space on a chip than a dynamic memory cell. Therefore, you get less memory per chip. Static Random-Access Memory uses multiple transistors, typically four to six, for each memory cell but doesn't have a capacitor in each cell. It is used primarily for cache. So static RAM is fast and expensive, and dynamic RAM is less expensive and slower. So, static RAM is used to create the CPU's speed-sensitive cache, while dynamic RAM forms the larger system RAM space.

1.2.2. Dynamic Random-Access Memory

Dynamic Random-Access Memory has memory cells with a paired transistor and capacitor requiring constant refreshing. DRAM works by sending a charge through the appropriate column (CAS) to activate the transistor at each bit in the column. When writing the row lines contain the state the capacitor should take on. When reading the sense amplifier determines the level of charge in the capacitor. If it is more than 50 percent, it reads it as a 1 otherwise it reads it as a 0.

A memory chip rating of 70ns means that it takes 70 nanoseconds to completely read and recharge each cell. It is one of the most common types of computer memory (RAM). It can only hold data for a short period of time and must be refreshed periodically. DRAMs are measured by storage capability and access time. Storage is rated in megabytes (8 MB, 16 MB, etc.). Access time is rated in nanoseconds (60ns, 70ns, 80ns, etc.) and represents the amount of time to save or return information. With a 60ns DRAM, it would require 60 billionths of a second to save or return information. The lower the speed, the faster the memory operates. DRAM chips require two CPU wait states for each execution. Can only execute either a read or write operation at one time. The capacitor in a dynamic RAM memory cell is like a leaky bucket. It needs to be refreshed periodically or it will discharge to 0. This refresh operation is where dynamic RAM gets its name.

Memory is made up of bits arranged in a two-dimensional grid. In which memory cells are fetched onto a silicon wafer in an array of columns (bit lines) and rows (word lines). The intersection of a bit line and word line constitutes the address of the memory cell. Memory cells alone would be worthless without some way to get information in and out of them. So, the memory cells have a whole support infrastructure of other specialized circuits. Identifying each row and column (row address select and column address select) Keeping track of the refresh sequence (counter) Reading and restoring the signal from a cell (sense amplifier) Telling a cell whether it should take a charge or not (write enable) Other functions of the memory controller include a series of tasks that include identifying the type, speed and amount of memory a checking for errors. The traditional RAM type is DRAM (dynamic RAM). The other type is SRAM (static RAM). SRAM continues to remember its content, while DRAM must be refreshed every few milliseconds.

1.2.3. Double Data Rate Synchronous Dynamic Random-Access Memory

Double Data Rate Synchronous Dynamic Random-Access Memory is the original form of DDR SDRAM. It is just like SDRAM except that is has higher bandwidth, meaning greater speed. Maximum transfer rate to L2 cache is approximately 1,064 MPs (for DDR SDRAM 133 MHZ). DDR RAM is clock doubled version of SDRAM, which is replacing SDRAM during 2001- 2002. It allows transactions on both the rising and falling edges of the clock cycle. It has a bus clock speed of 100MHz and will yield an effective data transfer rate of 200MHz. DDR come in PC 1600, PC 2100, PC 2700 and PC 3200 DIMMs. A PC 1600 DIMM is made up of PC 200 DDR chips, while a PC 2100 DIMM is made up of PC 266 chips. Go for PC2700 DDR. It is about the cost of PC2100 memory and will give you better performance. DDR memory comes in CAS 2 and CAS 2.5 ratings, with CAS costing more and performing better.

This paper is organized as follows. In Section 2, Block diagram of DDR controller and architecture of DDR controller will be described. In Section 3, different functional blocks will be explained. Finally, the Result and Conclusion will be given in Section 4 and Section 5 respectively.

2. DDR CONTROLLER ARCHITECTURE

The DDR SDRAM memory controller centre comprises of four primary hinders that are SDRAM controller, Control Interface, Command and Data Path Modules. The DDR SDRAM controller module is the best level module. That embodies three lower modules and unites the entire outline to give exact outcomes. The Control Interface module of SDRAM approaches summons and related memory addresses from the host and translating the charges and passing the look to the Command module. The Command module acknowledges summons and address from the Control Interface module for creating the correct orders to the SDRAM. The Data Path module handles the information way activities amid WRITE and READ orders. The best level module additionally instantiates two PPL's that are utilized as a part of CLOCK_LOCKS.

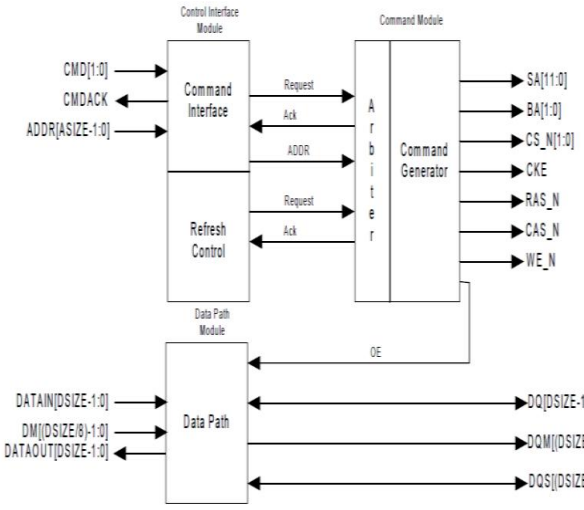


Figure 1: Functional Block Diagram of proposed DDR SDRAM Memory Controller core.

3. DIFFERENT FUCTIONAL BLOCKS

3.1 DDR SDRAM Control Interface Module

The control Interface Module deciphers and registers charges from the host and tracks the decoded NOP, WRITEA, READA, REFRESH, PRECHARGE, and LOAD_MODE summons, alongside ADDR, to the Command Module. The LOAD_REG1 and LOAD_REG2 summons are decoded and utilized intramural to load the REG1 and REG2 registers with the qualities from ADDR. The Control Interface Module additionally contains a 16-bit down counter and control circuit that is utilized to generate periodic invigorate charges to the Command Module. A 16-bit down counter is stacked with the most noteworthy esteem of REG2 and checks down to zero. The REFRESH_REQ yield is cited with the counter achieves zero and remains cited until the point that the Command Module recognizes the demand. The recognize from the Command Module attaches the down counter to be reloaded with REG2 and the procedure rehashes. REG2 is a16-bit esteem that speaks to the period between REFRESH summons that the controller issues. The esteem is set by the condition int (REF_PERIOD/CLK_PERIOD).

3.2 DDR SDRAM Command Module

The summon module gets decoded orders from the Control Interface Module, alongside invigorate demands from the revive control rationale and effectuates the suitable orders to the SDRAM. The module includes a basic referee that referees between the orders from the host interface and the invigorate demands from the revive control rationale. The invigorate supplications from the revive control rationale have need over the charges from the host interface. On the off chance that a summon from the host appears in the meantime or amid a dormant revive task, the mediator holds off the host by not declaring CMDACK until the point when the inert invigorate activity is finished. On the off chance that a hid invigorate order is gotten while a host task is in advance then the concealed revive is held off until the point when the host activity is finished. After the authority has gained a charge from the host, the summon is passed onto the order generator segment of the Command Module. The summon Module utilizes three move registers to provoke the hopeful planning between the charges that are issued to the SDRAM. One move enlist is utilized to control the planning of the ACTIVATE summon while a moment moves enlist is utilized to control the situating of the READA or WRITEA charges and the third move enlist is utilized for timing order terms, all together for the judge to decide whether the last asked for task is finished. The Command Module likewise plays out the convolution of the address (ADDR) to the SDRAM. The column section of the deliver is multiplexed out to the SDRAM yields A [11:0] amid the ACTIVATE (RAS) charge. The section fragment is then multiplexed out to the SDRAM address yields amid a READA (CAS) or WRITEA charge.

3.3 DDR SDRAM Data Path Module

One of the most difficult aspects of DDR SDRAM controller design is to transmit and capture data at double data rate. This module transmits data to the memories. The basic function of data path module is storing the write data and calculates the value for read data path.

The Data Path Module issues the SDRAM information interface to the host. Host information is acknowledged on port DATAIN for WRITEA charges and information is given to the host on port DATAOUT amid READA commands. The information path width into the controller is twice the information way width to the DDR SDRAM gadgets. The information path module's DATAIN and DATAOUT ports are settled at 32bits and the DQ port is settled at 16bits. To construct data paths bigger than 32bits, Data Path Modules can be fell to expand the information transport width for increases of32bits. The DATAIN compose way takes the compose information and exchanges over to the CLK200 (two times the frequency of CLK100), multiplexing the information to the DQ port. On the read side, DATAOUT way, the read data is capture utilizing the flag CLK200 to test the information amid the centre of the information substantial window. The read data is demultiplexed and exchanged to the flag CLK100 clock space, which is one a large portion of the recurrence of CLK200. The Data Path Module additionally creates the DQS motion amid compose tasks to the DDR gadgets.

Data path module handles all the data generation and sampling task of DDR controller. For Read access, data is sampled by data path. Data is then synchronized with the internal clock and transferred to the user interface one-word per clock cycle as normal data rate. For Write access, data is received from the user interface at the normal one word per clock data rate. The DDR controller's data path then resynchronized the data and transfers them using the double data rate.

4. RESULTS

The design is simulated and synthesized on Modelsim 6.6d and Xilinx ISE 9.2i accordingly. The simulation and synthesis results are shown in the following figures.

The DDR SDRAM memory controller designed and analyzes the operations of read, write, auto precharge, precharge, no operation, mode register, load_register1 and load_register2 by using DDR SDRAM micron specifications. To test the main control module, initially the clock is given 100MHZ. The module will not be initialized until reset_n signal is made high. Once this signal is made high, it is necessary to make the delay signal (sys_dly_10us) high. This implies that the controller is provided with the necessary delay. After providing the delay signal, after some ns delay the controller gets initialized. This is indicated by the signal (sys_init_done) becoming high. Reset is made high and required system delay (10us) is given. Once the system gets initialized, the required data is written to the particular memory location. Now we can read the same data from memory.

4.1 Synthesis Result: From Xilinx Tool.

Figure 2 shows the RTL schematic of designed DDR SDRAM controller.

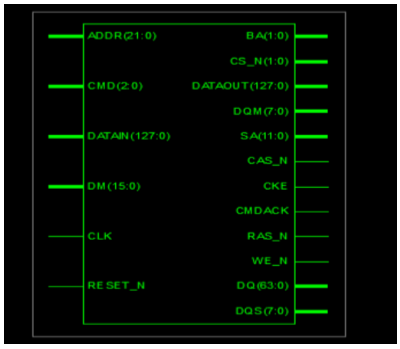


Figure 2: RTL schematic of DDR Controller

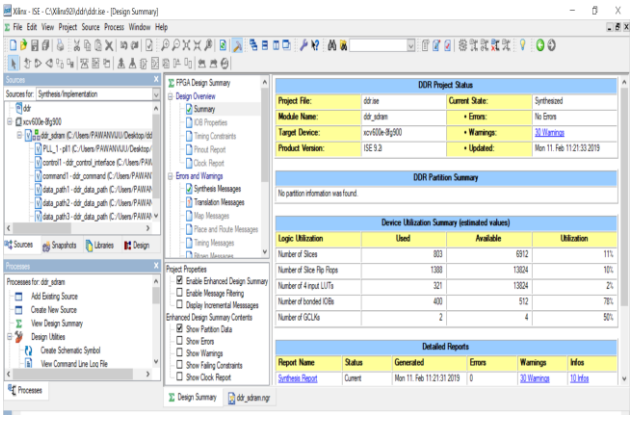


Figure 3: synthesis Report of DDR SDRAM Verilog HDL code.

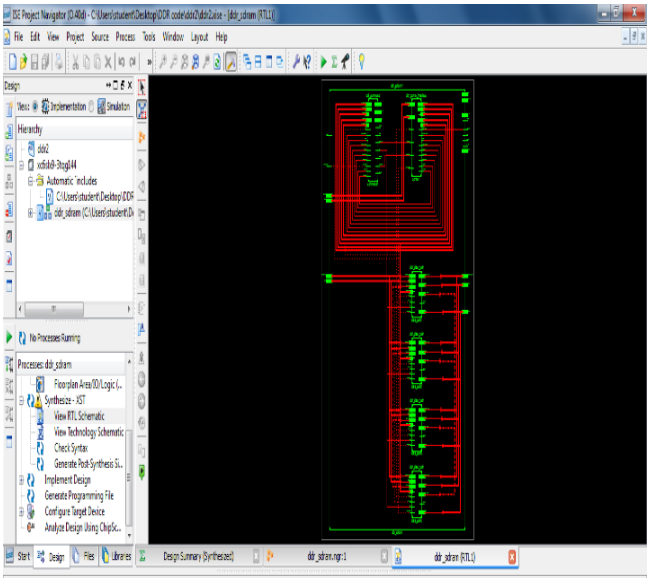


Figure 4: RTL schematic of DDR Controller

4.2 Simulation Results: From Modelsim tool.

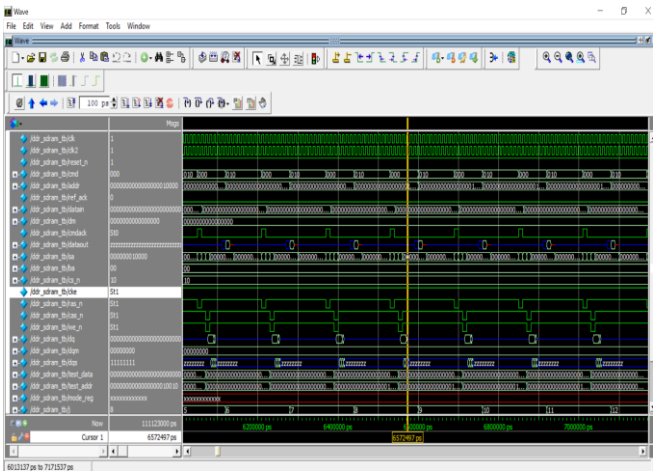


Figure 5: DDR SDRAM memory controller verification wave form

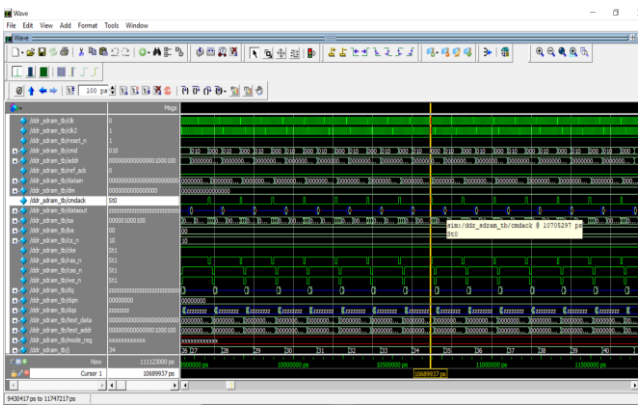


Figure 6: DDR SDRAM memory controller design wave form.

5. CONCLUSIONS

In this paper an efficient fully functional DDR SDRAM controller is designed and verified by using Verilog HDL. The controller generates different types of timing and control signals, which synchronizes the timing and control the flow of operation. The memory system operates at double the frequency of processor, without affecting the performance. Thus, we can reduce the data bus size. The drawback of this controller is complex schematic with large number of buffers in the circuit increases the amount of delay.

REFERENCES

[1] Micron Technology Inc. Double Data Rate (DDR) SDRAM, Datasheet, 1999
[2] Synthesizable DDR SDRAM controller, Application note, 2002. (Xilinx Inc., XAPP200)
[3] Using the vertex select I/O Resource, Application note, 1999. (Xilinx Inc., XAPP133)
[4] Using Delay Locked Loops in Spartans-II FPGAs. (Xilinx Inc., XAPP174)
[7] Bhaskar "A Verilog HDL primer- 2nd edition"
[8] John Wakerley "Digital system design – 8th edition."
[9] Burkhardt, A., Hekstra-Nowacka, E., and Chauhan, A., "A Real-time Streaming Memory Controller", Design,

BIOGRAPHIES

Pavan Kumar MP Received his B.E. degree in Electronics & Telecommunication Engineering From GMIT Davangere, Karnataka India in 2016. Presently pursuing his MTech. (Specialization in VLSI Design and Embedded system) from BNM institute of Technology, Bangalore, Karnataka, India from 2017-2019.

Dr. Subodh Kumar Panda is Presently working as an Associate Professor in Department of ECE, BNM institute of Technology, Bangalore, Karnataka, India. He has 18 years of teaching experience. His areas of interest are Digital system design Using VHDL, Verilog, Embedded systems and Sensors.

Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2019)
DVD Part Number:CFP19K34-DVD; ISBN: 978-1-5386-8112-1

Design and Verification of DDR SDRAM Memory Controller Using SystemVerilog For Higher Coverage

Pavan Kumar M P
M.Tech student, VLSI & Embedded System
BNM Institute of Technology
Bangalore, India
pavanmp54@gmail.com

Dr. Subodh Kumar Panda
Associate Professor, Dept. of ECE
BNM Institute of Technology
Bangalore, India
subodhpanda2013@gmail.com

Abstract— **In present electronic systems, DDR SDRAM (Double Data Rate Synchronous Dynamic Random-Access-Memory) is an next level advanced version of regular SDRAM, and it was developed with advanced key features such as effective use of memory bandwidth and its capability to transact data on both edges of clock cycles. DDR SDRAM is widely used in computer applications like laptops, DSP processing systems and networking. Cost and speed are the two important factors in designing memories like DDR SDRAM which will meet the standards in the field of DSP applications. Because of its high speed, burst access and pipeline feature DDR SDRAM becomes more popular. The main basic operations of DDR SDRAM memory controller are very much common to that of SDR (Single Data Rate) SDRAM memory controller and they differ only in their circuit design. DDR simply use sophisticated circuit techniques to achieve high speed in order to perform a greater number of operations per clock cycles. DDR SDRAM uses double data rate architecture wherein DDR SDRAM (also known DDR1) means transaction of data on both the rising and falling edge of the clock cycles. The DDR SDRAM controller makes many low-level tasks invisible to the user like refresh, initialization and timings. DDR SDRAM also designed with objective of using proper commands like Read/Write accesses, proper active and pre-charge command etc. In this work a DDR SDRAM controller is designed using Verilog HDL and Verification is carried out using SystemVerilog by Questasim Tool. Functional coverage of 100% is achieved by applying randomized test cases.**

Keywords— DDR-SDRAM, System verilog, Randomization, Coverage, Assertions.

I. INTRODUCTION

With the high increase in development of the semiconductor processor's industry, major importance given to operation speed and capacity of the memory device. The DDR controller is an extension of old traditional memory controller synchronous DRAM (SDRAM). The DDR controller is capable of transacting the data on both rising and falling edges of the clock cycles. Thereby, double the transfer of data rate in the memory device. The cost of DDR memory is low, because it is most commonly used in PC's, storage and buffers. The DDR SDRAM controller's capability of data widths of 16, 32 and 64 bits and its throughput increases by using pipelining command and bank management techniques. Changes in pipelining command and bank management techniques enhance DDR SDRAM memory module transfer data twice as speed as SDRAM memory. For an example, using a data rate of 133 MHz, DDR SDRAM memory transfer data rate at 266 MHz, that it means twice the frequency of clock cycles[1][2].

Compared to SDR SDRAM, DDR SDRAM have the burst length of 2, 4 and 8, thus one address is required to access sequential address [3]. There are two types of memories that can be purchased, namely Single In-line Memory Modules (SIMMs) and Double In-line Memory Modules (DIMMs). When in SIMM or DIMM both are small circuit that is capable to fit into common standard memory socket available in PC's. In DIMMs has two leads compared to the SIMMs has only one lead. They are soldered to the motherboard to supply desired memory. SDRAM memory usually comes in DIMMs. DIMMs are classified based on the data transfer rate, such has one is Single Data Rate (SDR) and other called Double Data Rate (DDR). SDR SDRAM transaction of data only on the rising edge of the clock cycle whereas DDR SDRAM transaction of data on both the rising as well as on the falling edge of the clock cycles, At present, other faster versions of available DDR SDRAMs, are DDR2, DDR3 and DDR4 [4].

II. DDR SDRAM MEMORY CONTROLLER ARCHITECTURE

A. DDR SDRAM Functional Blocks

The DDR SDRAM memory controller centre consists of four primary components that are basically SDRAM controller, control interface, command and data path module. The one main DDR SDRAM controller module is the high-level module. That mainly contains three lower level modules and gives out the entire outline to give exact outcomes. The Control Interface module of SDRAM makes an approach summons and that are related to memory addresses from the host and translating that charges and Transferring to the module called Command. The module command acknowledges from the SDRAM signals and address from the module called control interface module for creating the right way of building orders to the SDRAM controller. The module

data path manages or makes sure of the information way activities WRITE and READ orders. The SDRAM control interface is the best level module that additionally instantiates two phase locked loops that are effectively use as a part of CLOCK_LOCKS [3].

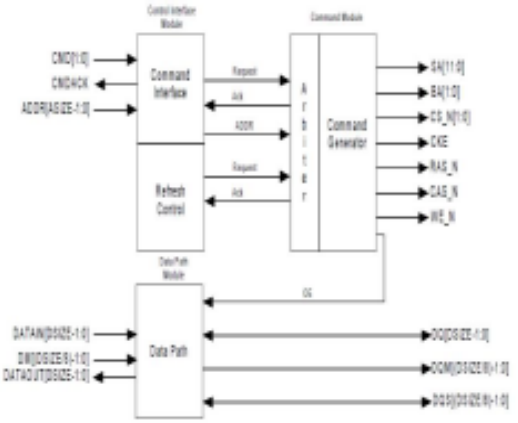


Figure 1: Block Diagram of proposed functional DDR SDRAM Memory Controller core.

B. DDR SDRAM Control Interface Module

The module control Interface is the primary module and it included the finite sate machine a programmable 16-bit counter, which is used to perform the auto refresh operation. The controller interface module accepts the commands from the processor and it will decode them and send the decoded WRITEA, NOP, READA, PRECHARGE, REFRESH, and LOAD_MODE signal commands are passed to the module command. There are two registers used, LOAD_REG1 and LOAD_REG2 signals are decoded and utilized to load the register REG1 and register REG2 from address ADDR signals The DDR SDRAM uses a different means separate clock signals: one is CLK and other is CLK` (the crossed of CLK going high and CLK` going low is just taken as the positive edge of the clock). The commands are taken down only on the positive edge of the clocks. The two signals READ and WRITE are the basic commands are used to access the SDRAM memory. The DDR SDRAM will support the burst length of 2, 4, or 8 locations for programmable READ/WRITE operations. When access to SDRAM started at any location and it will continue until it reaches the burst length. The READ and WRITE signal operation started by sending the ACTIVATE commands.

The controller Interface consists of a finite state machine shown in figure 2. The initial state is considered as ideal state and the next state of the controller can be in either PRECHARGE, REFRESH, LOAD_MODE, or ACTIVATE state those all states depending on the request from the processor [5].

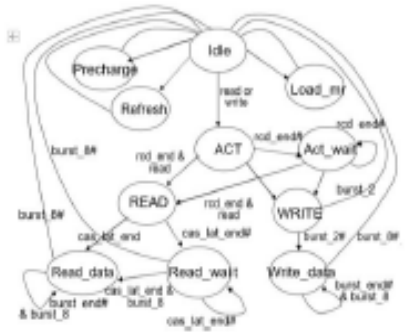


Figure 2: Finite state machine diagram

Those lines without indications in the figure 2 indicate an automatic sequence. The READ and WRITE command operation are followed by command ACT. The ACT command signal is used for particular bank to open a row, and where PRECHARGE command signal is used to close a bank which is open and if the open bank is other than that we want to access. When a WRITE command is passed and executed, the controller will first go to the ACT command state and then tit will enter to the WRITE state. The WRITE command operation will be executed till the burst length is completed or its Up to the burst terminate is inserted. When the READ operation is executed, the initial address is registered and during the READ operation the data will comes or arrival time is 1-3 clock cycles later on the data bus and this delay is because of the time that is actually needed to read the internal DRAM memory. This delay time is CAS latency. Once on completion of the burst READ and WRITE operations the controller will go settle back to the initial IDLE state.

C. DDR SDRAM Command Module

The command module mainly consists of an arbiter, multiplexer and three shift registers. An arbiter is used while in case if multiple requests come from different-different devices. If both high- priority and low- priority device need to transfer data at the same time then high priority device gets higher priority. The low priority device not able to transfer data until the high priority device has transferred all its data. In my work high priority is REFRESH control operation. So, if the REFRESH operation is executing on and thereby command from the host comes, then the controller will first continue the refresh operation to be done and it hold command requests by not assert this command to the command acknowledge (CMDACK). Once refresh operation is finished the command module will start performing the requested commands from the host. Module inside command module is the shift registers and is used to maintain the timing between the command signals assigned to the SDRAM memory controller. Other module it contains is multiplexer which is used to multiplexing the address. Where the row address is multiplexed to SDRAM during when the signal ACTIVATE (RAS) command. Other column portion is also multiplex the

address to SDRAM address outputs happen during a signal READ (WRITE) command [3][6].

D. DDR SDRAM Datapath module

The basic function of module data path is to store the write data and calculation for the value of read data path. Its main operation is to data generation and sampling of data to DDR. When the READ operation is executed, the data is sampled from the module data path. Data will synchronize the clock signal and transacted as one-word per clock cycle. When the WRITE operation is executed, the user interface gets the data at one word per clock cycle. The data will be resynchronized by the module data path and it will transfer at a rate called DDR double data rate.

The module data path performed as an interface between the processor and the memory. The data path module performs the two operations called latching of the data and dispatching of the data between the host and Memory. The module data path has tristate buffer which is controlled by OE signal, that comes from the module command. Whatever the data to be written is received on the DATAIN pin during the WRITE operation and during the READA operation data is provided on the DATAOUT pin [3]

III. Verification Enviornment

Verification is the most significant side of the VLSI configuration stream. It intends to discover the bugs in the RTL (Register Transfer Level) plan at a beginning time with the goal that it doesn't demonstrate out dangerous at the later stage in the outline procedure. Around 70% of the time is distracted in the verification procedure. In this way, it is the most time distracting process. Due to the enhance in number of transistors in the incorporated circuit (IC), lessening highlight estimate and enhanced outline apparatuses, the many-sided quality of the IC has expanded. This raises the likelihood of occurrence of bugs in the outline. Thus, the requirement for the confirmation of the IC ended up fundamental [10].

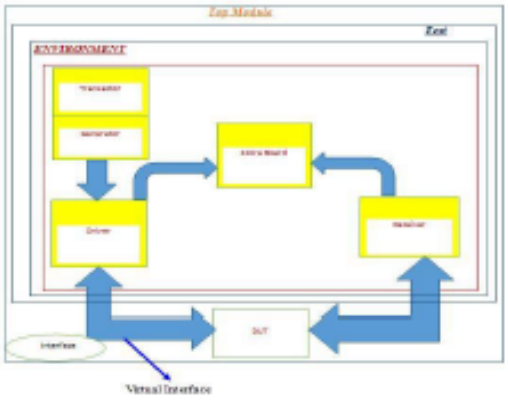


Figure 3: DDR SDRAM memory controller verification environment.

A. Transactor Class

In this class the inputs are declared with RAND and RANDC key words and the type of inputs declaration must be same as in the interface. Declare the outputs using the key word bit. Take one static variable of int type to store the no of transaction_ids. Display the input variables, output variables and transcation _id in display method which is void type function. Pass the arguments transaction class handle as input, output string message for displaying the success or failure (it is optional one) to compare function. Finally call the display function in post_randomize () method (in-built method).[10]

B. Generator Class

Create the 2 handles for transaction class, one for randomizing the input data and another for transferring data to the driver. Define a mailbox which is connecting the generator and driver at the time of object creation of generator class. Define the function new () constructor and pass mailbox as argument to provide connection between generator and driver. Provide connection between generator and driver. Create the object for transaction class. Define the function generate_data () here in this function input data is going to be randomized. Check the data is randomizing are not using randomize () method which is in-built one. Define virtual task start () and to put the randomized values into driver through mailbox.

C. Driver Class

First of all, declare the handle for virtual interface and 2 mailboxes. One is used for getting the inputs from generator, other one for driving the inputs to scoreboard. Define the handle for transaction class. Declare one event which is used to control the functioning of operation in correct way. Define a function new () constructor and pass above declared virtual interface, mailboxes, event as arguments and provide end to end connections. Write virtual task drive () to driving the values to virtual interface (it internally connects to DUT in top module) by receiving the data from generator via mailbox. Give some delay to complete the given task (driving the inputs to virtual interface) then trigger the event handler. Define the virtual task start () it is going to put the randomized values to scoreboard via mailbox.

D. Receiver Class

The receiver is also known as monitor in this declare the handle for virtual interface and mailbox which is used for putting the received data from receiver to scoreboard. Define a function new () constructor and passes above declared virtual interface, mailboxes, event as arguments and provide end to end connections. Write the virtual task receive () it will check the event handler is triggered are not. If it is triggered data is drived successfully and is not it will display the event driven is not triggered. And we are receiving data values from virtual interface.

Define virtual task start () and put received data to scoreboard via mailbox.

E. Score Board Class

This class defines 2 mailboxes one is generator to driver mailbox, another one is receiver to scoreboard. Declare the 3 handles for transaction class, for data transfer from driver, receiver and other for coverage purpose. Declare one event handler for verifying coverage using int variable. Define the cover group for coverage purpose (coverpoints, bins) and function new () constructor in this provide connections to mailboxes and create object for covergroup. Declare virtual task start (); and call virtual task check i.e. this virtual task check (); will compare function which is defined in transaction class. If compare function is not successful it will display the values stored in variables. If compare function, is successful then it will trigger the coverage by calling in-built functions like sample (). The data_verified is greater than or equal to no_of_transactions trigger the event for generating the report write the function report which will not return anything.

F. Enviornment Class

Using compiler directive (`include) copy the files (transactor, generator, driver, receiver, and scoreboard). Declare virtual interface handles and create the objects for mailboxes which are generator to driver, driver to scoreboard and receiver to scoreboard. Declare HANDLES for each and every class. Write the function new () constructor inside the new function to provide the end—to –end connections between blocks (generators, driver, receiver and scoreboard) by passing the mailboxes. Write the virtual task build () in this and create the objects for all classes. Write the virtual task start () and it will call the start (); task of all classes. Write the virtual task stop () it will stop the execution after the event is triggered. Write the task run () and call all the virtual tasks (build (), start (), stop () … and scoreboard report () function…).

G. Test Program

The test program is executing in reactive region and only one time it will execute and not allow always block to execute. Pass the interface modports (driver, receiver). [ex: program test_case_name (interface_name. modport_name. handler name) Import the whole package into test program. Create the handle to the environment class. Inside the initial block create the object for environment class and can modify the no_of _transactions and we invoke the build (), and run () methods of environment class.

H. Multiple Test cases

The test program is executing in reactive region and only one time it will execute and it will not allows always block. Pass the interface modports (driver, receiver). [ex: program test_case_name (interface_name. modport_name. handler name) Write the class inside the program inside the class we can write the constraints and we can declare the input variables as RANDC Declare the handles for environment class and the class which is written newly in program. Inside the initial block we are going to create the object for environment class and object for newly written class and assign the this newly class handler to generator and we can modify the no_of _transactions and we invoke the build (), and run () methods of environment class.

I. Interface

Declare all inputs and outputs with logic. We can write clocking blocks for driver and receiver (in the case of sequential ckts). We will write the modports for driver and receiver (modport are providing the particular connections in one direction…).

J. Top Module

Declare any global signal are present in the design like clock and reset. Define handle to interface and Instantiate test class. Instantiate the DUT and provide the connections by using name-based connections via interface handler and generate the clock signal.

K. SystemVerilog Assertion

The affirmation-based confirmation is an extreme administration to plan and check engineers who are vowed for the RTL outline of computerized pieces and systems. Declaration based check engineers catch the confirmation data amid the plan time and it additionally empowers inner state, information way, and blunder precondition scope examination. Attestations are primarily used to approve the conduct of an outline. The statements are check inserted in, or bound to an outline unit amid the reenactment. Notices or blunders are producing on disappointment of particular condition or grouping of occasions. The attestations are composed in HDL, yet HDL declarations can be long and entangled. This catastrophe the reason for statements, which is guarantee the accuracy of the outline. Extensive, complex HDL declarations can be difficult to make and subject to bugs themselves. System Verilog has highlights to determine declarations of a system and affirmations are indicates a conduct of the system. System Verilog Assertions can be grouped into two kinds.

I. Immediate Assertions

The immediate attestation articulation is a request of an articulation performed when the announcement is executed in the procedural code. On the off chance that the articulation assesses to X, Z or 0, at that point it is translated as being false and the affirmation is said to be fizzled. Something else, the articulation is translated as being valid and the affirmation is said to be passed. Prompt declarations are for all intents and purposes proportionate of VHDL attest

proclamation. Ordinarily favored $fatal, $error and $warning can be smothered in an instrument particular way, so that $info tells the declaration disappointment conveys no particular seriousness to print a few messages for pass or fizzled code. $display or any normal Verilog code, such as setting off an occasion or increasing a counter or calling capacity in pass/come up short piece code.

II. Concurrent Assertions

Simultaneous affirmations depict the conduct that reaches after some time. These are assessed just at the event of a clock ticks. The estimations of factors utilized as a part of this assessment are tested esteems. The result will acquired from the examination and paying little mind to the test system's inward instrument of connect with occasions and assessing occasions. This model of choking likewise compares to the union model of equipment translation from an RTL depiction. An articulation utilized as a part of a statement is constantly settled to a clock definition. The examined esteems are utilized to assess esteem change articulations that are required to decide a match of a grouping. The watchword property recognizes a simultaneous statement from a prompt declaration.

I. Functional coverage

Functional coverage is a measure of what functionalities/features of the design have been exercised by the tests. This can be useful in constrained random verification (CRV) to know what features have been covered by a set of tests in a regression. Function coverage is come from the RTL information and design spec information, it can reflect each test points and verification progress objectively, and the function coverage is born for measuring the test point verification degree.[11]

IV. Results

The DDR SDRAM memory controller designed and verified for the operations read, write, auto precharge, precharge, no operation, mode register, load-register1 and load-register2 by using DDR SDRAM micron specifications.

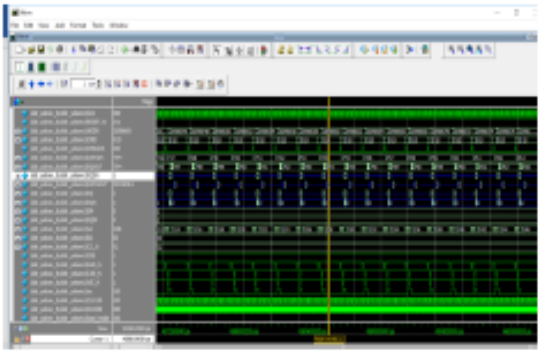


Figure 4: DDR SDRAM memory controller design wave form.

Once the reset signal is goes high, the module is iinitialized and run with 100MHz clock, then it's compulsory to provide the signal delay to high. Once the whole system gets initialized, Verilog design outputs are taken and observed as waveform shown in figure 4, Clearly observed that all the operations are executed with respect to commands given to DDR controller, So read and write operations are executed when command 001 and 010 are assigned respectively. DDR read and writes the data at the rate of twice means Double data rate per clock cycle is achieved. It's observed that the data written in particular memory address and reading the data from the same memory address are same.

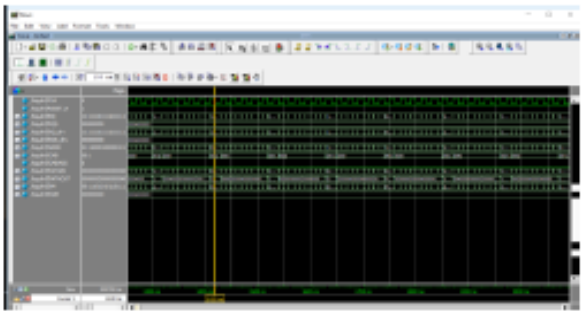


Figure 5: DDR SDRAM memory controller Verification wave form.

SystemVerilog code is used for verification of design to check for all the possible test case generations. So the main concern of designing DDR is to verify the read and write Operation, So figure 5 shows the output waveform, where read followed by write operation performed. The design works with all the testcase applied, that confirmed by writing coverage program by using cover groups and coverbins. As per DDR SDRAM Micron design specifications expected and actual data is matching after 45us delay. For verifying above statement in SystemVerilog, assertions and coverage's are used. By using coverage higher functional efficiency is gained, which is greater than Verilog verification and functional coverage is achieved 100% by using coverbins.

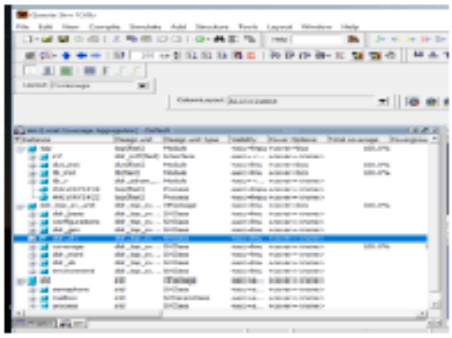


Figure 6: DDR SDRAM memory controller Functional coverage report.

Where the figure 6 shown the coverage report, In that coverage getting 100% by using random test cases in test bench and in coverage program , it cover bins to cover all the test cases that are necessary for the required operation and finally by running simulation 100% utilization of random test cases, that we applied to inputs namely ADDR and DATAIN.

V. CONCLUSION

Fully functional DDR SDRAM controller is designed and verified by using Verilog HDL and SystemVerilog HVL. Coverage is used to check all random test cases are used to covered or not. The DDR controller generates different types of control signals and timing signals, which are used to synchronize the operatios. The memory works at a rate twice the frequency of the processor, without changing or affecting the performance of the system and thereby, data bus size is reduced. Functional coverage is achieved 100% by using cover bins. The major drawback of DDR controller is the complex schematic with the use of large number of buffers. That increases the delay.

References

[1] Micron Technology Inc. Double Data Rate controller sdram data sheet.

[2] Hiroaki Ikeda and Hidemori Inukaiq,"High-Speed DRAM Architecture Development," IEEE Journal of Solid-State Circuits, Vol. 34, No. 5, May 1999.

[3] JEDEC, Solid state technology association -Double data rate SDRAM specification.

[4] DDR SDRAM Wikipedia.

[5] Synthesizable DDR SDRAM controller, Application note, 2002(Xilinx Inc., XAPP200).

[6] ALTERA, DDR SDRAM Controller White Paper, Ver1.1, 2002, 8.

[7] John Wakerly "Digital systems design – 8 th Edition.".

[8] J. Bhasker "A VHDL Primer", 3 rd Edition ,Pearson Education.

[9] Anders Nordstrom, —Sub-cycle Functional Timing Verification Using SystemVerilog Assertions, Verilab Inc.

[10] CHRIS SPEAR, —System Verilog for verification,‖ Synopsys, Inc.

[11] Harry Foster, Adam Krolnik, David Lacey, "Assertion Based Design, 2nd Edition," Springer,

[12] Using SystemVerilog Assertions in Gate-Level Verification Environments, Mark Litterick, DVCon 2006.

APPENDIX -A

512Mb DDR SDRAM SPECIFICATION DATASHEET

SPECTEK®
Brilliant Memory Solutions

512Mb: x4, x8, x16
DDR SDRAM

DOUBLE DATA RATE (DDR) SDRAM

FEATURES

- ROHS Compliant
- PC2100, PC2700 and PC3200 compatible
- VDD = +2.5V ±0.2V, VDDQ = +2.5V ±0.2V (For –6A & -75A)
- VDD = +2.6V ±0.1V, VDDQ = +2.6V ±0.1V (For -5B)
- Bi-directional data strobe (DQS) transmitted/ received with data, i.e. source-synchronous data capture (x16 has two: LDQS and UDQS – one per byte)
- Internal, pipelined double-data-rate (DDR) architecture; two data accesses per clock cycle
- Differential clock inputs (CK and CK#)
- Commands entered on each positive CK edge
- DQS edge-aligned with data for READs; center-aligned with data for WRITEs
- DLL to align DQ and DQS transitions with CK
- Four internal banks for concurrent operation
- Data mask (DM) for masking write data (x16 has two: LDM and UDM – one per byte)
- Programmable burst lengths: 2, 4, or 8
- Auto precharge option
- Auto Refresh
- Longer lead TSOP for improved reliability (OCPL)
- 2.5V I/O (SSTL_2 compatible)
- These devices are optimized for single rank DIMM applications

Options:	Designation:
Family:	
SpecTek Memory	SGG[1] SMG[2]
Configuration:	
128 Meg x 4 (32 Meg x 4 x 4 banks)	128M4
64 Meg x 8 (16 Meg x 8 x 4 banks)	64M8
32 Meg x 16 (8 Meg x 16 x 4 banks)	32M16
Design ID:	
DDR2 512 Megabit Design (Call SpecTek Sales for details on availability of "x" placeholders)	Tx7x[3]
Voltage and refresh:	
2.5V, Auto Refresh	V8
2.5V, Self or Auto Refresh	R8
Package Types (Lead-Free):	
66-pin Plastic TSOP, OCPL (400 mil width, 0.65mm pin pitch)	TLF
60-ball (10mm x 12.5mm) FBGA	FNF
Package Types (Leaded):	
66-pin Plastic TSOP, OCPL (400 mil width, 0.65mm pin pitch)	TLL
60-ball (10mm x 12.5mm) FBGA	FNL

Timing – Cycle Time:

5ns @ CL=3 (PC3200 or DDR400B)	-5B
6ns @ CL = 2.5 (PC2700 or DDR333)	-6A
7.5ns @ CL = 2.5 (PC2100 or DDR266)	-75A

Part number example: SGG64M8T27ZV8TLF-5B

(For part numbers prior to December 2004, refer to page 12 for decoding, or for part numbers prior to July 7th 2006, refer to page 13 for decoding.)

Note: 1 "SGG" is a SpecTek original device.
2 "SMG" is a SpecTek logo-blasted device.
3 Contact SpecTek Sales for details on availability of the "X" placeholders.

PIN ASSIGNMENT (TOP VIEW) 66-PIN TSOP

x4	x8	x16			x16	x8	x4
VDD	VDD	VDD	1	66	VSS	VSS	VSS
NF	DQ0	DQ0	2	65	DQ15	DQ7	NF
VDDQ	VDDQ	VDDQ	3	64	VSSQ	VSSQ	VSSQ
NC	NC	DQ1	4	63	DQ14	NC	NC
DQ0	DQ1	DQ2	5	62	DQ13	DQ6	DQ3
VSSQ	VSSQ	VSSQ	6	61	VDDQ	VDDQ	VDDQ
NC	NC	DQ3	7	60	DQ12	NC	NC
NF	DQ2	DQ4	8	59	DQ11	DQ5	NF
VDDQ	VDDQ	VDDQ	9	58	VSSQ	VSSQ	VSSQ
NC	NC	DQ5	10	57	DQ10	NC	NC
DQ1	DQ3	DQ6	11	56	DQ9	DQ4	DQ2
VSSQ	VSSQ	VSSQ	12	55	VDDQ	VDDQ	VDDQ
NC	NC	DQ7	13	54	DQ8	NC	NC
NC	NC	NC	14	53	NC	NC	NC
VDDQ	VDDQ	VDDQ	15	52	VSSQ	VSSQ	VSSQ
NC	NC	LDQS	16	51	UDQS	DQS	DQS
NC	NC	NC	17	50	DNU	DNU	DNU
VDD	VDD	VDD	18	49	VREF	VREF	VREF
DNU	DNU	DNU	19	48	VSS	VSS	VSS
NC	NC	LDM	20	47	UDM	DM	DM
WE#	WE#	WE#	21	46	CK#	CK#	CK#
CAS#	CAS#	CAS#	22	45	CK	CK	CK
RAS#	RAS#	RAS#	23	44	CKE	CKE	CKE
CS#	CS#	CS#	24	43	NC	NC	NC
NC	NC	NC	25	42	A12	A12	A12
BA0	BA0	BA0	26	41	A11	A11	A11
BA1	BA1	BA1	27	40	A9	A9	A9
A10/AP	A10/AP	A10/AP	28	39	A8	A8	A8
A0	A0	A0	29	38	A7	A7	A7
A1	A1	A1	30	37	A6	A6	A6
A2	A2	A2	31	36	A5	A5	A5
A3	A3	A3	32	35	A4	A4	A4
VDD	VDD	VDD	33	34	VSS	VSS	VSS

	128 Meg x 4	64 Meg x 8	32 Meg x 16
Configuration	32 Meg x 4 x 4 banks	16 Meg x 8 x 4 banks	8 Meg x 16 x 4 banks
Refresh Count	8K	8K	8K
Row Addressing	8K (A0–A12)	8K (A0–A12)	8K (A0–A12)
Bank Addressing	4 (BA0, BA1)	4 (BA0, BA1)	4 (BA0, BA1)
Column Addressing	4K (A0–A9, A11, A12)	2K (A0–A9, A11)	1K (A0–A9)

GENERAL DESCRIPTION

The 512Mb DDR SDRAM is a high-speed CMOS, dynamic random-access memory containing 536,870,912 bits. It is internally configured as a quad-bank DRAM.

The 512Mb DDR SDRAM uses a double-data rate architecture to achieve high-speed operation. The double data rate architecture is essentially a 2*n*-prefetch architecture with an interface designed to transfer two data words per clock cycle at the I/O pins. A single read or write access for the 512Mb DDR SDRAM effectively consists of a single 2*n*-bit wide, one-clock-cycle data transfer at the internal DRAM core and two corresponding *n*-bit wide, one-half-clock-cycle data transfers at the I/O pins.

A bi-directional data strobe (DQS or LDQS/UDQS) is transmitted externally, along with data, for use in data capture at the receiver. DQS is a strobe transmitted by the DDR 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 x16 offering has two data strobes, one for the lower byte and one for the upper byte.

The 512Mb DDR SDRAM operates from a differential clock (CK and CK#); the crossing of CK going HIGH and CK# 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 DDR 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 registration of an ACTIVE command, which is then followed by a READ or WRITE command. The address bits registered coincident with the ACTIVE command are used to select the bank and row to be accessed. The address bits registered coincident with the READ or WRITE command are used to select the bank and the starting column location for the burst access.

The DDR SDRAM provides for programmable READ or WRITE burst lengths of 2, 4, or 8 locations. An auto precharge function may be enabled to provide a self-timed row precharge that is initiated at the end of the burst access.

As with standard SDR SDRAMs, the pipelined, multibank architecture of DDR SDRAMs allows for concurrent operation, thereby providing high effective bandwidth by hiding row precharge and activation time.

An auto refresh mode is provided, along with a power-saving power-down mode. All inputs are compatible with the JEDEC Standard for SSTL_2. All full drive strength outputs are SSTL_2, Class II compatible.

NOTE 1: The functionality and the timing specifications discussed in this data sheet are for the DLL-enabled mode of operation.

NOTE 2: Throughout the data sheet, the various figures and text refer to DQs as "DQ." The DQ term is to be interpreted as any and all DQ collectively, unless specifically stated otherwise.
Additionally, the x16 is divided in to two bytes — the lower byte and upper byte. For the lower byte (DQ0 through DQ7) DM refers to LDM and DQS refers to LDQS; and for the upper byte (DQ8 through DQ15) DM refers to UDM and DQS refers to UDQS.

ABSOLUTE MAXIMUM RATINGS*

(Voltages Relative to VSS)

V_{DD} Supply	-1V to +3.6V
$V_{DD}Q$ Supply	-1V to +3.6V
V_{REF} and Inputs	-1V to +3.6V
I/O Pins	-0.5V to $V_{DD}Q$ +0.5V
Operating Temperature, TA (ambient)	10°C to +70°C
Storage Temperature (plastic)	-55°C to +150°C
Power Dissipation	1W
Short Circuit Output Current	50mA

APPENDIX B

ModelSim and Questasim

What are Projects?

Projects are collection entities for designs under specification or test. At a minimum, projects have a root directory, a work library, and "metadata" which are stored in an .mpf file located in a project's root directory. The metadata include compiler switch settings, compile order, and file mappings. Projects may also include:

- Source files or references to source files
- other files such as READMEs or other project documentation
- local libraries
- references to global libraries
- Simulation Configurations (see Creating a Simulation Configuration)
- Folders (see Organizing Projects with Folders)

Project metadata are updated and stored only for actions taken within the project itself. For example, if you have a file in a project, and you compile that file from the command line rather than using the project menu commands, the project will not update to reflect any new compile settings.

What are the Benefits of Projects?

Projects offer benefits to both new and advanced users. Projects

- simplify interaction with Questa SIM; you don't need to understand the intricacies of compiler switches and library mappings
- eliminate the need to remember a conceptual model of the design; the compile order is maintained for you in the project. Compile order is maintained for HDL-only designs.
- remove the necessity to re-establish compiler switches and settings at each session; these are stored in the project metadata as are mappings to source files
- allow users to share libraries without copying files to a local directory; you can establish references to source files that are stored remotely or locally.
- allow you to change individual parameters across multiple files; in previous versions you could only set parameters one file at a time
- enable "what-if" analysis; you can copy a project, manipulate the settings, and rerun it to observe the new results
- reload the initial settings from the project .mpf file every time the project is opened

Project Conversion Between Versions

Projects are generally not backwards compatible for either number or letter releases. When you open a project created in an earlier version, you will see a message warning that the project will be converted to the newer version. You have the option of continuing with the conversion or cancelling the operation.

As stated in the warning message, a backup of the original project is created before the conversion occurs. The backup file is named <project name>.mpf.bak and is created in the same directory in which the original project is located.

Getting Started with Projects

This section describes the four basic steps to working with a project.

- Step 1 — Creating a New Project

  This creates an .mpf file and a working library.

- Step 2 — Adding Items to the Project

  Projects can reference or include source files, folders for organization, simulations, and any other files you want to associate with the project. You can copy files into the project directory or simply create mappings to files in other locations.

- Step 3 — Compiling the Files

  This checks syntax and semantics and creates the pseudo machine code Questa SIM uses for simulation.

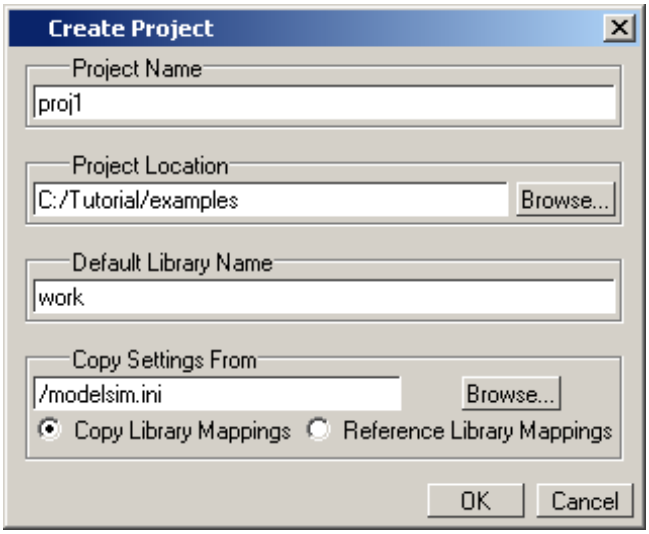
- Step 4 — Simulating a Design

  This specifies the design unit you want to simulate and opens a structure tab in the Workspace pane.

Step 1 — Creating a New Project

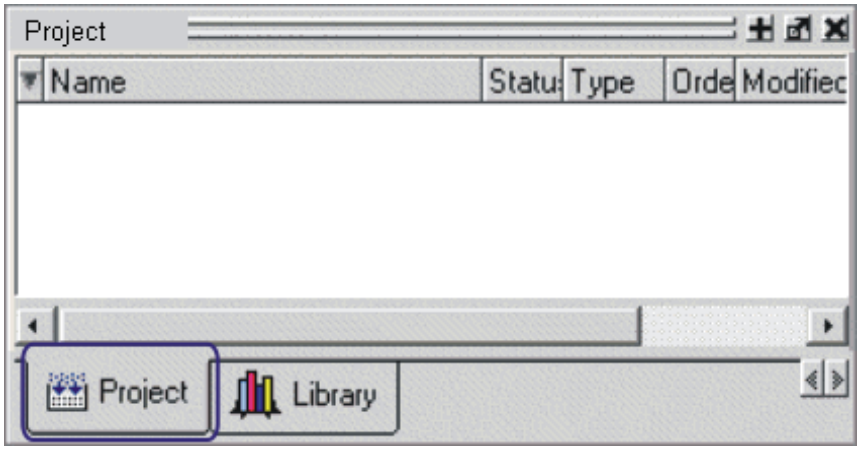
Select **File > New > Project** to create a new project. This opens the **Create Project** dialog where you can specify a project name, location, and default library name. You can generally leave the **Default Library Name** set to "work." The name you specify will be used to create a working library subdirectory within the Project Location. This dialog also allows you to reference library settings from a selected .ini file or copy them directly into the project.

Figure 5-1. Create Project Dialog



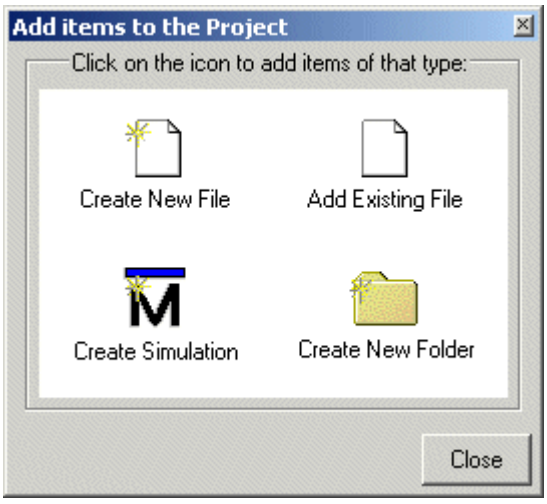
After selecting OK, you will see a blank Project window in the Main window (Figure 5-2)

Figure 5-2. Project Window Detail



and the **Add Items to the Project** dialog (Figure 5-3).

Figure 5-3. Add items to the Project Dialog



The name of the current project is shown at the bottom left corner of the Main window.

Step 2 — Adding Items to the Project

The **Add Items to the Project** dialog includes these options:

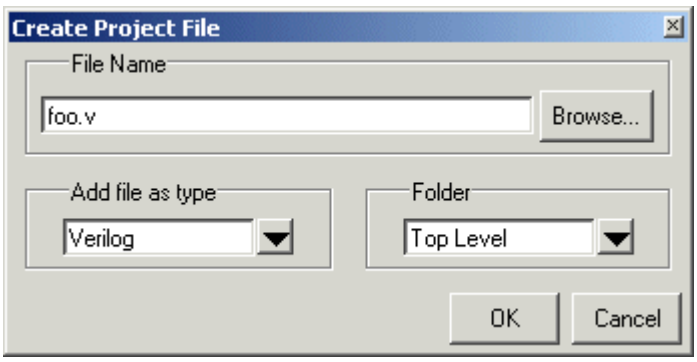
- **Create New File** — Create a new VHDL, Verilog, SystemC, Tcl, or text file using the Source editor. See below for details.
- **Add Existing File** — Add an existing file. See below for details.
- **Create Simulation** — Create a Simulation Configuration that specifies source files and simulator options. See Creating a Simulation Configuration for details.
- **Create New Folder** — Create an organization folder. See Organizing Projects with Folders for details.

Create New File

The **File > New > Source** menu selections allow you to create a new VHDL, Verilog, SystemC, Tcl, or text file using the Source editor.

You can also create a new project file by selecting **Project > Add to Project > New File** (the Project tab in the Workspace must be active) or right-clicking in the Project tab and selecting **Add to Project > New File**. This will open the Create Project File dialog (Figure 5-4).

Figure 5-4. Create Project File Dialog



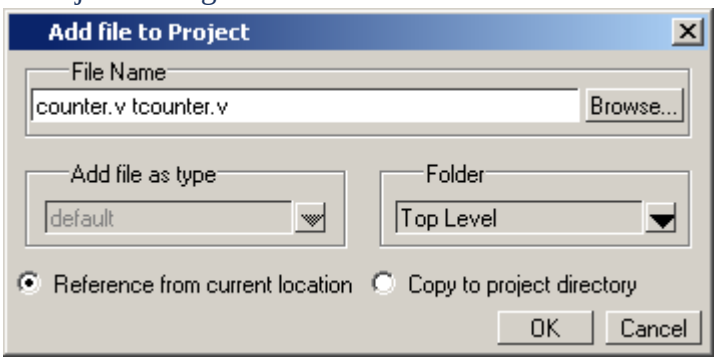
Specify a name, file type, and folder location for the new file.

When you select OK, the file is listed in the Project tab. Double-click the name of the new file and a Source editor window will open, allowing you to create source code.

Add Existing File

You can add an existing file to the project by selecting **Project > Add to Project > Existing File** or by right-clicking in the Project tab and selecting **Add to Project > Existing File**.

Figure 5-5. Add file to Project Dialog

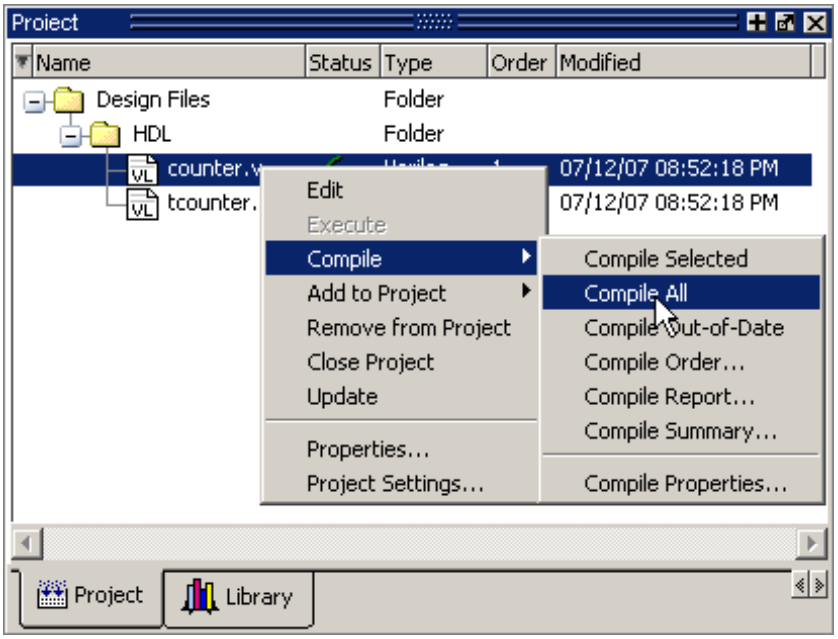


When you select OK, the file(s) is added to the Project tab.

Step 3 — Compiling the Files

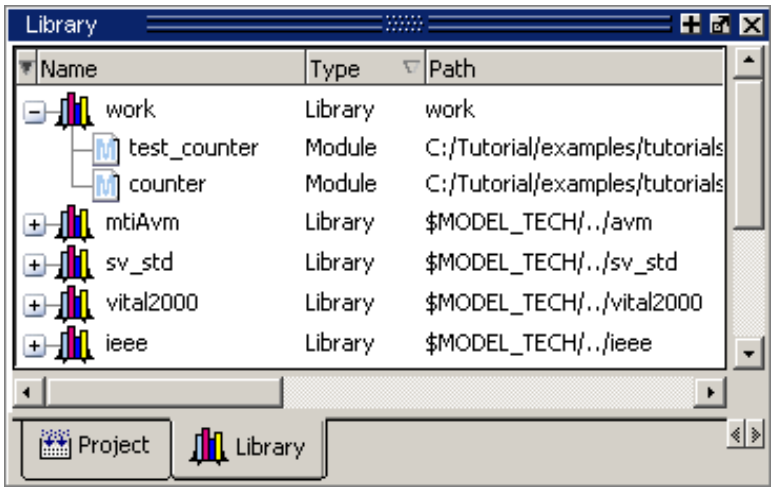
The question marks in the Status column in the Project tab denote either the files haven't been compiled into the project or the source has changed since the last compile. To compile the files, select **Compile > Compile All** or right click in the Project tab and select **Compile > Compile All** (Figure 5-6).

Figure 5-6. Right-click Compile Menu in Project Window



Once compilation is finished, click the Library window, expand library work by clicking the "+", and you will see the compiled design units.

Figure 5-7. Click Plus Sign to Show Design Hierarchy



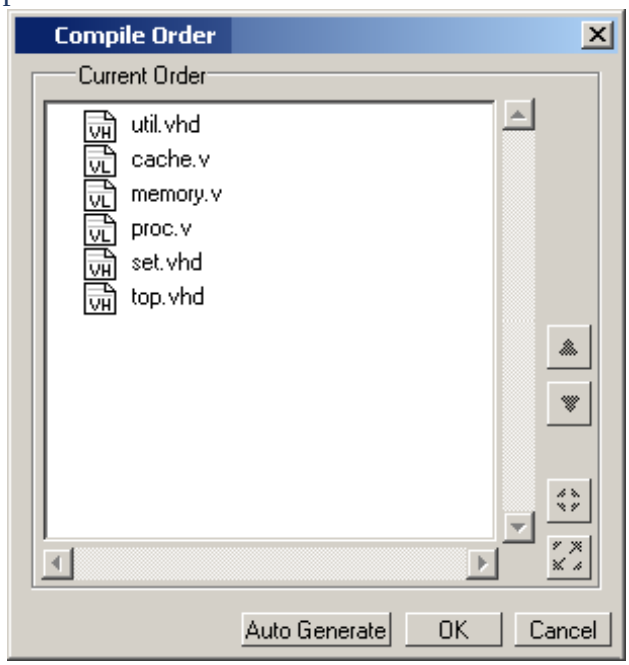
Changing Compile Order

The Compile Order dialog box is functional for HDL-only designs. When you compile all files in a project, Questa SIM by default compiles the files in the order in which they were added to the project. You have two alternatives for changing the default compile order: 1) select and compile each file individually; 2) specify a custom compile order.

To specify a custom compile order, follow these steps:

1. Select **Compile > Compile Order** or select it from the context menu in the Project tab.

Figure 5-8. Setting Compile Order



2. Drag the files into the correct order or use the up and down arrow buttons. Note that you can select multiple files and drag them simultaneously.

Auto-Generating Compile Order

Auto Generate is supported for HDL-only designs. The **Auto Generate** button in the Compile Order dialog (see above) "determines" the correct compile order by making multiple passes over the files. It starts compiling from the top; if a file fails to compile due to dependencies, it moves that file to the bottom and then recompiles it after compiling the rest of the files. It continues in this manner until all files compile successfully or until a file(s) can't be compiled for reasons other than dependency.

Files can be displayed in the Project window in alphabetical or compile order (by clicking the column headings). Keep in mind that the order you see in the Project tab is not necessarily the order in which the files will be compiled.

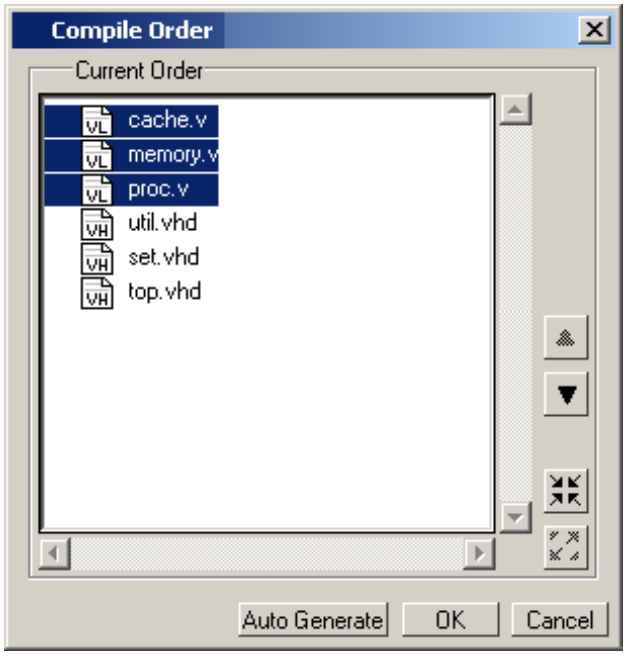
Grouping Files

You can group two or more files in the Compile Order dialog so they are sent to the compiler at the same time. For example, you might have one file with a bunch of Verilog define statements and a second file that is a Verilog module. You would want to compile these two files together.

To group files, follow these steps:

1. Select the files you want to group.

Figure 5-9. Grouping Files



2. Click the Group button. 

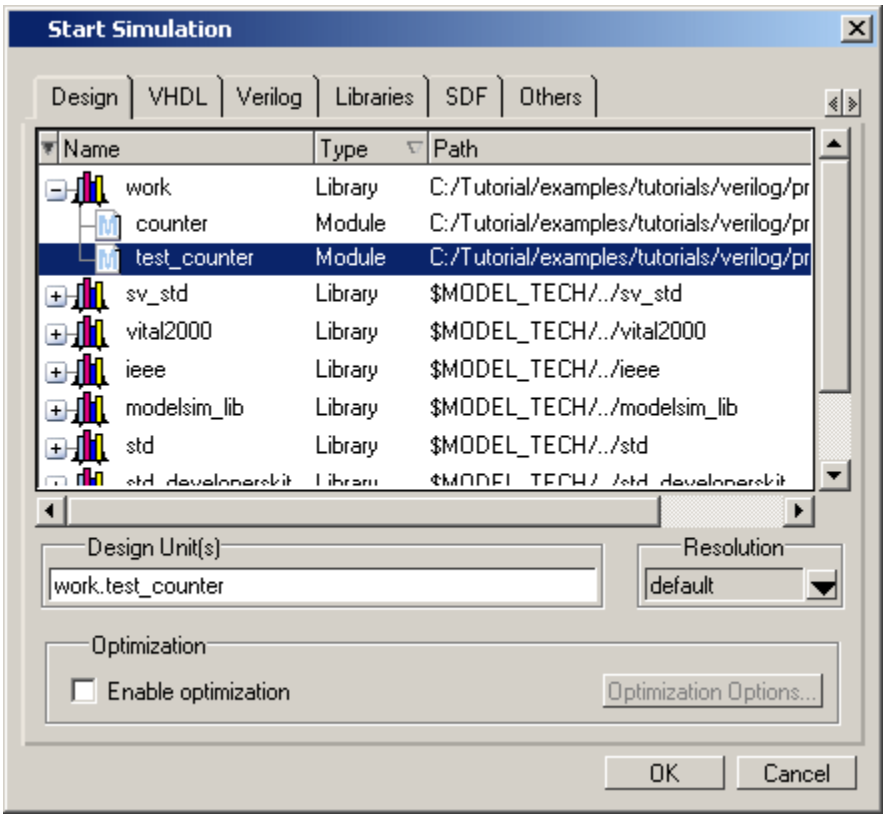
   To ungroup files, select the group and click the Ungroup button. 

Step 4 — Simulating a Design

To simulate a design, do one of the following:

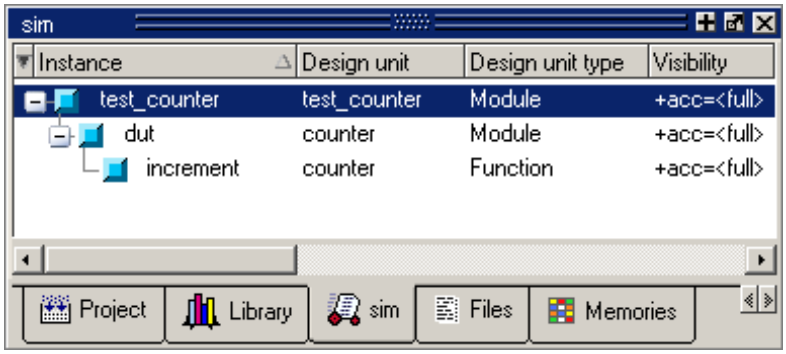
- double-click the Name of an appropriate design object (such as a test bench module or entity) in the Library window
- right-click the Name of an appropriate design object and select **Simulate** from the popup menu
- select **Simulate > Start Simulation** from the menus to open the Start Simulation dialog (Figure 5-10). Select a design unit in the Design tab. Set other options in the VHDL, Verilog, Libraries, SDF, and Others tabs. Then click OK to start the simulation.

Figure 5-10. Start Simulation Dialog



A new Structure window, named sim, appears that shows the structure of the active simulation (Figure 5-11).

Figure 5-11. Structure WIndow with Projects



At this point you are ready to run the simulation and analyze your results. You often do this by adding signals to the Wave window and running the simulation for a given period of time. See the Questa SIM Tutorial for examples.

APPENDIX C

Plagiarism Rrport

1BG17LVS05

ORIGINALITY REPORT

10%	8%	3%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	download.micron.com Internet Source	3%
2	Submitted to CSU, San Jose State University Student Paper	1%
3	ineltec.de Internet Source	1%
4	Submitted to Upper Iowa University Student Paper	1%
5	Submitted to Visvesvaraya Technological University Student Paper	<1%
6	www.altera.co.jp Internet Source	<1%
7	www.hkmjd.com Internet Source	<1%
8	www.piecemakers.com.tw Internet Source	<1%
9	Bakshi, A., S. S. Pandey, T. Pradhan, and R.	

	Dey. "ASIC implementation of DDR SDRAM Memory Controller", 2013 IEEE International Conference ON Emerging Trends in Computing Communication and Nanotechnology (ICECCN), 2013. Publication	<1%
10	www.geeksforgeeks.org Internet Source	<1%
11	www.freepatentsonline.com Internet Source	<1%
12	www.nanya.com Internet Source	<1%
13	www.issi.com Internet Source	<1%
14	www.smartmodular.com Internet Source	<1%
15	www.ejournal.aessangli.in Internet Source	<1%
16	www.promos.com.tw Internet Source	<1%
17	www.ijcsit.com Internet Source	<1%
18	Submitted to Higher Education Commission Pakistan Student Paper	<1%

19	www.opengpu.org Internet Source	<1%
20	Submitted to MVJ College of Engineering, Bangalore Student Paper	<1%
21	www.micross.com Internet Source	<1%
22	ijates.com Internet Source	<1%
23	docplayer.net Internet Source	<1%
24	Submitted to VIT University Student Paper	<1%
25	Michael Smith, Dongcheng Deng, Syed Islam, James Miller. "Enhancing Hardware Assisted Test Insertion Capabilities on Embedded Processors using an FPGA-based Agile Test Support Co-processor", Journal of Signal Processing Systems, 2013 Publication	<1%
26	www.silicon-power.com Internet Source	<1%
27	www.hyclassproject.com Internet Source	<1%
	www.spectek.com	

28	Internet Source	<1%
29	Submitted to De Montfort University Student Paper	<1%
30	ptucse.loremate.com Internet Source	<1%
31	Submitted to University of Westminster Student Paper	<1%
32	www.ihanker.com Internet Source	<1%
33	www.webpriceguide.co.uk Internet Source	<1%
34	www.ijetcse.com Internet Source	<1%
35	Power Aware Design Methodologies, 2002. Publication	<1%
36	cache.freescale.com Internet Source	<1%
37	www.slidegur.com Internet Source	<1%
38	cs.utsource.net Internet Source	<1%
39	www.jedec.org Internet Source	<1%

40	Submitted to Middlesex University Student Paper	<1%
41	pdfs.icsupply.com Internet Source	<1%
42	www.winbond.co.jp Internet Source	<1%
43	Submitted to CTI Education Group Student Paper	<1%
44	"Professional Verification", Springer Nature, 2004 Publication	<1%
45	primrosebank.net Internet Source	<1%
46	attila.kinali.ch Internet Source	<1%
47	marsohod.org Internet Source	<1%
48	www.mmx.co.il Internet Source	<1%
49	Chu, . "SRAM and SDRAM Controllers", Embedded SoPC Design with Nios II Processor and Verilog Examples Chu/Embedded, 2012. Publication	<1%