A VHDL SGRAM Model for the Validation Environment of a High Performance Graphic Processor

Michael G. Wahl
Universität Siegen
Siegen, Germany

Holger Völkel
SP3D Chipdesign
Starnberg, Germany

Abstract

To validate the functionality of a new highly complex graphics processor described in VHDL the working environment of the processors has to be modelled. In some cases appropriate models for the external components are commerically available, in other cases these models have to be created. In this paper a general memory model for SGRAMs is presented which had to be implemented to have a flexible simulation environment for a high speed graphics processor at hand. Key features are the generality, the support of SGRAM arrays of various shapes and functions supporting the simulation process. This functionality goes far beyond the capabilities of currently commercially available SGRAM models.

1 Introduction

Current systems are no longer described at gate level but using a hardwrae description language like VHDL or Verilog. In parallel to the system itself an appropriate description of the environment must be at hand to be able to simulate the system. The testbench (Figure 1) consists of the interconnections of the surrounding components as well as their timing and behaviour. The VHDL model described below covers significantly more than the pure functionality of a single SGRAM. First, parameters of various SGRAMs have been extracted to get a model which covers the worst case of all considered SGRAMs to become independent of a single vendor as well as the size of the currently offered SGRAMs. Second, the memory of a graphics processor usually consists of more than a single SGRAM which leads to the requirement that arrays of SGRAMs have to be supported. Finally, before simulation the memory array has to be initialized and after the simulation run the contents of the memory has to be dumped for further analysis.

2 SDRAMS and SGRAMS

Contrary to the standard DRAMs, the SDRAMS and SGRAMS are synchronuous memories. This means that the function of the memory is programmed and while one operation is in progress the control bus can be used to program the second memory bank. This allows the creation of a continuous data stream to and from the memory. The behaviour of an SDRAM is described by a state machine. This state machine is quite complex because most of the operations can be interrupted and any other operation may follow. In addition, most of the state transitions depend on the clock edge while other transitions happen asynchronously. The SGRAMS cover all functions of the SDRAMS. In addition they allow operations which usually occur in a graphic environment like filling a set of memory locations with the same value (colour) or changing only specific bits which are masked by the mask register. This helps to avoide time consuming read-modify-write cycles.

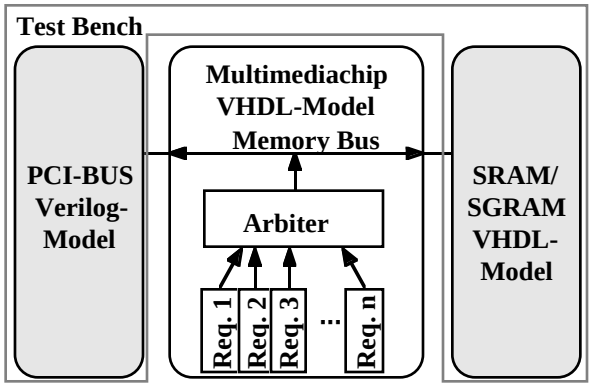

Figure 1: Testbench for a multimedia chip

3 The VHDL model

The VHDL model consists of three major parts. The SGRAM core model describes the behaviour of a single SGRAM chip. The system model allows to describe a memory module consisting of one or more memory chips, which can be configured in arbitrary ways, whereas the debug module allows the writing and dumping of the memory without the necessity to simulate the complete model. In addition, all commands issued to the SGRAM and all intended state transitions are checked if they are valid or not.

3.1 The SGRAM core model

The SGRAM core is the largest part of the model. The challenge to this model was not only to describe one specific SGRAM but provide a scaleable general model for SGRAMs. They are available in various

sizes currently in the range from 8 to 16 Mbit. The width of the data bus varies from 4 to 32 Bits. The size of the used address bus depends on the memory size and the width of the data bus.

The SGRAMs are available in two different architecture variants. The architecture can support either pipelined operation or prefetch operation. In pipelined mode a new column address can be sent to the memory with each cycle which is a feature often used for graphics applications. The prefetch architecture allows the loading of a new column address only every second clock cycle, but the operating frequency of the memory is higher.

The interface of course supports the normal I/O according to the SGRAM specification (address bus, data bus, control bus with clk, cke, cas, ras, cs, we and dsf, and the dqm bits). Internally there is an interface to the debug module and to the file I/O for the load/store function. The latter is done using fixed file names, assuming that the file names are modified using a shell script.

3.2 Timing parameters

The timing parameters of the memory have been taken from [4] as well as from the data sheets of the individual manufacturers [2][3][5]. The parameters used in the model as default values are the worst case specifications. They are described using generics and hence can easily be updated.

The implementation of the state machine is straightforward. There are only a few things to consider: First, the SGRAM has two banks, so there has to be made a decision if the command is issued for bank a or bank b. Second, both banks can have different states, so the state calculation has to be done for both banks at each clock. Finally, the state of one bank can affect the state of the other bank.

For the outside world the memory is synchronous, but internally it works asynchronously. This becomes visible at the point where the read latency has to be programmed because the SGRAM itself does not know about its operation speed and hence cannot determine by itself in which clock cycle the data will be available. For the model another point is of more importance: The states, respectively the transitions in the states precharge, row_active and write_rec, depend not only on the clock but on timing constraints, too. So the model has to check if a command is issued in time or too early. To simulate this behaviour "virtual states" had to be introduced which behave like the other states but generate an error message if the timing constraints are violated.

3.3 The System Model

To be able to describe memory systems consisting of two or more SGRAMs a method is implemented allowing the specification of arrays of memories. The total size of the memory system is currently limited by the default width of the address bus of 32 bits. Because this is a generic, an extension of the capactiy can easily be done. An example of a system consisting of two memories is given in Figure 2.

3.4 The Debug Model

One of the main requirements for the SGRAM model was the support of the initialization of the memory before the start of the simulation and the ability to dump the memory at the end of the simulation. The implementation of this option helped to reduce the simulation time, because initialization and dumping can now be done in one simulation tick. In addition, the memory can be initialized so that it is possible to trace back if a memory cell has been accessed or not.

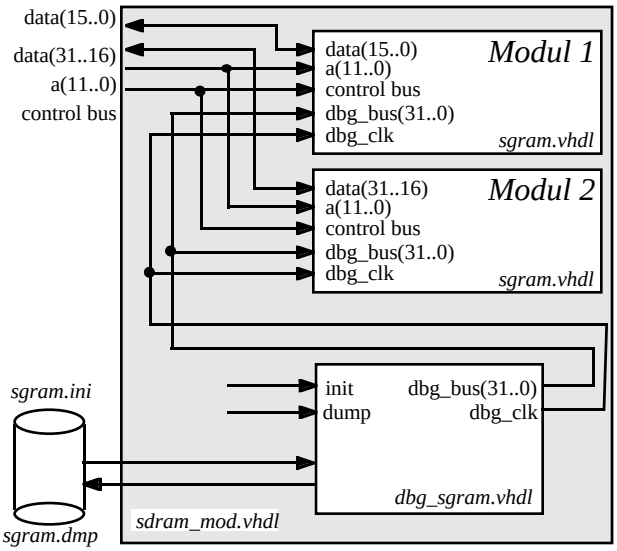

Figure 2: An example configuration

4 Summary

The general SGRAM module model described above covers all aspects which are required for the simulation of a graphic processor. The functionality of the model goes far beyond what a single SGRAM does. It is vendor independent and scaleable and it supports debug functions reducing the simulation time. It was successfully used in the design of a multimedia chip which is now in production.

References

[1] Y.C. Hsu, K. F. Tsai,J. T. Liu, E. S. Lin, *VHDL Modelling for Digital Synthesis*. Kluver Academic Publishers, Dordrecht, 1995

[2] *Hitachi Synchronous DRAM Application Notes*. Hitachi Semiconductor and IC Division, 1995

[3] *IBM Datasheets for 0316409C, 03116809C, SA 14-4711-01, Revised 01/96*. IBM Corporation, 1996

[4] Micron Technical Notes *SDRAM / SGRAM: Design for Compatibility, Part 1-3*. Micron Technology Inc., 1995

[5] *NEC Datasheets for uPD4516412, uPD4516821, uPD4516161, ID-3394A, Published 10/94*. NEC Corporation, 1994