OCM-to-DDR memory controller for VirtexII-Pro FPGA

Jaap van Muijden, Georgi Kuzmanov, Georgi N. Gaydadjiev
Computer Engineering, EEMCS
TU Delft
Mekelweg 4, 2628CD Delft
e-mail: jaapvanmuijden@ce.et.tudelft.nl

Abstract— **We consider the memory subsystem of the Power PC 405 Processor embedded in a Xilinx Virtex-II Pro FPGA chip. There are three processor bus interfaces that can be used for memory access, namely: the On-Chip Peripheral Bus (OPB), the Processor Local Bus (PLB) and the On-Chip Memory Controller (OCM) bus. We are interested in fast memory transfers, therefore we consider the fastest of the above interfaces, the OCM interface. In this paper, we propose a design of a Dual Data Rate (DDR) Memory controller that can be connected to the OCM bus. The controller comprises an OCM bus interface module and a modified Xilinx DDR controller core. To implement time-efficient data transfers between the PPC interface and the external DDR chips, we finely synchronize the OCM interface and the DDR. Our FPGA modules are developed using Xilinx Project Navigator 6.3i and Xilinx Platform Studio 6.3i and are described using VHDL. ModelTech Modelsim is used to simulate and fine-tune the designs. The design has been simulated and synthesized. The results suggest that the hardware costs of the proposed solution are trivial (1%-2% of the available reconfigurable resources). Furthermore, the timing analisys report that the solution can run on a maximum frequency f_{max} of 100 MHz. The controller can be employed as part of a reconfigurable caching structure. The proposed OCM DDR controller solves the size limitation problem of the on-chip memory on Virtex II-Pro. It provides high bandwidth to external memory which typically has a larger volume compared to the on-chip memory. Our proposal makes it possible to build fast and cost efficient memory subsystems including flexible caches.**

Keywords— **FPGA design, DDR memory controller, VHDL, OCM bus**

I. Introduction

One of the key research projects of the Computer Engineering laboratory from the TU Delft is MOLEN [1]. The main goal of this project is to design a polymorphic processor that can be reconfigured at the hardware level during runtime, depending on the target application, at runtime. The processor is realized using existing FPGA technology.

A reconfigurable cache system has to be developed as part of the MOLEN project. This cache memory is to be directly connected to the Power PC Processor core trough the On-Chip Memory interface (OCM). The OCM bus only supports an interface to on-chip Block RAM (BRAM). This type of RAM is very fast, however limited in size. To access bigger volumes of data, external DDR RAM memory can be used, but access to the DDR memory trough the OCM bus is not supported and no OCM compatible DDR controller is provided by Xilinx. The goal of this paper is to design and implement a system that enables the OCM to communicate with the external DDR memory module. This system consists of an existing DDR controller in conjunction with a dedicated controller that matches both protocols.

The system will be implemented on the ML310 development platform[4], which consists of a Virtex-II Pro [3] XC2VP30 FPGA (30,816 logic cells, two embedded IBM PowerPC405 processor blocks, 2,448KB Block RAM), 256MB DDR RAM, System ACE Compact Flash (to store bit stream configurations for the FPGA and programs/data for the PowerPC) as well as various other components. The Virtex-II Pro FPGA contains the processor that will use the DDR memory interface as well as the DDR controller that will be developed in VHDL and implemented in the FPGA fabric. The synthesis results suggest that the hardware costs of the proposed solution are trivial in respect to the available reconfigurable resources(1%-2%). The timing analisys report that the solution can run on a maximum frequency f_{max} of 100MHz.

The remainder of this paper is structured as follows: Section II will present some background, required for understanding the proposed design. This background covers the traditional interfaces of the DDR controller used to communicate with the external DDR RAM and the interface towards the rest of the system. The primary proposal of this paper, the OCM2DDR controller design, is proposed in Section III and an implementation of the controller is presented. Section IV presents the validation of the design through accurate VHDL simulations. Finally, Section V summarizes the findings and presents the conclusions.

II. Background

The external memory type used in this project is DDR RAM (Dual Data Rate RAM). It is similar to SDRAM (Synchronous Dynamic RAM), with the difference that it achieves dual data transfer rate by utilizing both of the clock edges, instead of just the rising or the falling edge in the traditional SDRAM. To interface with such a memory, the existing Xilinx DDR IP core is used. The Dual Data Rate clocking behaviour of the DDR RAM that is used to double the throughput of the DDR interface also brings some additional timing issues. To interface with such dually clocked data bus, the Xilinx DDR controller needs a special clocking infrastructure and uses specially designed input and output buffers in order to process the DDR signals. The External DDR Module is implemented as a Dual In-line Memory Module (DIMM) inserted in the DDR slot available on the ML310 Platform.

The controller logic needed to communicate with external DDR memory is implemented in FPGA using the standard Xilinx DDR controller core. This core is suited for the OPB or PLB bus. However, by removing the OPB or PLB front end of the core, the DDR IP core can be directly interfaced with the more standard IP InterFace (IPIF)[5]. This protocol is used by the proposed OCM2DDR controller to communicate with the Xilinx DDR controller core. The DDR controller is designed using the VHDL language and the core is part of the standard soft core library distributed with the Xilinx Embedded Design Kit (EDK). It can be customized by changing it's parameters or by editing the source code. We will prefer the former method in our approach. The controller takes care of resetting, initialising and refreshing the external DDR memory and provides an easy interface for reading and writing using the IPIF interface. All clock signals are generated with Digital Clock Manager (DCM) modules outside the DDR controller IP; the Controller IP contains only the necessary controller logic.

The IP InterFace (IPIF) protocol is part of the CoreConnect bus architecture developed by IBM. It allows reusability of FPGA (soft) cores over different buses. This is accomplished with the creation of an common IP interface standard that can be connected to any bus supported by the CoreConnect framework, namely the PLB and OPB buses. All EDK peripherals made for these busses consist of an IP core which connects to an IPIF module using the corresponding protocol. Such an IPIF module handles the translation from IPIF to the appropriate bus protocol. The OCM (On Chip Memory) bus of the 405 Power PC is one of the three data buses available for communications to the Power PC microprocessor. It is specifically designed for a fast interface with the on-chip Block RAM memory. The OCM bus is connected to the OCM controller inside the PPC core. It is split up in two parts, the data- and instruction side, usually referred to as DSOCM and ISOCM. Both sides have similar control signals, but the data bus of the instruction side is 64 bits wide, and the data bus of the data side is only 32 bits. This project concentrates only on the DSOCM. The DSOCM has a simple protocol and does not support handshaking of any kind, therefore it only supports fixed access times[1]. The BRAM access time is usually one clock cycle of the PPC system clock, This is referred to as Single Cycle mode. However, it is possible to run the DSOCM bus in a Multi Cycle mode which results in access times of two, three or four clock cycles. This can be used when the BRAM memory block grows too big, since larger BRAM memories usually require two clock cycles or more. The address bus has a width of 22 bytes and the data bus is 32 bits wide. This allows the DSOCM to address 16kb of memory. The address bus of the PPC is originally 32 bits wide: the upper 8 bits are used as an address mask to select the internal DSOCM controller and the lower two bits are dropped, since all DSOCM access is 32-bit aligned. The clock signal used to time the DSOCM bus has to be generated externally by a Digital Clock Manager (DCM). The DSOCM bus features a byte mask containing 4 write enable signals (`DSOCMBRAMBYTEWRITE`), one line for every byte. All DSOCM transactions are perform both a read and a write access. The DSOCM side does not have a read/write indicator and all transactions are both a read and write operations: the masked bytes are written, and the new modified data is copied to the read bus. Since the PowerPC does not support this functionality (it only knows separate read and write operations), the data read back during writes is discarded. This results in the fact that all transactions are considered write transactions with the `DSOCMBRAMBYTEWRITE` as a byte enable signal, except for the case where the byte mask is completely empty: this indicates a read transactions, where all the read data is presumed valid.

III. Design Description

The OCM2DDR controller is implemented in a setup composed of a single PPC connected to a the PLB bus. In this design, both the data and instruction side of the OCM are used: the instruction side is used to store the testing program and the data side is used to test the OCM2DDR controller. The program data itself is stored in the BRAM

[1] The newer Virtex4 does support a non-fixed access time OCM interface.

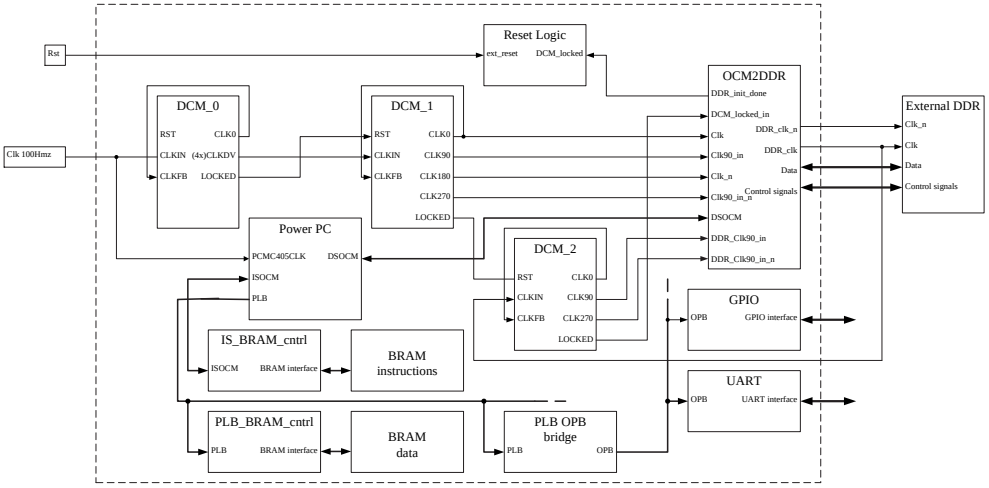

Fig. 1. The organisation of the proposed design.

connected to the PLB bus, making the execution of the program independent of the data side OCM.

Clock Architecture

The standard timing architecture used to create a system with a DDR interface consists of two DCM's: one is used to create main system clock, including the phase shifted versions of the main clock signal. A second DCM is used to create separate phase shifted versions of the incoming DDR clock used to drive the internal DDR controller logic[2]. In order to support a multi-cycle OCM bus, a third DCM is needed to generate the divided CPU clock signal. Another timing issue that is a direct result of the fixed OCM access time is the fact that the system boots immediately after all DCM's are initialised while the External DDR memory needs 200us to initialise. This is resolved by incorporating the DDR2OCM controller into the DCM boot chain. The first OCM starts up automatically since it's reset signal is connected to ground. When the first OCM is initialised, the second OCM's reset is deasserted, and starts up. Additional OCM cores are linked together in this fashion to ensure that all clock signals are initialised before the system boots up. By inserting the OCM2DDR controller into this system, the system boot can be delayed until the DDR has been initialised. this results in the system organisation shown in Figure 1. The CPU clock runs at 100Mhz, with the Multi-Cycle OCM working at 25Mhz. The DDR controller logic runs at 100Mhz.

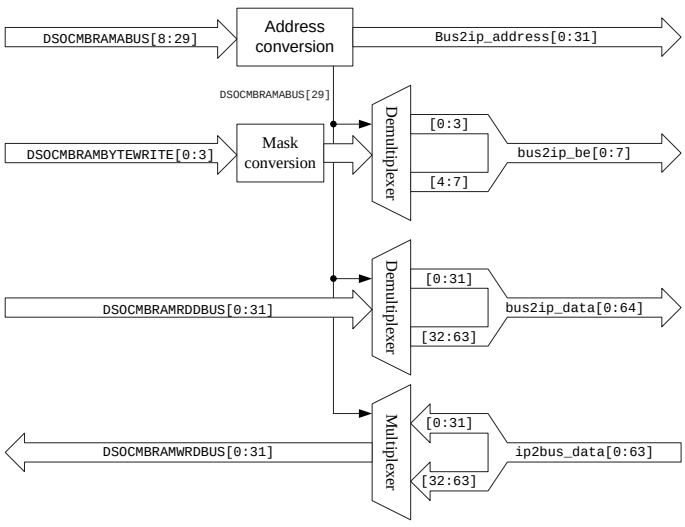

Fig. 2. The data flow of the signal translation logic

Signal Translation

The OCM2DDR controller has to translate the signals created by the OCM controller into the corresponding IPIF signals and vice versa. The Address signals of the DSOCM can be directly translated to address signals on the IPIF side. The IPIF has an address width of 32 bytes, the DSOCM has a width of only 22 bits. Since the IPIF address is byte aligned and the DSOCM is 32-bit aligned, the two lower bytes of the IPIF address will be zero and the 22 bits of OCM address will be placed behind that. The remaining 8 bits will be constantly set to zero. This results in the fact every address of the DSOCM address space is mapped to a subgke address of the DDR controller. In the future, the upper 8 bits can be used to make multiple address spaces and to differentiate between Instruction Side- and Data Side OCM access. The IPIF protocol uses a system called "Byte Steering". This means that the peripheral can address the memory space byte aligned, but the data must be provided in the correct byte lanes, in compliance to the base bit alignment of the bus. This means that the address is given is a byte address, but the byte mask and data are aligned to the width of the data bus (in this case 64 bit).

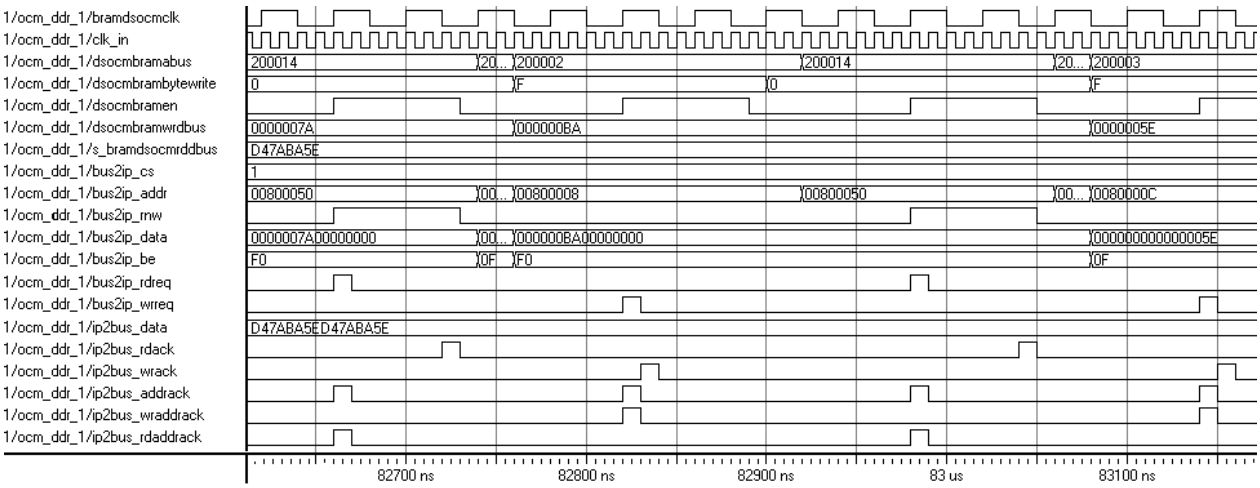

Fig. 3. Simulation of the OCM2DDR controller

The address generated from the DSOCM is always aligned to 32 bits, leaving only two possibilities: the data is aligned on 64 bit, meaning that the data is copied to the lower 32 bits of the IPIF data bus, or the data has an offset of 32 bits from the nearest 64 bit alignment boundary, meaning that the data is put in the upper 32 bits of the IPIF bus. This depends on the Least Significant Bit (LSB) of the DSOCM data bus, when this bit is '0', the address is 64-bit aligned, if it is '1', the data has to be put in the higher 32 bit of the IPIF data bus. This conversion holds for both the incoming, and outgoing data, and the data mask has to be shifted accordingly. Both the data mask of the IPIF and the OCM mask the data on a byte level. The byte mask of the IPIF masks the bytes that contain valid data and the DSOCM mask masks the bytes that are to be written to the BRAM. For write operations, this means that the byte mask can be simply copied. However, for reading operations, the meaning of the byte mask differs: The DSOCM byte mask is completely empty, but all the data on the bus is expected to be valid. The IPIF bus has separate read/write indicator signals, and the byte mask validates the data in both a read and a write operation. This means that in the case of a read operation, the DSOCM byte mask is empty, but the translated IPIF byte mask should be completely asserted. This leads to the signal translation as shown in Figure 2.

IV. Synthesis Data and Simulation

The design has been synthesized using Xilinx Platform Studio 6.3i. Synthesis results, presented in table I suggest that the hardware costs are trivial in respect to the available reconfigurable resources (1%-2%). The reported delays suggest a maximum frequency f_{max} of 100 MHz. The design has been tested using multiple simulations done in ModelTech ModelSim 6.2 SE. The biggest issue is the fact that the DDR access time can vary greatly and are very difficult to simulate correctly. For simulation purposes, a test program is created that writes and reads into the address space of the DSOCM to test the functionality of the OCM2DDR controller. A variety of peripherals are connected to both the OPB and PLB to provide extra functionality to facilitate the testing process: General Purpose IO drivers to enable the manipulation of the various GPIO interfaces including the led array on the development platform and an UART interface peripheral to communicate with an external PC. This latter is the main debugging interface used by the running software. The standard input and output streams of the running C programs are redirected to this peripheral to create an effective interface system to the user. The program reads and writes linearly into the DSOCM address space, resulting in the data flow shown in Figure 3.

As can be seen in Figure 3, the DDR access is completed within the OCM bus assertion. The OCM runs in 4x Multi Cycle Mode. The DDR simulation model has a CAS latency of two. The simulation implies that the system works, although it will not work as soon as the DDR access time rises above the best-case access time. In order to fix this, the system clock frequency has to be lowered, or the fixed-access time problem of the OCM has to be resolved.

V. Conclusions

In this paper we presented a feasible and efficient way to access a large memory volume trough the OCM bus with the VirtexII-pro. This interface can be used in future projects e.g. as starting point for the design of a caching subsystem in the FPGA fabric. The modular construction of this system enables the user to change the DDR inter-

TABLE I
SYNTHESIS DETAILS OF THE OCM2DDR CORE

Minimum clock period	9.613ns
Minimum input arrival time before clock	7.285ns
Maximum output required time after clock	1.480ns
Maximum combinational path delay	2.097ns

Logic Type	Amount	Prercentage
Number of Slices	311	2%
Number of Slice Flip Flops	394	1%
Number of 4 input LUTs	22	1%

face without changing the rest of the system. Optionally, any data interface that implements the IPIF protocol can be connected to the OCM2DDR controller. In this context, the OCM2DDR controller can be used as a universal solution to connect IPIF compatible peripherals to the OCM. The synthesis details are presented in Table I.

At this time, the fact that the OCM bus doesn't not support all functions of the IPIF makes it impossible to support a non-fixed access time. Interrupts and DMA accesses are also not supported at this time. Since the OCM does not support these systems natively, a workaround has to be found outside the current OCM interface.

Future work will include the development of a configurable caching system using the OCM2DDR interface as a basis for communication with the DDR controller. To make full use of the Dual Data Rate of the DDR memory, burst-access should be introduced. In addition, the access time of this cache subsystem will be variable. This is not possible with the current OCM controller implementation, A possible solution is to use a different base architecture. (e.g. the Virtex4) which does support a non-fixed access time on the OCM.

REFERENCES

[1] S. Vassiliadis, S. Wong, G. N. Gaydadjiev, K. Bertels, G. Kuzmanov, and E. M. Panainte. The molen polymorphic processor. *IEEE Transactions on Computers*, pages 1363– 1375, November 2004.

[2] H. Winkler. Clocking strategy for a virtex-ii-pro ddr sdram controller.

[3] Xilinx. Virtex-II Pro platform FPGA handbook. Xilinx, October 2002.

[4] Xilinx. Xilinx ml310 development board. Xilinx, http://www.xilinx.com/products/boards/ml310 2002.

[5] Xilinx. Plb ipif (v2.01.a. *Xilinx Logicore*, http://www.xilinx.com/bvdocs/ipcenter/data_sheet/plb_ipif.pdf 2004.