

Socket-Centric IP Core Interface Maximizes IP Applications

Semiconductor intellectual property (IP) designers strive to ensure their IP can be utilized by the widest possible range of applications to ensure maximum return on their engineering investment. However, the common older practice of supporting a *bus-centric* protocol as an IP core's native interface ultimately limits the market into which an IP core can subsequently be utilized or sold. Fortunately, there is now an optimized and fully-supported interface approach available that utilizes the benefits of the OCP SOCKET.

Bus protocols are based upon traditional printed circuit board styles of interconnect structures that consist of hierarchical wire bundles, and are proving to be ineffective for System-On-Chip (SOC) designs. All bus protocols rigidly define an inter-block data-flow communication methodology, and demand the use of this signaling style to the exclusion of any other technique. Also, sideband control and test signals are typically not supported by computer-bus style protocols, and require the system integrator to deal with them in an ad-hoc way for each system design. This means that if an IP designer inflicts a particular bus protocol upon his IP core, reuse with a different bus protocol may only be possible via the loss of features or performance, or may not be possible at all.

The solution to maximizing an IP core's potential reuse, and realizing the advantages leveraged by proven industry standards, is to adopt a well-specified *core-centric* protocol as the IP's native interface. A comprehensive and scalable interface specification between IP cores and on-chip interconnect systems allow IP core developers to focus on core generation without having to know any details about the end-systems in which a core might be eventually utilized, nor anything about the other IP cores that might be a part of those end applications. Essentially, the problem is best solved with the use of a complete and fully supported SOCKET.

With this socket-based approach, system integrators benefit by not having to deal with an unending series of unique core protocols and delivery styles. The use of standard IP core interfaces eliminates having to repeatedly adapt each core in preparation for each SOC integration, and instead allows the system integrator to focus on system level design issues. Also, since standard interfaces decouple the cores from the on-chip interconnect and from each another, it is simple to change out one core for another to better meet changing system requirements.

For an IP core to be truly reusable it must remain untouched as it moves from system to system. A core's interface must represent the unchanging requirements of the core, rather than the continuously differing requirements of each system's interconnect. A core need not be re-adapted every time the bus width, frequency, or electrical loading changes if a complete socket is specified.

Since core interface requirements are as diverse as the IP cores themselves; there is no rigid one-size-fits-all interface. A standard core interface specification must be scalable and configurable to adapt to the wide range of requirements. It is also not sufficient for an interface specification to only capture data-flow signaling. It is important that *all* signaling between a core and the system are captured. Non-dataflow control signals (such as interrupts, error signals, flow-control signals) and test signals (used for debug and test of the completed chip) are typically ignored or incomplete; except for the **OCP**.

The freely available **Open Core Protocol™** (OCP™) is a *bus-independent* protocol that meets all the *core-centric* considerations discussed above, and completely captures *ALL* of an IP core's communication requirements. The highly configurable OCP is not a one-size-fits-all protocol, but is instead analogous to a continuum of protocols that have a common definition structure. Sideband signals are explicitly supported via optional extensions to the basic OCP for reset, interrupts, errors, control/status information, etc. In addition, a generic flag bus is used to accommodate a core's unique signaling needs. The optional test interface extensions of the OCP support scan, JTAG, and clock control, enabling debug and manufacturing test of the core when integrated into the system-on-a-chip. A core's specific OCP configuration is, therefore, tailored to match the core's requirements exactly. A simple, low-performance core can have a very simple interface, while a complex, high-performance core can be accommodated just as effectively.

An IP developer can therefore *complete* an IP core design using the OCP interface. No end-application knowledge is required beyond the OCP, allowing complete independence between members of (often) global design teams. The system integrator is also free to choose the on-chip interconnect that best suits the system requirements of the application, then effectively "wraps" that interconnect to present OCP interfaces to the cores.

An IP core utilizing the OCP as its native interface can be easily reached by *any* bus structure the end-customer chooses through simple bridge or "wrapper" structures. Since the OCP doesn't constrain a core's functionality, every bus bridge to the OCP is able to reach that core's maximum capabilities. An IP developer could also pre-design a selection of bus bridges for some of the more common, non-OCP-compliant bus structures that a customer may choose. The work required to design such an OCP interface "wrapper" for a core is bound by the distinct choices that the OCP protocol itself offers, so there typically is only a *regular set* of wrappers that need to be provided. In fact, the wrapper generation process is regular enough to be amenable to automatic interface synthesis.

The alternative of designing multiple bus bridges from a core whose native interface was made to conform to a bus-centric protocol, (one that may have been chosen by an IP designer who perceived that bus to be the most popular at the time of implementation), creates a starting point with rigidly defined limitations. Whenever there are differences in data and address presentation sequences between that core's rigid native bus protocol and the target bus, the core's performance will likely suffer. This is due to the bridge-on-a-bridge effect of having to correlate the signaling between the two disparate bus structures

at their lowest common denominator. The implementation gate count for the bridged core is also likely to be higher.

An interesting and very relevant customer-based case study showed that installing the OCP on a slave USB core as the native interface, then building a bridge to one of the ARM AMBA bus protocols required no more implementation gates than installing the AMBA AHB protocol as the core's native interface. In this instance, both implementations also passed through the full capabilities and performance of the core, although the AHB-native core required custom-defined implementations for all control and test signals since the AMBA bus standard did not directly address their implementation. When these two cores were further interfaced to an IBM CoreConnect bus protocol (the OPB), the OCP-native core simply required a replacement bridge that again delivered all the core's capabilities and performance to the OPB. The AHB-native core, however, required incremental bridge logic from the AHB interface, and was only capable of performing at *half* its bandwidth capability to the OPB due to the differences in Address and Data signaling between the two busses.

In summary, the core-centric OCP is an openly licensed, royalty-free protocol that does not impose restrictions or interfere with an IP cores' inherent capabilities. It is scalable and configurable to match the different communication requirements associated with different IP cores. The OCP functions ideally as a native IP core interface since bridges to any bus or integration structure can be implemented without the gate count or performance penalties typical of bus-centric protocols. Cores with OCP interfaces and wrapped interconnect systems, enable true plug-and-play hardware integration, thus allowing the system integrator to choose the best cores and best interconnect system for an application. Finally, verification and test suites, when written to the OCP are completely portable to multiple designs, possibly requiring only minor and occasional adjustment for a particular interface bridge.

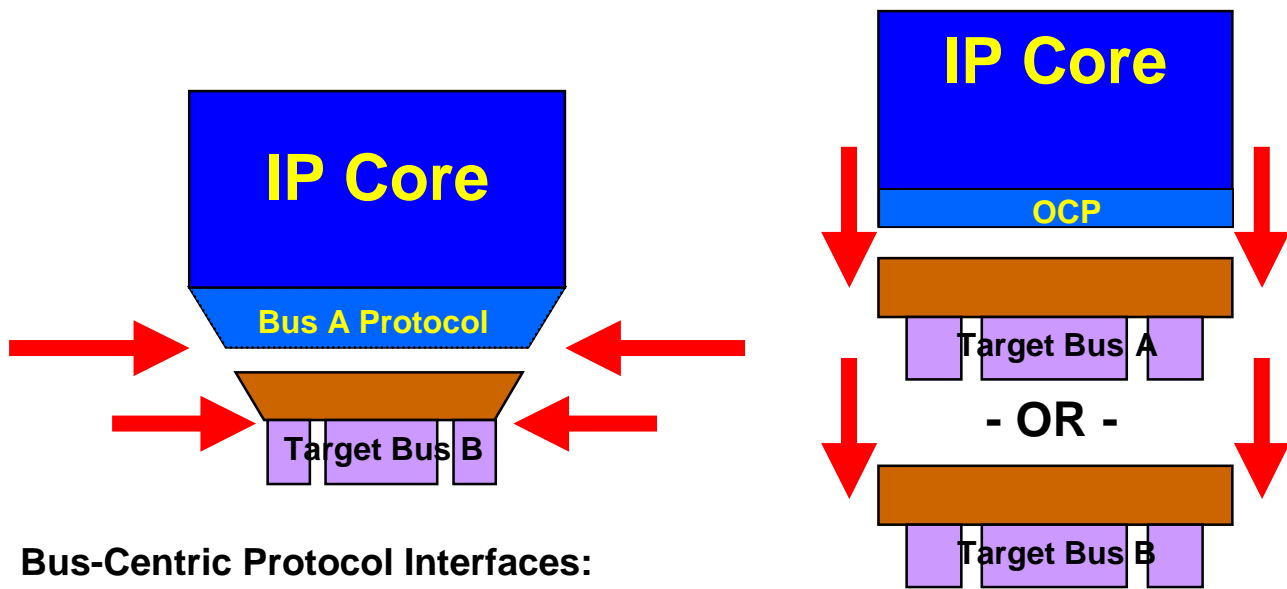
A standard IP core protocol is essential to the SOC design community and OCP is the **ONLY** complete, fully supported and proven socket. With immediate adoption we are able to avoid the proliferation of non-compatible proprietary solutions, and accelerate the market for, and reuse of, commercial and legacy IP cores. The complete, fully supported core-centric OCP delivers substantial and demonstrable benefits over older style bus-centric protocols.

OCP-IP members receive, at no charge, the CoreCreator™ tool as an OCP protocol compliance checker, and environment "packager" for all the representations necessary for efficient reuse of an IP core.

The OCP specification is freely available at <http://www.ocpip.org>

SEE ATTACHED CORE VERSUS BUS-CENTRIC DIAGRAM...

Core-Centric vs. Bus-Centric



Bus-Centric Protocol Interfaces:

- Bus configuration assumptions for a native interface limit IP Core features
 - *AND* – Eliminate or limit capabilities in successive bus structures
- Do not address *ALL* core communication
- Sideband Control and Core Test are not addressed

OCP is a Core-Centric Protocol:

- Facilitates unrestricted delivery of *ALL* Core signals and features
- Enables unconstrained interface bridge to *ANY* bus structure