EXPERIENCE REPORT ON THE USE OF CORBA AS THE SOLE MIDDLEWARE SOLUTION IN SCA-BASED SDR ENVIRONMENTS

Fabio Casalino (SELEX Communications, Pomezia (Rome), Italy;
fabio.casalino@selex-comms.com)
Giovanni Middioni (SELEX Communications, Pomezia (Rome), Italy;
giovanni.middioni@selex-comms.com)
Dominick Paniscotti (PrismTech Solutions Americas, Saddle Brook, NJ, USA;
dominick.paniscotti@prismtechusa.com)

ABSTRACT

This paper is an experience report detailing SELEX Communications' findings in the use of CORBA as the sole middleware in SCA-based Software Defined Radios.
This paper describes the different approaches used historically by SCA based systems to connect general-purpose processors to resource constrained processing elements such as FPGAs and DSPs. Following this, focus is turned to asserting that CORBA technology is ready to be fully exploited on SDRs, by describing how CORBA pluggable transports in SCA-based radio systems can be used to reduce the latency and throughput overhead associated with using CORBA's default transport (TCP/IP).

As an experience report, benchmarks will be provided, presenting findings for a custom CORBA transport, based on RapidIO interconnect, used among heterogeneous devices.

1. INTRODUCTION

The Software Communications Architecture (SCA) [1] favors the use of CORBA to connect waveform and platform components to create the connections required to implement data and control flows within an SDR. CORBA is favored as it can support myriad heterogeneous distributed hardware across various bus architectures. However, in order to achieve this, implementations of both CORBA and associated CORBA-transports must exist (or be developed) to support the array of hardware and networking elements used.

Moreover, the SCA allows communication between processing elements not supporting CORBA defining the MHAL (Modem Hardware Abstraction Layer).
If CORBA were available for all the key processing elements (GPP, DSP, FPGA) used in SDRs, then the MHAL communication service would not be needed anymore.

CORBA solutions are today available for most types of GPPs and, contrary to common belief, they are also a reality for DSPs and FPGAs.

2. SCA COMPLIANT CONNECTIVITY: CORBA AND MHAL

2.1 MHAL connectivity

MHAL has been adopted and standardized by the JTRS program to move data to and from modem hardware elements. The MHAL specifies the interfaces to be used and the command, control and data messages that flow across these interfaces. The intent of the MHAL specification is to offer an alternative to CORBA when dealing with processor and bus technologies that have no off the shelf CORBA support.

" Waveforms shall use the MHAL Communications Service for all data and control flowing between software components residing in different CEs (Computational Elements) where at least one CE does not support CORBA ..."[2]

Figure 1 illustrates how MHAL is used in an SCA-based radio. The concept is to create a proxy on a processor supporting CORBA that implements the CORBA interfaces

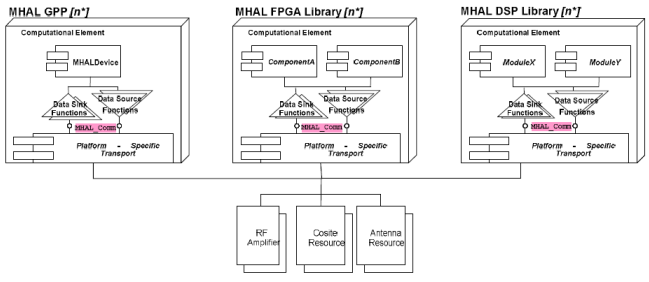

Figure 1 MHAL deployment diagram

Proceeding of the SDR 08 Technical Conference and Product Exposition. Copyright © 2008 SDR Forum. All Rights Reserved

using an SCA Device component. The proxy further implements a proprietary protocol engine that is used to move data to another processor. This processor (typically, non-CORBA capable) adopts a corresponding implementation of the proprietary protocol engine.
The MHAL standard provides an asynchronous variable length messaging service to be used between Data Sources and Data Sinks. This data structure is illustrated in Figure 2.

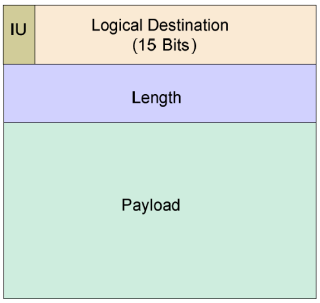

Figure 2 Standard message structure for MHAL

Communicating Data Sources and Sinks may be located in separate CEs or within the same CE.

The MHAL extends CF::Device by adding data/command input and output ports (as shown in Figure 3) and defines a message structure used to pass data to those ports (for transmission to other MHAL compliant processors) rather than defining strongly typed interfaces that express the behavior needed or provided by the port.
In order to isolate assembly waveform component (e.g. BaseBand Component) from MHAL message oriented protocol interface component, adapters are often used. This isolation allows reuse of "true" waveform components but adds additional middleware latencies otherwise not required.

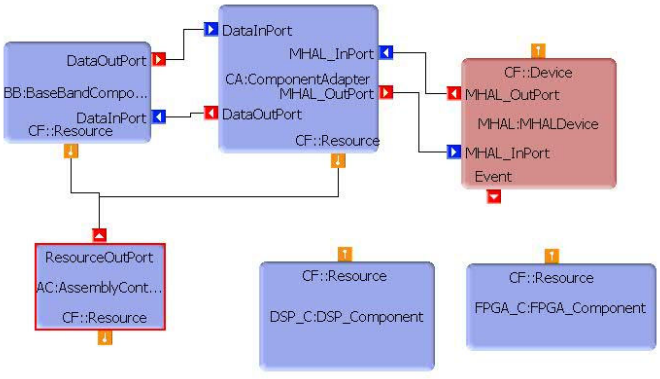

Figure 3: MHAL used in a Waveform

Although MHAL solves the problem of moving data to the modem hardware elements, it has several issues that can not be ignored:

- The interface between components is defined as a simple stream
- The interface semantics are captured in the protocol messages that travel over that stream
- The "on the wire" definition of the protocol is left to each developer to implement (and as such may not agree)
- As the MHAL message ports only serve to act as a conduit to move MHAL message commands and data, each message must be cataloged and the equivalent of an ICD must accompany an MHAL implementation to express the dataflow. This can create significant ambiguity and lead to inconsistencies in implementation (as the message definition is not expressed in a strongly typed language such as IDL or UML). Further compounding matters, this type of implementation limits the ability to transform the interface definitions using automated processes (ie. Using IDL compilers).

2.2 CORBA Connectivity

CORBA middleware can be thought of as a "Logic Bus" which enables distributed processing by allowing software entities (called "CORBA objects") to communicate with each other.

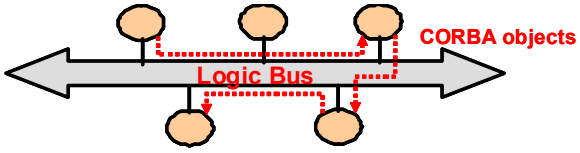

Figure 4: CORBA as a logic bus

The "Logic Bus" is actually based on transport mechanisms that provide a service for reliable delivery of GIOP (General Inter-ORB Protocol) messages exchanged by ORBs.
The use of the CORBA technology offers several advantages over alternative connectivity solutions:

- Object Location Transparency - The client does not need to know where an object is physically located.
- Language Transparency - Client and server can be written in different languages.
- Implementation Transparency - The client is unaware of how objects are implemented.
- Architecture Transparency - The idiosyncrasies of CPU architectures are hidden from both clients and servers (for example, endianess).

Proceeding of the SDR 08 Technical Conference and Product Exposition. Copyright © 2008 SDR Forum. All Rights Reserved

- Operating System Transparency - Client and server are unaffected by each other's operating system.
- Protocol Transparency - Clients and servers do not care about the data link and transport layer.

All these advantages are key factors for considering a CORBA everywhere approach in SDR sets.

3. CORBA EVERYWHERE SOLUTION

As was mentioned earlier, the SCA favors the use of CORBA wherever possible. As CORBA is available for most types of GPPs, then our focus shifts to DSPs and FPGAs.

Contrary to common belief, CORBA implementations for DSPs are available. In fact, support for both the C++ and C languages are available by COTS vendors, and have been for many years. These ORBs also are quite small and designed to fit into the internal memory provided by many DSPs. Typical implementations of CORBA ORBs on DSPs are in the 100kByte memory range or less. Choosing the C CORBA bindings also allows application implementations to remain quite small (often yielding an order of magnitude smaller implementation than for C++).

Which leaves us with CORBA on FPGAs. Here we find a few ways to implement CORBA. The first is to simply purchase an FPGA that includes a microprocessor as part of its architecture and run CORBA on the microprocessor. Another would be to instantiate a microprocessor within an FPGA using an IP core, and run CORBA there. Each of these approaches has drawbacks. These drawbacks fall into categories such as performance (e.g. not being able to clock these processors at high enough speeds), or size (e.g. the IP core taking up large amounts of gates). But we will not focus on these issues, instead we will focus in section 5 on the recent trend of ORB vendors to implement ORBs natively in gates within FPGAs.

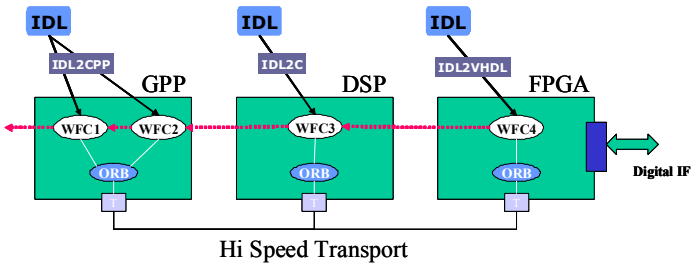

Figure 5: "CORBA Everywhere" scenario

Regardless of how one implements an ORB in an FPGA however, it should be noted than once all SDR processors are CORBA enabled it serves to eliminate the use of adapter patterns (such as MHAL) and:

- Reduce complexity
- Support waveform component location transparency (thereby easing waveform porting).
- Obviate the need for proprietary communication protocols
- Eliminate reimplementation of Device adapters when porting to a new hardware platform
- Likely reduce latency and increase throughput in the waveform communication path.

4. CORBA TRANSPORTS

It has been widely demonstrated that CORBA "latencies" are mainly due to the underlying transport protocol used. The default CORBA Transport is IIOP (Internet Inter-ORB-Protocol), which is based on the TCP/IP protocol stack.
It must be said that TCP/IP is a reliable communication protocol, but in real-time embedded systems it is often not appropriate as the transport infrastructure for CORBA.
The main disadvantages are: TCP/IP software processing is very resource consuming; it is intrinsically non-deterministic, with variable latencies which increase according with the traffic and network complexity; the stack implementation on FPGAs suffers for the limited performances of embedded processors.

In conclusion, IIOP is not well suited in these real-time embedded environments (and software defined radios typically fall into this category). This is the reason why there is the need of an optimized transport protocol.
CORBA middleware can potentially use any data transport mechanism; in fact, ORB vendors allow the plug-in of custom transports, in order to optimize data transfer performance.
The interface between the ORB and the transport can be ORB's proprietary or may use ETF (Extended Transport Framework), which is an OMG standard presently in the finalization phase.
The differences between the ETF interface and a generic ORB proprietary interface are usually not so wide, so the eventual migration to the ETF standard is expected not to require much effort, affecting only the higher layer of the transport.

4.1 "CORBA over RapidIO" transport

Selex Communications has matured a consolidated experience in the development of custom CORBA transports, starting from the "CORBA over VMEbus", the

Proceeding of the SDR 08 Technical Conference and Product Exposition. Copyright © 2008 SDR Forum. All Rights Reserved

transport protocol developed for the "SW Radio Demonstrator" project [3].

Exploiting this important experience, Selex Communications has then developed a CORBA transport for its vehicular SDR, the "CORBA over RapidIO" transport.

The "CORBA over RapidIO" is the transport used by the ORBs hosted on all the devices (GPP, DSP, FPGA) to inter-communicate (i.e. to exchange GIOP messages), through the physical Rapid I/O interconnect.
Rapid I/O is an open-standard defining a packet-switched interconnect, enabling chip-to-chip and board-to-board high performance serial and parallel communications [4].

Three distinct implementations of the "CORBA over RapidIO" transport have been developed for GPP, DSP and FPGA, which are interoperable in order to assure the communication among ORBs executed on heterogeneous Processing Elements. In fact a CORBA object has to be able to invoke the methods of another CORBA object, regardless of where the latter is hosted ("location transparency").

The Selex Comms "CORBA over RapidIO" transport is logically partitioned into three distinct levels: RIO-IOP, RIO-Transport and RIO-Drivers.

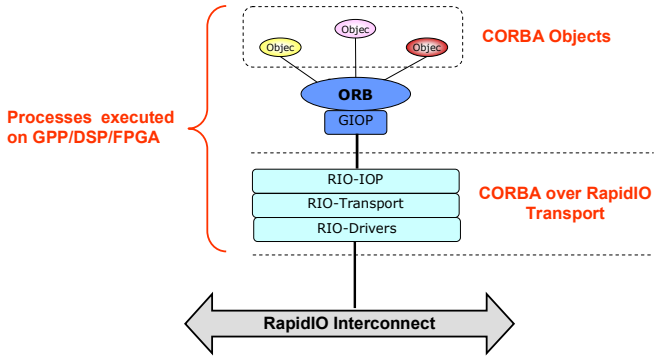

Figure 6: "CORBA over RapidIO" transport layers

RIO-IOP is the transport's 'highest' layer, whose responsibility is the implementation of the interface expected by the ORB to interact, through the GIOP layer, with the transport.
RIO-Transport layer enables connection-oriented communications among processes executed both on the same processing element and on other PEs.
In the RapidIO network each processing element is univocally identified by its "device id", instead at the RIO-Transport level processes on the same PE are discriminated by "ports". RIO-Transport "port" concept is analogous to TCP port.
The RIO-Transport layer provides the means to: 1) create connections; 2) use connections for data transmission and reception; 3) shutdown connections. That is, it allows the management of the entire life-cycle of a connection.
A "connection" can be thought as a virtual "pipe" used by two processes to exchange data in both directions (similarly to a TCP socket).
RIO-Drivers is the transport's 'lowest' layer: it is a medium used by the above RIO-Transport layer to access the RapidIO interconnect.
The RIO-Drivers layer provides an interface to access RapidIO without having to know the involved hardware mechanisms. The 3-layers Rapid I/O stack (Input/Output Logical, Common Transport, 1x Serial Physical) is implemented as an IP Core on FPGA.

Every CORBA transport defines its own "endpoint string" format, where the addressing information is coded. The general format is:

```
transport_name://address_information
```

The endpoint string format of the "CORBA over RapidIO" transport is:

```
rio://<Device id>:<Port number>
```

where: "rio" is the transport name (in fact an ORB can use more transports contemporaneously, each with its name); <Device id> is an integer number (one byte) which univocally identifies a processing element in the Rapid I/O interconnect ; <Port number> is an integer number (one byte) which univocally identifies a process/ORB hosted on a processing element.

In the Selex Communications vehicular SDR, the non real-time (controls, configurations) and real-time data (with the only exception of the high-speed I/Q base band samples traffic to/from DUC/DDC) information streams are transferred by means of the "CORBA over RapidIO" transport.

5. CORBA ON FPGA

As described earlier, this paper will focus on the recent trend of ORB vendors to produce native gate level implementation of ORBs. In this section will be discussed one such implementation, PrismTech's Integrated Circuit ORB (ICO). With such an ORB the need to develop custom proxies on General Purpose Processors (GPPs) and Digital Signal Processors (DSPs) that are solely used to establish

Proceeding of the SDR 08 Technical Conference and Product Exposition. Copyright © 2008 SDR Forum. All Rights Reserved

communication to waveform objects residing within FPGAs can be eliminated. These proxies, are used when designing to Software Defined Radio (SDR) architectures such as the SCA and are meant to increase portability and re-use, but in practice, they tend to increase latency, reduce throughput, and lower re-use.

ICO additionally serves to eliminate the need to embed general purpose processing cores into FPGAs in order to offer software ORB capability. Although a viable approach, it tends to require significant gate count and memory utilization and generally these processing cores cannot be clocked fast enough to deal with the ever-increasing performance requirements of SDR applications.

The ICO engine, is delivered as an IP core, and is responsible for implementing the transfer syntax used in CORBA messages. The engine unmarshals an incoming GIOP stream and extracts header and data fields. Endian conversion is performed on all incoming data based on information in the GIOP message header. In the incoming direction, the engine performs operation named 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 engine generates a reply message. The engine will perform a read operation to an object, if necessary, to obtain data for the reply. When a reply message has been built, the ICO engine transfers the data to the outside world via a FIFO-like interface.

Similar to IDL compilers offered by software ORBs that convert IDL into software languages, an IDL to VHDL compiler accompanies ICO. This compiler is also responsible for generating configuration parameters needed by the ICO engine to perform operation name demultiplexing and data routing described earlier.

Using this IDL compiler, IDL such as that which accompanies the SCA and user defined component interfaces, can be compiled and supported in a native FPGA implementation of an SCA component, thereby eliminating the need for the MHAL in such CORBA-enabled environment.

The hardware developer treats ICO as any other IP interface core. The core can be instantiated in the HDL capture of the FPGA design between the native waveform logic and the system side (transport). The system side of the core appears as a typical FIFO interface. The native side of the core has a simple and open interface to communicate with the waveform logic.

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 host radio elements in an FPGA and still have them be addressable and callable from an SCA-compliant software as though it was an SCA object and not residing in an FPGA.

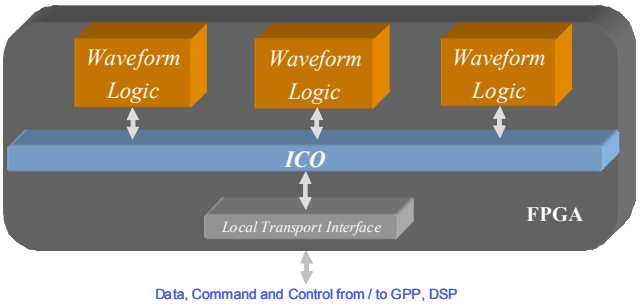

Figure 7: ICO employment

6. CORBA OVER RAPID I/O: PERFORMANCES

The "CORBA over RapidIO" transport performances have been measured in two distinct benchmarks:

1. GPP-to-GPP
2. GPP-to-FPGA

The GPP device is a PowerQUICC II Pro (Freescale), with VxWorks (Wind River) as RTOS and ORBexpress RT (OIS) as ORB.

The FPGA device is a Stratix (Altera) with ICO (PrismTech) as hardware ORB.

In both test facilities, the measurements have been obtained with a CORBA Client invoking on a CORBA Servant the operation `pushPacket()`, which has only one "in" parameter of OctetSequence type (byte buffer to be pushed to the Servant), returning an octet as a result of the operation.

The CORBA Client is hosted on GPP in both the benchmarks, whereas the CORBA Servant is executed on another GPP in the first test facility and on FPGA in the second test facility, as shown in Figure 8 e Figure 9.

Proceeding of the SDR 08 Technical Conference and Product Exposition. Copyright © 2008 SDR Forum. All Rights Reserved

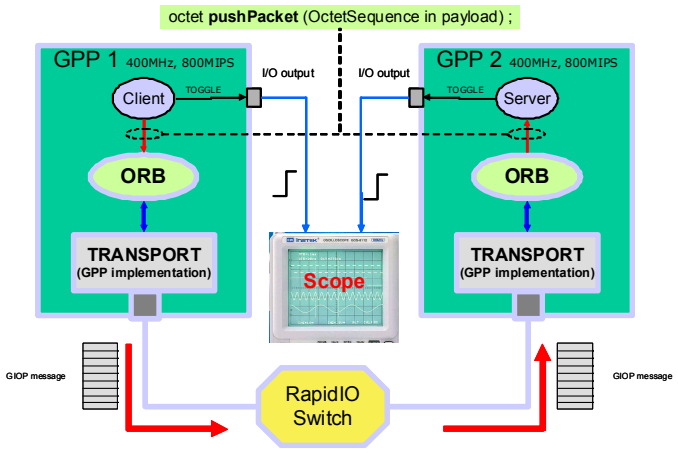

Figure 8: GPP-to-GPP test facility

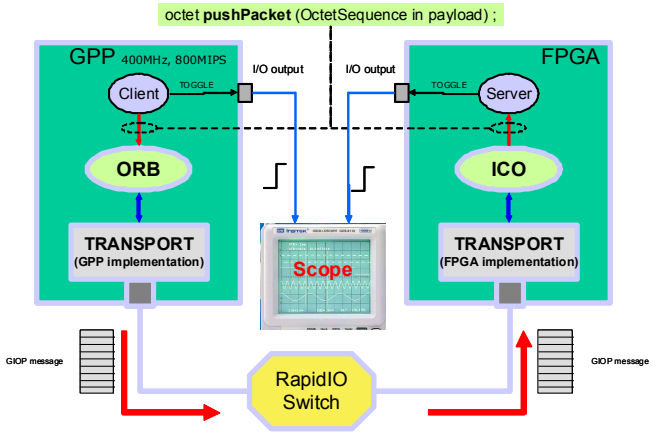

Figure 9: GPP-to-FPGA test facility

The "CORBA over RapidIO" transport is used by both ORBs (ORBexpress RT on GPP and ICO on FPGA) to exchange GIOP messages carrying CORBA requests and replies.

The time latency between the instant just before the pushPacket() method invocation in the Client side and the instant when the execution of the method actually starts in the Server side has been measured using hardware tools.
Two distinct physical signals (coming from the GPP for the Client side and from the other GPP or the FPGA for the Server side) are asserted in correspondence of each event. The two digital signals are displayed in an oscilloscope, in order to measure the time interval between their up edges.
This procedure is repeated at least ten times in order to get a statistically valid average from the obtained values (which actually show a very little variance), representing the time latency for the delivery of a byte buffer of a defined size.

The entire process is in turn repeated for different values of the OctetSequence parameter length, doubling the payload size starting from 32 bytes up to 4 Kbytes.

The "CORBA over RapidIO" performances measured through the two described benchmarks are reported in Table 1 and displayed in Figure 10 e Figure 11.

Packet Size (Bytes)	GPP-to-GPP Latency (us)	GPP-to-FPGA Latency (us)
32	114	51
64	122	51
128	121	55
256	118	48
512	121	60
1024	132	73
2048	152	92
4096	180	148

Table 1 "CORBA over RapidIO" Transport performances

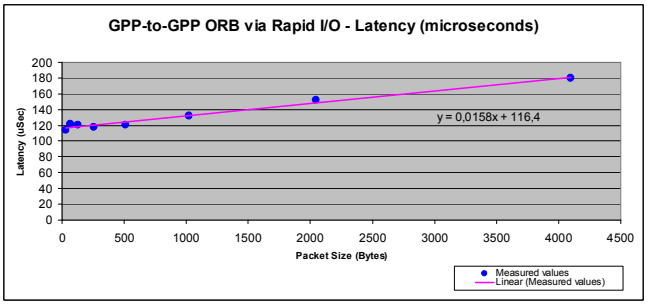

Figure 10 GPP-to-GPP performances

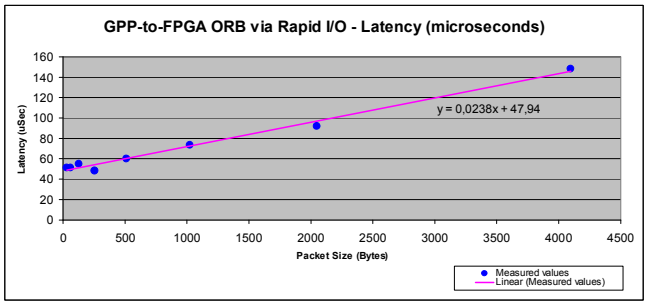

Figure 11 GPP-to-FPGA performances

Proceeding of the SDR 08 Technical Conference and Product Exposition. Copyright © 2008 SDR Forum. All Rights Reserved

Figure 10 and Figure 11 clearly show the high linearity of all measured delays: the dots refer to actual measured values and the straight line represents the best fit (trend line).

7. CONCLUSION

This paper describes how a 'CORBA everywhere' solution, e.g. a solution where CORBA is used also in resource-constrained processing elements such as FPGAs and DSPs, is today feasible and actually represents a preferable choice than MHAL.

The development of an adequate transport, that can be plugged into ORB implementations, is the key factor to achieve best performances in terms of latency and throughput.

In particular, benchmarks regarding GIOP message transfers between an ORB on a GPP and an ORB on FPGA (ICO), show the applicability of CORBA technology in such environments.

8. REFERENCES

[1] SOFTWARE COMMUNICATIONS ARCHITECTURE SPECIFICATION Version 2.2.2 - FINAL / 15 May 2006

[2] Joint Tactical Radio System (JTRS) Standard Modem Hardware Abstraction Layer Application Program Interface (API) -Version: 2.11.1 - 02 May 2007

[3] Giovanni Middioni, "CORBA over VMEbus Transport for Software Defined Radios", 2005

[4] http://www.rapidio.org/home

Proceeding of the SDR 08 Technical Conference and Product Exposition. Copyright © 2008 SDR Forum. All Rights Reserved

Proceeding of the SDR 08 Technical Conference and Product Exposition. Copyright © 2008 SDR Forum. All Rights Reserved