MAKING FPGAs "FIRST CLASS" SCA CITIZENS

Fred Humcke (Hardware Director SDR Products, PrismTech Corp., Burlington, Mass.; fh@prismtech.com)

ABSTRACT

In recent years, Software Defined Radio (SDR) developers using the Software Communications Architecture (SCA) have been making a steady march towards the antenna of the software defined radio. The next obstacle for the SCA to cross lies in the interface between embedded processors and FPGAs. Previous attempts to solve this problem have resulted in awkward and non SCA-compliant Hardware Abstraction Layers (HALs) that have only added latency, decreased portability and lowered reuse of the SDR processing elements residing in the FPGAs. The result is that FPGAs are "second class" citizens who do not contribute to the cost lowering business model of the SCA.

This paper will discuss an Integrated Circuit Orb (ICO) that supports a drop-in SCA compatible interface between distributed software objects running on processors and waveform objects residing in silicon. Using techniques discussed in this paper, the connection between Software and Hardware clients and servant is made seamless, fast and uses fewer system resources. It will be shown that waveform designers can use ICO to make their FPGA designs first class SCA citizens without the time consuming task of becoming experts in the minutia of the SCA specification.

1. INTRODUCTION

In the years since it was first defined, the SCA has made steady progress towards producing true software defined radios and realizing the financial benefits of portable radio design. The goal is to push the SCA as close to the antenna as possible to reap the maximum benefits of the business model. Yet, the pace of acceptance within the industry has been slowed by certain perceptions about the SCA and CORBA that may no longer be valid. These perceptions include:

- The SCA operating environment is large and takes up valuable system resources such as memory.
- The SCA/CORBA is low performance because the TCP/IP stack adds too much overhead for simple data transfer.
- The SCA is complex and difficult (expensive) to learn and utilize.
- There is no SCA solution for communicating with waveform components implemented on silicon devices such as FPGAs and ASICs.

From a historical perspective, these issues are similar to those faced in the transition from low level programming languages such as assembly to higher level programming languages such as C. There was much resistance to C at first because assembly code is faster and takes less memory space than C and the early C compilers and emulators had many problems associated with them. The historical experience with C shows that there are two items necessary to make a transition to a new level of design abstraction. The first and most important item is the business case. In a competitive market, companies will strive to implement a new paradigm if there are financial rewards associated with it. The business case is called the driver and in the example of C, the driver was cost savings through portability. Assembly might be faster and smaller than C code, but it has to be rewritten every time it needs to run on a new processor while C code is written once and can be quickly ported to various platforms. While the business case drives the industry to work towards a paradigm shift, the new level of abstraction still cannot be accepted until technological breakthroughs enable its practical use. Thus, the second item that is required for abstraction acceptance is the technical enabler. In the case of C the enablers were technical breakthroughs in both hardware and software and included such items as

- increases in processor performance,
- increases in memory size and density,
- smaller software footprint,
- more efficient software processing tools (compilers),
- system elements that were bundled together in a single package with no need to make elements from different vendors work together (operating systems to handle low level hardware interface),
- high level tools to remove complexity from user (emulation and debugging environments).

The combined effect of these enablers allowed C to achieve wide industry acceptance and be used in many applications. Thus, acceptance occurs when the overhead of a high level abstraction no longer has significant impact on system performance. Assembly is still used today in

applications that have strict performance requirements; however, it is known that this comes at the cost of portability in a changing design.

Turning now to the current state of the SCA and CORBA, we see the recent development of similar types of enablers for this field. Advances in performance, size and density continue to be made for processors, memory and FPGAs. The size of CORBA and SCA implementations has shrunk dramatically in recent years. Products such as PrismTech's E*ORB require less than 80KB of memory compared to 1MB for the equivalent TAO ORB. Indeed, PrismTech's entire Spectra OE middleware for GPP takes less than 1MB. These ultra compact CORBA/SCA implementations are now available and running on embedded GPPs and DSPs residing on radio processing platforms once thought too limited in memory and processing power to contain them.

Another breakthrough has been in the area of domain specific modeling and automatic code generation. Just as better C compilers and operating systems obviated the need for the software designer to be involved in the time consuming minutia of the underlying hardware, PrismTech's Spectra tool suite allows the SDR developer to step back from the complexities of the SCA and work at a higher level. The developer may now work at the graphical level to describe an SDR system and then have the Spectra tool generate the required SCA source code.

The Spectra Modeling tools are Eclipse plug-ins that enable the graphical modeling of both applications and platforms at a high level of abstraction. Spectra Modeling Tools help application developers to efficiently model platform-independent, portable applications and "map" those application models to different platforms. These tools also support optional plug-ins for complementary 3rd party tools used in the development lifecycle – thus supporting complete domain specific tool-chain integration.

PrismTech's automatic code generators for various programming languages provide extremely efficient generation of source code and unit tests from the waveform models. Efficiency gains of up to 50 x (several months to a single day or less) have been experienced. Furthermore, they inherently ensure standards-compliance in the source code.

The Unit Test Framwork offered by PrismTech supports 'in-cycle' unit testing of generated source code; if necessary, well in advance of platform availability. This strategy allows defects to be corrected during design verification instead of during run-time test integration which results in huge productivity gains.

A truly disruptive technology breakthrough has been achieved in PrismTech's Integrated Circuit ORB (ICO). The ICO brings CORBA communications directly onto hardware platforms such as FPGAs and ASICS. It accelerates the marshaling and unmarshaling of CORBA messages at 100 x the speeds of software ORBs. When used in SCA applications, ICO eliminates the need for Hardware Abstraction Layers (HALs) that have only added latency, decreased portability and lowered reuse of the SDR processing elements residing in the FPGAs. Waveform objects may now be moved from software implementations on the GPP/DSP to hardware blocks on the FPGA and back again with no changes needed in the high level communication protocol. The ICO is completely based on open standards so there are no issues with proprietary protocols or interfaces. PrismTech is working with the SDR community on VDHL language bindings for the hardware ORB and plans to have them standardized by the OMG.

The sum total of all the enablers discussed above brings CORBA and the SCA to the edge of a new era of development and worldwide acceptance. These breakthroughs have revitalized CORBA and made it the equal if not the better of any middleware solution on the market today. The SCA has now become as easy to work with as C and other high level programming languages. PrismTech strongly believes that the continuing advances in processor performance, memory density, software efficiency, support tools and hardware ORBs will enable the SCA's business model to propel SDR forward into widespread use in all military and commercial markets in the near future.

2. HISTORICAL PROBLEMS IN USING HARDWARE WAVEFORM COMPONENTS WITH THE SCA

In the past, two methods have been used to establish communications between SCA waveform objects residing on a GPP/DSP and waveform objects implemented in FPGAs. The first method makes use of the hardware abstraction layer. An example of this is shown in Figure 1 below.

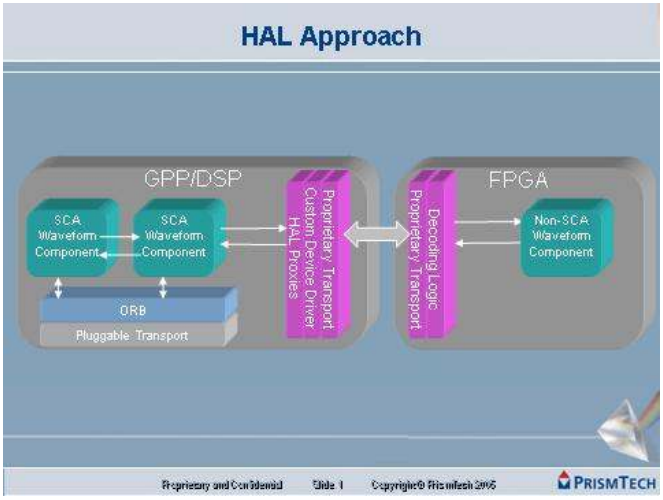

Figure 1.

With this method, a proxy on the GPP/DSP stands in for the hardware waveform component. When a software object

makes a call on a function implemented in hardware, the proxy is activated and passes data to the local transport via a driver. The data is transferred over the local transport, such as a system bus, to a transport interface on the FPGA side. The transport interface performs address decode and passes the data to the desired waveform object. There are several problems with this method. One problem is that the proxy is awkward to implement and is not portable because it must contain low level address mapping information in order to identify the destination waveform object correctly. In addition, several layers of software may be needed to implement the proxy/driver stack which introduces latency. Finally, this solution is not SCA compliant because the CORBA message ends at the proxy and there is no ORB on the FGPA side.

Other attempts have been made to bring the SCA into the FPGA by running an ORB on a processor embedded within the FPGA. This only allows CORBA communications and SCA compliance with software objects running on the embedded processor itself. However, waveform objects implemented in the hardware of the FPGA still require the HAL in order to communicate with the embedded processor as shown in Figure 2.

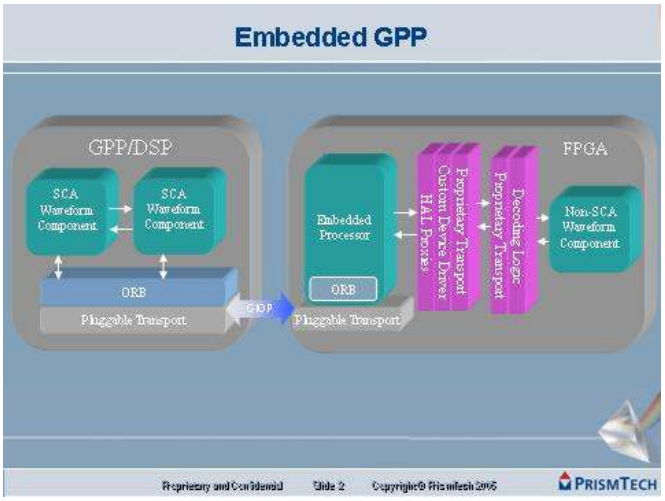

Figure 2.

Thus, all the problems associated with the HAL have merely been moved from the external bus to the FPGA's internal bus and there is still no SCA compliance for the waveform hardware objects. Additionally, FPGAs with internal processors are expensive and the memory needed to support the processor uses up valuable resources. At a particular price point, an internal processor and associated support memory take FPGA resources away from waveform logic. The introduction of the embedded processor also requires that software development and debug be done in the FPGA environment.

The problems discussed above conspire to make FPGAs second class SCA citizens and slow the adoption of the SCA for military and commercial applications. This shows the need for a CORBA ORB implemented at the gate level within the FPGA. The hardware ORB allows software objects located on the external GPP/DSP to communicate with the waveform objects on the FGPA without the need for a proxy or embedded processor. The communication, as far as the objects are concerned, takes place at a high level of abstraction with no knowledge required of the local transport or address mapping scheme. This concept is illustrated in Figure 3. The result is seamless SCA compliance across all objects in the waveform and the elimination of layers of software that introduce design complexity and performance latency to the system.

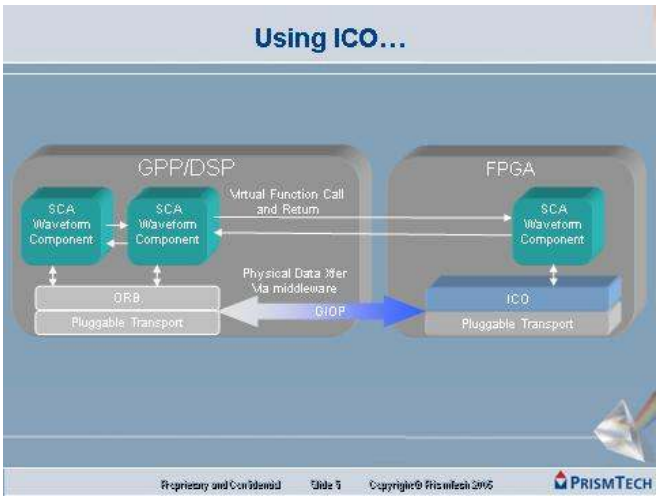

Figure 3.

3. DESCRIPTION OF THE INTEGRATED CIRCUIT ORB

The Integrated Circuit ORB (ICO) is part of PrismTech's family of products for CORBA middleware applications. Hardware elements of an enterprise system may now be made CORBA compliant and reap the benefits of software portability. In addition to supporting general purpose CORBA communications, the ICO has also been tightly integrated into PrismTech's Spectra tool suite for Software Defined Radio. This brings the portability of the SCA onto silicon devices.

ICO is a hardware implementation of a CORBA ORB. It supports a general subset of CORBA functions that will support the SCA architecture. While ICO may be used to provide SCA compatibility, it is primarily a CORBA core and may also be used in pure CORBA applications with no SCA requirements. For SCA applications, additional functionality may be added via PrismTech tools to implement the SCA component. The ICO has been written in portable VHDL that can be synthesized onto any FPGA or ASIC platform.

The ICO design environment consists of:

- The ICO core,
- IDL to VHDL compiler,
- Spectra Modeling Tool,
- The optional SCA waveform component.

The ICO core is responsible for implementing the transfer syntax used in CORBA messages. The core unmarshals the incoming GIOP octet stream and extracts header and data fields while discarding padding. Endian conversion is performed on all incoming data based on information in the GIOP message header. In the incoming direction, the core performs operation name demultiplexing to determine which object the data in the GIOP message is being transferred to. Message data is then extracted for transfer to the appropriate logic.

If a message indicates that a response is expected, the ICO core generates a reply message. The core will perform a read operation to an object, if necessary, to obtain data for the reply. It then populates the header field and aligns the data. When a reply message has been built, the ICO core transfers the data to the local transport via a FIFO-like interface.

An optional feature may be added to ICO in order to support SCA implementations. The Spectra tools will instantiate and parameterize an SCA component that performs functions such as registering with the naming service, configure and query. It will also decode SCA messages such as runTest, start, stop, initialize, release object and pass these instructions on to the appropriate waveform objects. Thus, seamless compatibility with the SCA is achieved.

The ICO can also be configured to support FPGA waveform objects that act as clients. When the embedded object makes a function call request, ICO will gather the data from the object and then marshal a GIOP request message and send it to the desired client via the local transport. Any returning reply messages will be processed by ICO and the data passed back to the client.

4. THE ICO DESIGN FLOW

4.1 Waveform IDL Description

In order to incorporate ICO into an FPGA design, the user must first describe the memory elements of the waveform component with the Interface Description Language (IDL). Each accessible register and memory in the waveform component is given an operation name and its I/O properties are described in IDL. Registers can be accessed alone or in groups depending on how they are described in the IDL. If it is desired to write a register and read it back at a different time, the register would require two operation names; one for write and one for read. If it is desirable to write a register and read the results immediately it could be described as a single operation with an inout parameter. PrismTech provides tools that can aide hardware engineers in writing the IDL descriptions.

4.2 IDL to VHDL Compiler

The IDL description of the waveform component must then be processed by the IDL to VHDL compiler to generate ICO configuration files. The IDL to VHDL compiler is part of PrismTech's IDL compiler family. This software tool is responsible for generating configuration parameters needed by the ICO core to do operation name demultiplexing and data routing to the appropriate waveform objects. The compiler also adds parameters to VHDL package files that configure the physical aspects of the ICO interface and internal storage elements. Parts of the VHDL code for the ICO are also generated at compile time by the code generator. Each operation receives its own unique read and or write strobe depending upon how it was described in the IDL. The IDL to VHDL compiler produces a list of operation names and the strobes assigned to them so that the user can make the appropriate connections in the top level VHDL code. PrismTech tools can automatically make these connections for the user.

4.3 SCA Compatibility

To achieve SCA compatibility the designer must also pass the IDL description of the waveform through PrismTech's Spectra Modeling Tool. In an SCA-compliant environment, ICO communicates with the hardware developer's native waveform logic via an SCA waveform component. Spectra instantiates and parameterizes the VHDL for this component in the design. The tool then creates a VHDL wrapper around the ICO and the SCA waveform component to present the developer with a single core. The ICO is now ready to be instantiated in the FPGA HDL. The process for waveform component generation with the Spectra tool is illustrated in figure 4.

Additionally, a loadable device component will also have to be developed for the SCA OE. For initial implementations of ICO, the loadable device component must reside on an external GPP/DSP that will be responsible for loading an image into the FPGA at power up. It may be possible to move the loadable device functionality into the FPGA itself using partially reconfigurable devices as that technology becomes more mature.

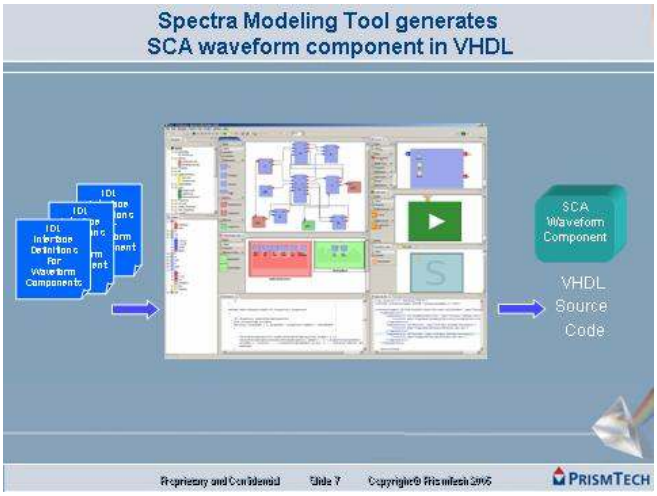

Figure 4.

4.4 Adding ICO to the FPGA Design

The hardware developer treats the ICO as any other IP interface core. In typical FPGA designs the ICO core resides between the waveform logic and the local transport and takes the place of the address decode block found in conventional bus interface designs. The basic design process of the FGPA is unchanged as it relates to waveform and system bus performance considerations. Figure 5 shows how ICO might be used in a typical application that communicates with command and control logic in the waveform object.

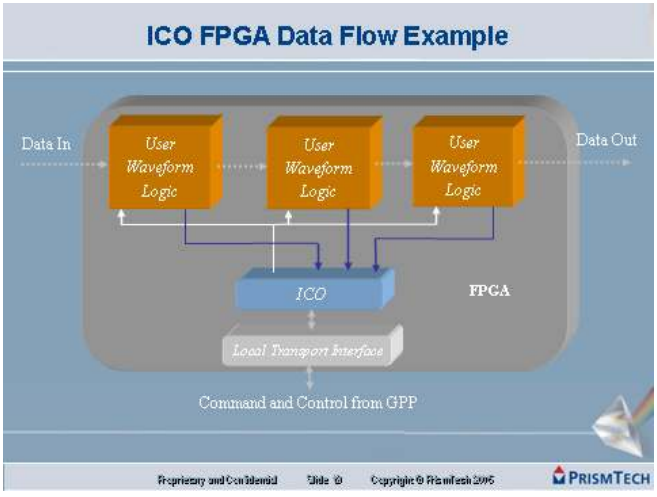

Figure 5.

ICO may also be used in data paths between waveform objects within FPGAs and between FPGAs themselves, however, this is a system consideration and the performance requirements must be taken into consideration. Just as in a software system where assembly code might still be chosen over C for high end performance, the FPGA designer may choose to omit CORBA from custom point-to-point high speed communications with the understanding that portability is sacrificed. Enhancements to the ICO will enable it to be used in more diverse high speed communication applications in the near future.

Software developers treat ICO components as they would any other CORBA object. This design approach makes communication between the S/W and H/W objects seamless. Using ICO, radio developers can now host radio elements in an FPGA and still have them be addressable and callable from an SCA-compliant software core framework as though it was an SCA object and not an FPGA.

5. ICO AND ETF

The ICO communicates with software ORBs over the local transport. In many cases the local transport will be a low level system bus that does not support high level communication protocols. This situation requires that software ORBs make use of the Extensible Transport Framework (ETF) defined in the CORBA specification. An ETF must be created to transfer GIOP messages from the software ORB to the local transport along with any header or encapsulation data required. This situation is not unique to ICO, but would be required for any ORB that does not support the standard TCP/IP. Typically, the ORB vendor supplies the ETF at the customer's request. PrismTech has developed an ETF for Ethernet and plans to create similar software for several other common transport schemes as per customer requirements.

On the ICO side, the equivalent ETF functionality must be handled in hardware. An ETF interface block may need to be inserted between the ICO and the local transport. This ETF interface is required if ICO is used to support user waveform blocks that act as CORBA clients. When an embedded client waveform component makes a function call, ICO will read IOR information from an internal lookup table and use that data to marshal the message header. In addition, ICO will prepend transport address data to the front of the GIOP message. The ETF interface block may then use that information to encapsulate the GIOP message as per the requirements of the local transport. The encapsulated message will then be sent to the transport interface for transmission to the servant ORB. PrismTech will work with the user to develop the ETF interface design according to the needs of the system.

6. CONCLUSION

PrismTech has created a powerful solution for implementing CORBA messaging on silicon devices such as FPGAs and ASICs. CORBA is an open standard and the Integrated Circuit ORB (ICO) is a hardware implementation of the standard. Using CORBA to communicate between GPP/DSPs and FPGAs simply is, by definition, SCA-compliant because the SCA specifies CORBA as the preferred protocol to be used to communicate between SCA waveform components. Using CORBA protocols between GPP/DSPs and FPGAs also eliminates the need to implement needlessly complex Hardware Abstraction Layers (HALs) in the waveform data flow chain as well as the supporting proprietary protocols that go with them. While ICO may be used to provide SCA compatibility, it is primarily a CORBA core and may also be used in pure CORBA applications with no SCA requirements if needed in the future.

The Integrated Circuit ORB is flexible, highly configurable and uses a minimum of FPGA resources. It frees the hardware developer from learning and implementing the complexities of CORBA protocols and allows concentration on custom waveform design elements. The software engineer is presented with a seamless environment in which to communicate between client and server applications. System developers have a solution that is portable across platforms sharing the same interconnect fabric. The PrismTech Integrated Circuit ORB is part of new generation of products and tools that will revitalize CORBA and the SCA and allow SDR to become ubiquitous in commercial and military applications.