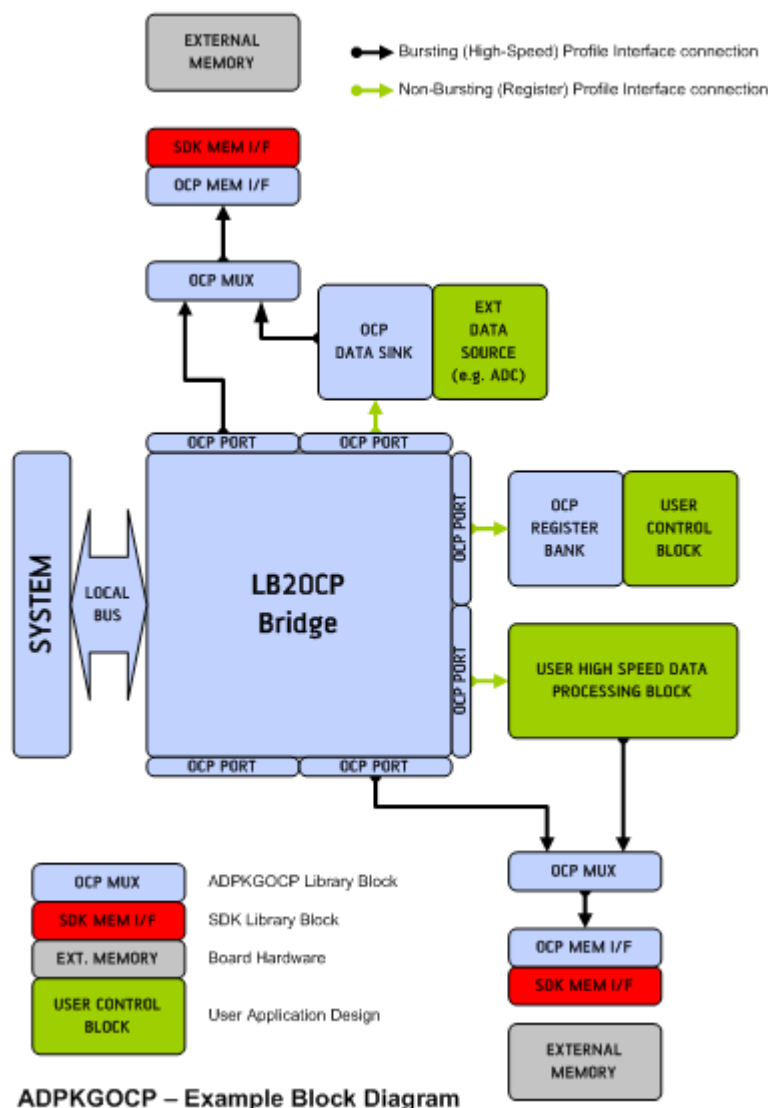


# An Introduction to using OCP on Alpha Data Products

## Introduction

This App. Note describes the operation of a sample OCP design implemented in one of the Alpha-Data FPGA boards, explaining what it is, and what it is not.

Firstly, **OCP** is more of a methodology than a standard. It makes both the specification of a wide variety of interfaces possible, while allowing their connection with relative ease. It does not impose overly restrictive standardisation, making it difficult to develop IP, however it does provide a framework, which if adhered to produces interfaces which can be connected with minimal manual intervention.



This library package therefore defines a number of interfacing protocols consistent with the OCP-IP<sup>1</sup> standard however it does not set out to support automatic connection of modules, as the number of cases where this will be useful will be limited.

The library packages aim primarily to provide useful IP blocks for connecting up memories, off-chip buses and application processing modules, to create an application within an Alpha Data FPGA board, easing the definition of host software controlled registers and data flows.

This is effectively a higher level library which sits on top of the basic Alpha ADM-XRC-SDK. It does not

replace this and in many cases relies heavily on the modules provided in the SDK. It does however provide a standard module connection to the local bus, removing the complex and potentially error prone task of attaching and arbitrating between many modules on that bus. Some knowledge of the ADM-XRC-SDK<sup>2</sup> is assumed, although some of the finer details of local bus use is not required, and overall understanding of

<sup>1</sup> <http://www.ocp-ip.com>

<sup>2</sup> See [www.alpha-data.com](http://www.alpha-data.com) for more details on the ADM-XRC SDK

how the Alpha Data PMC cards work, how they connect to memory and the host, will be very useful before using this library.

The library is provided in VHDL as packages as this language best supports a higher level of hardware design. While the strong typing of the language is useful in detecting bugs which might go unnoticed in Verilog, it is the support for record types, allowing the collection of interfacing profiles into a single type for each direction of data flow which is most useful.

The library also defines 2 general classes of interfacing profiles to handle the different ways in which memory mapped data is transferred: low speed, non-bursting profiles are defined for register type access, such as the host controlling the run-time operation of the FPGA application, or checking the status; high speed bursting profiles are defined for memory and other large data accesses, and these can be used for bulk data transfer from the host to the FPGA and vice versa.

The block diagram above shows how the ADPKGOC blocks can be used to connect up a system. Package blocks provide basic connectivity functionality such as the local bus bridge and multiplexing. They also provide interfacing for memory devices although the complex memory specific SDRAM interface logic is all contained in the SDK Memif block. For user applications, in some cases the Data Sink and Data Source blocks can provide the connection between a basic data-flow interface (e.g. an A2D port or a Xilinx Local Link), and the more complex OCP profile needed to store the data in memory. In other cases, library blocks can provide a register bank interface for the simple instantiation of a small number of control registers. Users can also build their own custom OCP profile blocks, either by following the protocol and implementing it directly, or by modifying an existing block to meet a specific requirement, or by using several interface library blocks in the same module and using MUX and DEMUX blocks to connect them up.

The main advantage of this approach is that the standardised framework allows significant reuse of developed modules. Many different designs can use the board support features provided by the local bus bridge and memory interfaces. Modules designed for this framework should be easily re-used. By basing the interface profiles on the OCP standard importing existing OCP modules should also be fairly straight forward, as should the export of modules designed in this framework into other OCP designed systems.