AN OCM BASED SHARED MEMORY CONTROLLER FOR VIRTEX 4

Bas Breijer, Filipa Duarte, and Stephan Wong

Computer Engineering, EEMCS
Delft University of Technology
Mekelweg 4, 2826CD, Delft, The Netherlands
email: S.D.Breijer@student.tudelft.nl, F.Duarte@ce.et.tudelft.nl, J.S.S.M.Wong@ewi.tudelft.nl

ABSTRACT

In this paper, we present a shared instruction and data memory controller for the On-Chip Memory (OCM) bus of the PowerPC embedded in the Virtex-4 chip. The traditional design of the memory controller is implemented on the Processor Local Bus (PLB), which is an undeterministic bus. Our design utilizes the OCM bus which is a dedicated memory bus. As such, the use of the OCM bus allows larger off-chip memories (compared to internal BRAMs) to be connected without the overhead of PLB arbitration. Moreover, our design is advantageous because its performance does not depend on the number of connected peripherals on the bus but merely on the delay of the connected DDR memory. Results indicate that, using the shared instruction and data memory controller on the OCM bus, introduces an improvement from 6% to 13% over the PLB based design.

1. INTRODUCTION

In embedded systems, memory can be located on the same physical chip as the processor (on-chip) to increase performance. To improve flexibility, designers can opt to attach off-chip memory modules that can vary in size and speed. Memory situated on the same chip as the processor often has a small capacity because of limited available chip area. These kind of memories may be sufficient for small applications, but are rarely large enough to support large applications like, for instance, an operating system. Therefore, it is important to provide a fast and reliable connection to off-chip memories.

In this paper, we present an off-chip DDR memory controller connected to the PPC using the On-Chip Memory (OCM) buses. The controller shares one physical memory interface to store both data and instructions. Our design has the advantage of having an access time to the memory that is independent of the bus but merely on the memory latency, as it will be explained in Section 3. Compared to a system based on a Processor Local Bus (PLB) DDR controller, a system based on the shared OCM DDR controller presented, introduces a performance improvement from 6% to 13%.

This paper is organized as follows: Section 2 discusses related work and the main differences between our solution and the solutions proposed by other authors. Section 3 motivates and explains the design, Section 4 discusses the implementation and verification of the design targetting the Virtex-4. In Section 5 we present the results obtained comparing our design to a PLB based reference design. Finally, in Section 6 we present our conclusions.

2. RELATED WORK

In this section, we first discuss techniques for performance improvements presented by different authors and, second, we address work related to arbitration in general and specifically for the Intellectual Property Interface (IPIF).

Improving memory performance is a well-discussed issue. Improvements are achieved by use of scheduling [1], prefetching [2, 3], or pipelining if separate banks are addressed [4]. In addition, caches are generally utilized to improve performance [5, 6]. The use of the Data-Side OCM (DSOCM) bus to connect off-chip DDR memory is only considered in [7]. No work was found on the utilization of the Instruction-Side OCM (ISOCM) bus to connect off-chip DDR memory.

The shared character of our design implies that the presented controller must arbitrate requests for instructions and requests for data in a fair manner. Memory arbitration is an important issue in multiprocessor systems. In [8], a multiprocessor system using IPIF arbitration is implemented. In [9], three important metrics for memory arbitration in general are introduced. Memory arbitration must be fair, a memory arbiter must have a low overhead and a memory arbiter must be easy to insert.

Our design introduces a mechanism to connect memories with a latency larger than one clock cycle to the ISOCM bus. Moreover, we introduce a controller which combines both ISOCM bus and DSOCM bus to connect the same physical memory and our controller allows it to be addressed as if the memory is part of a Von-Neumann architecture. Furthermore, our design still allows techniques like scheduling, prefetching, pipelining, and caching to be applied to improve performance even more. In comparison to [7], we present the performance of system utilizing a combination of instruction read, data write and data read operations on implementation level, while [7] considers the post-synthesis simulation results of exclusive data read or write operations.

3. MOTIVATION & OVERALL DESIGN

The PowerPC (PPC) embedded in the Virtex-4, utilizes two main buses to connect peripherals and memory (both on-chip and off-chip): the OCM and the PLB. In [10], a theoretical comparison is made between the PLB and the OCM bus using simulation results. The comparison focuses on the use of BRAM blocks connected to the OCM bus and to the PLB. As the amount and type of memory connected to both the PLB and OCM buses is the same, the comparison only reflects the bus performance. The author shows that especially large applications benefit from using the OCM by offloading traffic from the PLB. The system used in [10] is based on the Virtex II Pro, but since the Virtex-4 is conceptually the same, the comparison should hold for Virtex-4 based systems. The difference in performance between the PLB and the OCM bus is mainly due to the different bus characteristics. We will give a short summary of the most important differences.

The PLB is an undeterministic bus, meaning the latency of an operation varies depending on the number and performance of the connected devices due to the hand-shaking protocol implemented on the bus. The PLB bus is a shared bus, a request on the bus may have to wait for other slaves to complete their operations. The PLB uses a 64-bit data bus and the addressable range of the PLB is 4 Gigabytes. The operating frequency of the PLB depends on the arbiter.

The OCM bus is split into two separate parts, the ISOCM bus and the DSOCM bus. Both ISOCM and DSOCM are designed to interface with the on-chip BRAMs with a fixed latency. The OCM bus is a deterministic bus, meaning that the latency of the bus only depends on the latency of the connected device. The DSOCM is a 32-bit wide data bus, the ISOCM is a 64-bit wide data bus and the addressable range is 16 Megabyte for each bus. The operating frequency of the OCM bus can not operate on a higher frequency than the PPC. Since, the OCM bus is a dedicated bus, the bus does not need to wait for other devices to finish their operation.

Our design utilizes both OCM buses to connect external DDR memory. The controller introduces a shared address space for instructions and data. This controller consist of four basic blocks: the DSOCM IPIF controller, the ISOCM IPIF controller, the IPIF Arbiter, and the IPIF DDR controller. The blocks are internally connected using the IPIF bus. Figure 1 depicts the internal controller organization.

ISOCM IPIF: The ISOCM bus is designed to interface

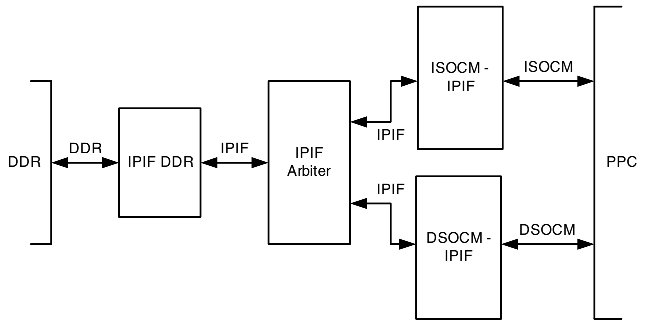
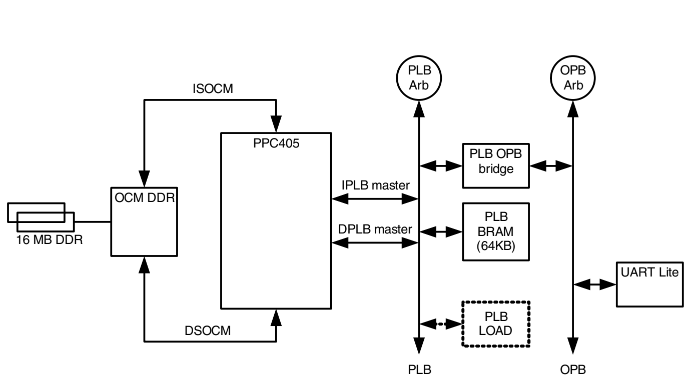


Fig. 1. Internal OCM DDR controller organization.

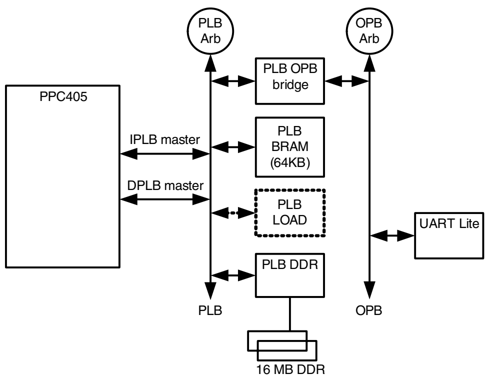
with on-chip BRAM blocks that provide data one clock cycle after a request is sent. Unfortunately, DDR memory has a latency that is greater than one clock cycle. Therefore, we introduce a mechanism based on the scheme used in the Molen [11] μ-instruction arbiter [12]. The scheme used by [12] provides the PPC repeatedly with the *bl* instruction causing the PPC to store the next instruction address in the link register (LR) and branch to the current instruction. This scheme will cause the PPC to loop infinitively. Once the PPC is allowed to continue, the *blr* is provided to the PPC to branch on the LR and continue normal execution. In our design, the PPC needs to loop until the DDR provides valid data. Since we are applying the scheme in normal execution, the contents of the LR must remain the same. As long as the DDR memory does not provide valid data, the *b* (branch) instruction is provided to the PPC, this will cause the PPC to branch to its current instruction (a branch). Summarizing, the authors of [12] allow the processor to change its internal state, while in our design the processor must be kept in the same state, therefore, we use the *b* instruction.

The normal program flow of the PPC can by interrupted by an internal or external interrupt request. To correctly handle interrupted program execution, the requested instruction address is compared to the previous requested address. As soon as the address changes without supplying valid instructions to the PPC, the ISOCM IPIF controller detects an interrupt. We introduce a waiting mechanism to delay the interrupt request until the DDR memory finished its previous operation. Because the internal state of the processor does not change, the interrupt can be delayed.

DSOCM IPIF: The DSOCM bus uses a 32-bit data path in comparison the the 64-bit data path of the IPIF bus. On a read operation, the least significant bit of the address is used to select the upper or lower 32-bit of the IPIF data bus. The DSOCM bus includes a mechanism to select a byte combination of the 4-byte word for writing. On a write request, this byte selection mechanism combined with the least significant bit of the address is translated to the 8-byte wide selection mechanism used by the IPIF bus. The DSOCM bus on the Virtex-4 supports variable latency memory to be con-



(a) Shared OCM DDR-based.



(b) PLB DDR-based.

Fig. 2. Reference system overview.

nected. The DSOCM bus introduces a read/write complete signal to provide handshaking between memory and PPC. Therefore no mechanism has to be introduced to handle the variable latency of the DDR memory.

IPIF Arbiter: In the shared OCM DDR controller, we use the IPIF Arbiter to allow either the ISOCM IPIF or DS-OCM IPIF controller to gain access to the IPIF DDR controller. The IPIF arbiter is designed in a way that no request will be lost nor information regarding a request will be lost. The IPIF Arbiter monitors each connected IPIF controller to wait for a request using a bus monitor. Once a request is detected, the IPIF signals are buffered to be reused once the controller is granted access to the DDR memory. The IPIF Arbiter uses the read/write request and acknowledge signals to determine the start and end of a memory read or write.

IPIF DDR: The IPIF DDR controller is provided by Xilinx as part of the PLB DDR controller included in the Xilinx Embedded Development Kit (EDK) 8.1. This controller handles all DDR specific requirements like, initialization, precharging and the translation of linear addresses to a combination of bank and row address.

4. VIRTEX-4 IMPLEMENTATION & VERIFICATION

We implemented and tested the OCM DDR controller on the Virtex-4FX12 FPGA on the Xilinx ML403 development board. The development board has 64 Megabytes of DDR memory available of which only 16 Megabytes are used due to address bus limitations. All components of the shared OCM DDR controller are developed using the hardware description language VHDL and are synthesized using the XST synthesis compiler which is distributed with the Xilinx EDK 8.1. The design is implemented as a custom module for EDK. The PPC is configured to run at 300Mhz (max available), the PLB and OCM buses at 100Mhz. Figure 2(a) gives an overview of the reference system based on the shared OCM DDR controller and connected peripherals. Figure 2(b) gives an overview of the reference system based on the PLB DDR controller and connected peripherals.

The VHDL design is verified using the Modelsim SE 6.0 simulator and Xilinx Chipscope Pro 8.1. For simulation, the VHDL simulation model of the Micron MT46V32M8 chip is used.

To verify the correct functionality of both the DSOCM to DDR path and the ISOCM to DDR path, we implemented a memory test program. The test program consists of six main phases of testing, each separated by a debug statement to monitor the progress. The memory test consists of the following stages:

1. Set every location to zero.
2. Compare data read from every location to zero.
3. Write constant value to a random address.
4. Read constant value from a random address.
5. Write random value to a sequential address.
6. Read value from a sequential address and compare to random value.

Random values are generated using the random function implemented in memtest86 [13]. The process is repeated ten times. The software test described, intensively uses the DS-OCM bus. To verify a correct functionality of the DSOCM to DDR path, we stored the program in BRAM blocks connected to the PLB. This test was successful for the complete address range of the DSOCM bus (16 Megabytes). To

Table 1. Resource utilization

	Total	OCM DDR	PLB DDR
Slices	5,472	622(11%)	597(10%)
Flip Flops	10,944	778(7%)	781(7%)
LUTs	10,944	952(8%)	746(6%)

include the ISOCM bus, we used the Xilinx Microprocessor Debugger (XMD), to load the program into the external DDR memory. If the program is stored inside the DDR memory, the region containing the program is excluded from the test.

5. RESULTS

Table 1 presents an overview of the post-synthesis resource utilization. Post-synthesis results of the shared OCM DDR controller indicate that a comparable PLB based controller utilizes slightly less resources. This difference is mainly due to the result of the hardware needed for arbitration included in our design, while the PLB arbitration is not taken into account in the PLB design. Post-synthesis timing estimates present a theoretical value for the maximal operating frequency of the shared OCM DDR controller of 123Mhz and 157Mhz for the PLB design. The difference is due to different VHDL coding techniques, we utilized high-level behavioral VHDL versus device specific structural VHDL for the PLB design. However, the real operating frequency of both designs is 100Mhz.

```
for (j = 0; j < 10; j++){
  p_ddr = &ddr[0]
  for (k = 0; k < SIZE; k++){
    *p_ddr = k+j
    ddr[k] = *p_ddr
    p_ddr++
  }
}
```

Fig. 3. Synthetic benchmark pseudocode.

For performance analysis, we utilized a synthetic benchmark based on two nested for-loops. The equivalent pseudocode is depicted in Figure 3. The boot procedure and timing functions from the Xilinx Board Support Package (BSP) are utilized. Both instructions and data should reside in the DDR memory. Therefore, instructions should be loaded into memory by means of a boot-loader or manually. In this case, due to the small code segment, we decided to load the code manually.

The program is executed on a reference design implemented on the Xilinx ML403 development board. A UART Lite peripheral, connected to the On-chip Peripheral Bus (OPB), is used to print debug and timing information. Since part of the DDR memory contains instructions and the local variables are also stored inside DDR memory, the benchmark can not utilize the entire 16MB.

Table 2. Benchmark results w/o PLB load (x1000 cycles)

SIZE	OCM DDR	PLB DDR	Improvement
128 KB	3485841	3706177	6.32%
256 KB	6968810	7412237	6.36%
512 KB	13940600	14824359	6.34%
1 MB	27884687	29648602	6.33%
2 MB	55745890	59297088	6.37%
4 MB	111528345	118594062	6.34%
8 MB	223048473	237188006	6.34%
15 MB	418315051	444727411	6.31%

Table 3. Benchmark results w/ PLB load (x1000 cycles)

SIZE	OCM DDR	PLB DDR	Improvement
128 KB	3485841	3950069	13.32%
256 KB	6968810	7900061	13.36%
512 KB	13940600	15800044	13.34%
1 MB	27884687	31600012	13.32%
2 MB	55745890	63199947	13.37%
4 MB	111528345	126399818	13.33%
8 MB	223048473	252799557	13.34%
15 MB	418315050	473999103	13.31%

Table 2 and Table 3 give an overview of the obtained results using eight different sizes of the occupied memory. If peripherals connected to the PLB are not used, the OCM based design introduces an average performance improvement is 6.34% compared with the PLB based design. If the peripherals connected to the PLB request the bus every cycle it is available (worst-case), the improvement is 13.34% on average. All results are obtained with the DDR burst mode and with the PPCs hardware caches disabled.

6. CONCLUSION

In this paper, we presented a shared OCM DDR controller implemented and tested on the Virtex-4 chip. We achieved an improvement from 6% (best-case) to 13% (worst-case) over a PLB based design. The best-case scenario is based on a reference system which only uses the PLB to connect memory. The worst-case scenario is based on a reference system with memory and other PLB peripherals connected. In this scenario, a request is constantly sent if the PLB is available. The presented design utilizes a simple arbitration mechanism in comparison to a more complex PLB arbitration. In addition, the presented design benefits from the deterministic nature of the OCM buses.

The resources used to implement the shared OCM DDR controller are comparable to the PLB DDR controller. In our design, arbitration is included in the resource utilization, while on the PLB design arbitration is excluded.

7. REFERENCES

[1] S. Rixner, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory Access Scheduling," in *ISCA 2000: Proceedings 27th annual international symposium on Computer architecture*, pp. 128–138.

[2] W.-F. Lin, S. K. Reinhardt, and D. Burger, "Designing a Modern Memory Hierarchy with Hardware Prefetching," *IEEE Transactions on Computers*, vol. 50, no. 11, pp. 1202–1218, 2001.

[3] J. Carter, W. Hsieh, L. Stoller, M. Swanson, L. Zhang, E. Brunvand, A. Davis, C.-C. Kuo, R. Kuramkote, M. Parker, L. Schaelicke, and T. Tateyama, "Impulse: Building a Smarter Memory Controller," in *HPCA 1999: Proceedings International Symposium on High Performance Computer Architecture*, pp. 70–79.

[4] B. Jacob, "A Case for Studying DRAM Issues at the System Level," *IEEE Micro*, vol. 23, no. 4, pp. 44–56.

[5] A. J. Smith, "Cache Memories," *ACM Computing Surveys*, vol. 14, no. 3, pp. 473–530, 1982.

[6] S. Prybylski, M. Horowitz, and J. Hennessy, "Performance Tradeoffs in Cache Design," in *ISCA 1988: Proceedings 15th Annual International Symposium on Computer architecture*. IEEE Computer Society Press, pp. 290–298.

[7] B. Donchev, G. Kuzmanov, and G. N. Gaydadjiev, "External Memory Controller for Virtex II Pro," in *Proceedings of International Symposium on System-on-Chip 2006*, November 2006, pp. 37–40.

[8] E. Vlachos, "Design and Implementation of a Coherent Memory Sub-System for Shared Memory Multiprocessors," Master's thesis, Computer Science Department, University of Crete, July 2006.

[9] I. Ouaiss and R. Vemuri, "Efficient Resource Arbitration in Reconfigurable Computing Environments," in *DATE 2000: Proceedings Design, Automation and Test in Europe*, pp. 560–566.

[10] K. Lund, *PLB vs. OCM Comparison Using the Packet Processor Software*, Xilinx Corporation, Xilinx Application Note 644.

[11] S. Vassiliadis, S. Wong, G. N. Gaydadjiev, K. Bertels, G. Kuzmanov, and E. M. Panainte, "The Molen Polymorphic Processor," *IEEE Transactions on Computers*, vol. 53, pp. 1363– 1375, November 2004.

[12] G. Kuzmanov and S. Vassiliadis, "Arbitrating Instructions in an $\rho\mu$-coded CCM," in *Proceedings International Conference on Field Programmable Logic and Applications (FPL'03)*, pp. 81–90.

[13] "Memtest86." [Online]. Available: http://www.memtest86.com/