PrismTech

Powering Netcentricity

Utilizing the Latest IP Technology for FPGAs to Achieve SDR Architectural Consistency

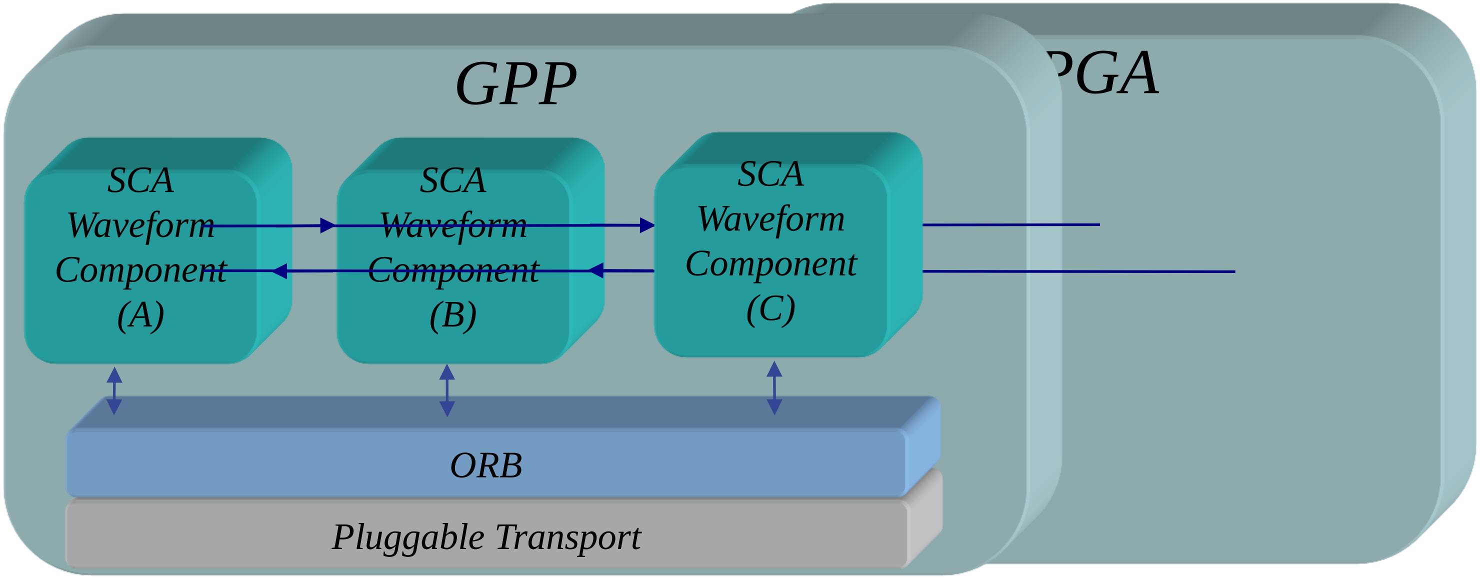
June 24th 2011 – Andrew Foster, Spectra Product Manager

Agenda

1. Objectives
2. SCA Device Model and MHAL
3. CORBA Everywhere
4. Spectra IP Core ORB
   - Architecture
   - IDL to VHDL Mapping
   - Design Flow & Example
5. ICO v2
   - Key Features
   - On-going Work
   - ICO v2 Performance and Footprint
6. Case study
7. Future Standardization
8. Summary

Objectives

- To seamlessly integrate waveform logic running on the FPGA… while still maintaining SCA compliance



GPP
PGA
SCA Waveform Component (A)
SCA Waveform Component (B)
SCA Waveform Component (C)
ORB
Pluggable Transport

Copyright PrismTech 2011

PrismTech

Powering Netcentricity

SCA Device Model and MAL

SCA Processor Coverage

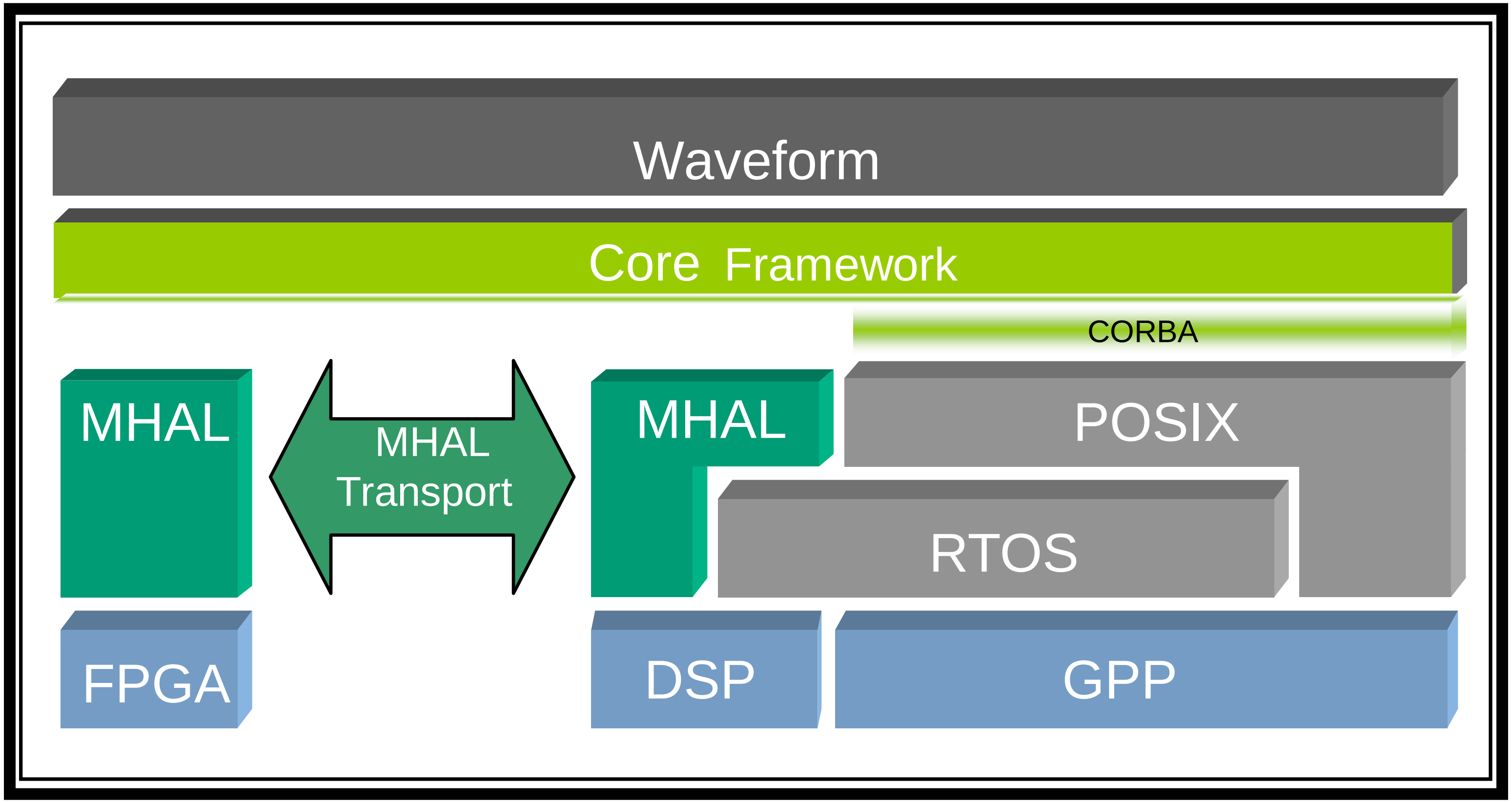
- “..*waveforms shall use the MHAL Communications Service for all data and control flowing between software components residing in different Computational Elements where at least one CE does not support CORBA…”*

  Reference: *Joint Tactical Radio System (JTRS) Standard Modem Hardware Abstraction Layer Application Program Interface (API) Version: 2.13, 29th June 2010*

- PrismTech interpretation….CORBA Everywhere would be optimal, if this were available……

- This is about to change – next revision of the SCA “SCA Next” will formally adopt a CORBA profile for DSPs and work ongoing for a lightweight CORBA profile for FPGAs

Current SCA Device Model

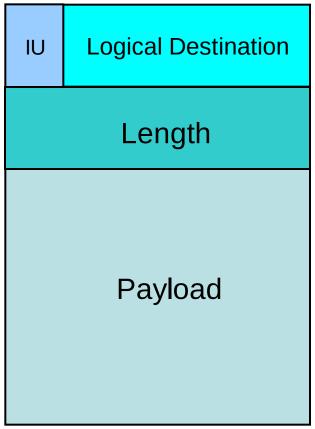


MHAL approach provides a degree of portability, however, the format and content of messages sent to the MHAL components is not standardised and must be written by each waveform developer

SCA/FPGA Connectivity – MHAL

- MHAL has been adopted and standardised by the JTRS program to move data to and from modem hardware
- MHAL interfaces are used for command, control and data messages
- Offers an alternative to CORBA when dealing with processor and bus technologies with no off the shelf CORBA support

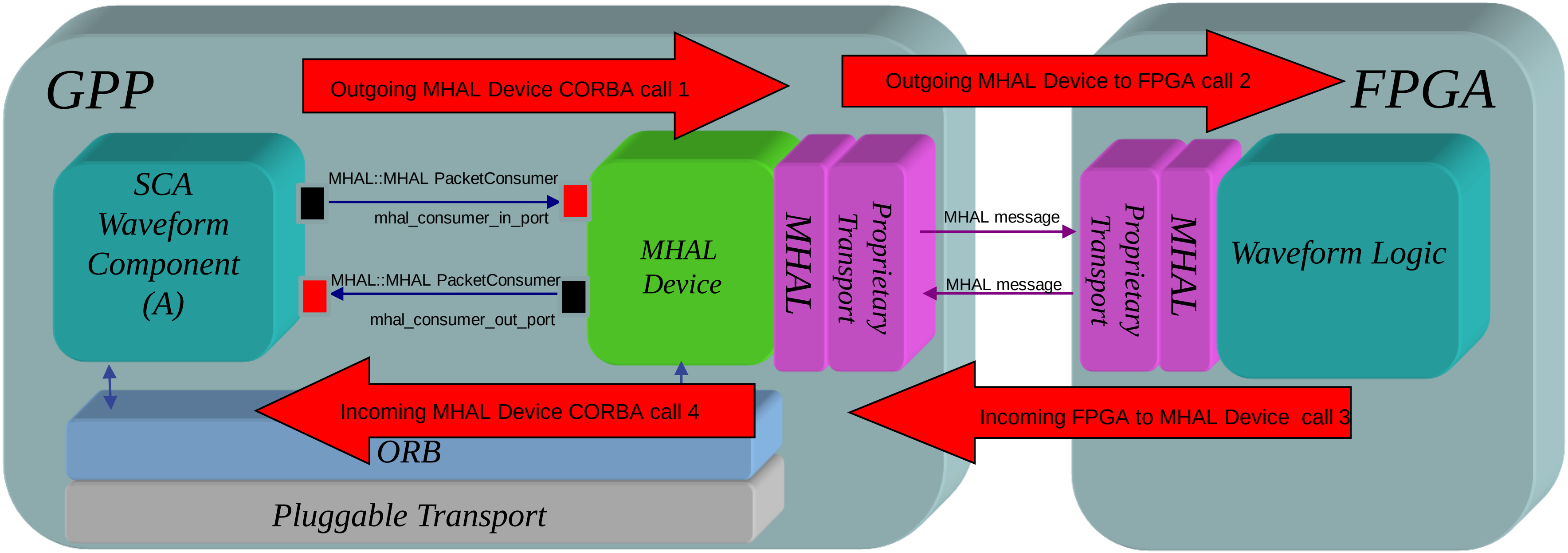
- Issues:
  - Interface between components is defined as a simple stream
  - The "on the wire" definition of the protocol is left to each developer
  - Interface semantics are captured in the protocol messages that travel over the stream
  - In order to isolate an assembly waveform component from the MHAL message oriented interface an "adaptor" or proxy" is often used



Standard
MHAL Message
Structure

1st Generation SCA/FPGA Connectivity – MHAL

- Attempts to implement MHAL have resulted in added complexity for the radio developer and increased overall system latency
- Using MHAL Device approach requires a double call hop for both outgoing (sink) and incoming (source) calls – 4 calls instead of optimal two


GPP
Outgoing MHAL Device CORBA call 1
Outgoing MHAL Device to FPGA call 2
FPGA
SCA Waveform Component (A)
MHAL::MHAL PacketConsumer
mhal_consumer_in_port
MHAL::MHAL PacketConsumer
mhal_consumer_out_port
MHAL Device
MHAL
Proprietary Transport
MHAL message
MHAL message
Proprietary Transport
MHAL
Waveform Logic
Incoming MHAL Device CORBA call 4
Incoming FPGA to MHAL Device call 3
ORB
Pluggable Transport

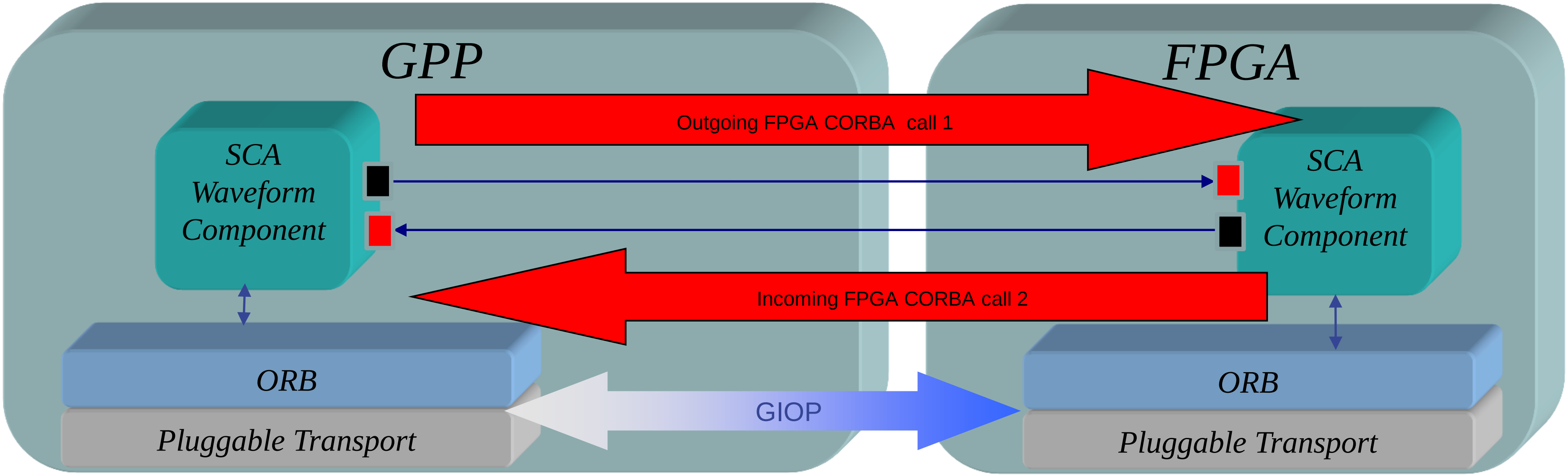
🟥 = SCA Provides Port

⬛ = SCA Uses Port

PRISMTECH

What if…

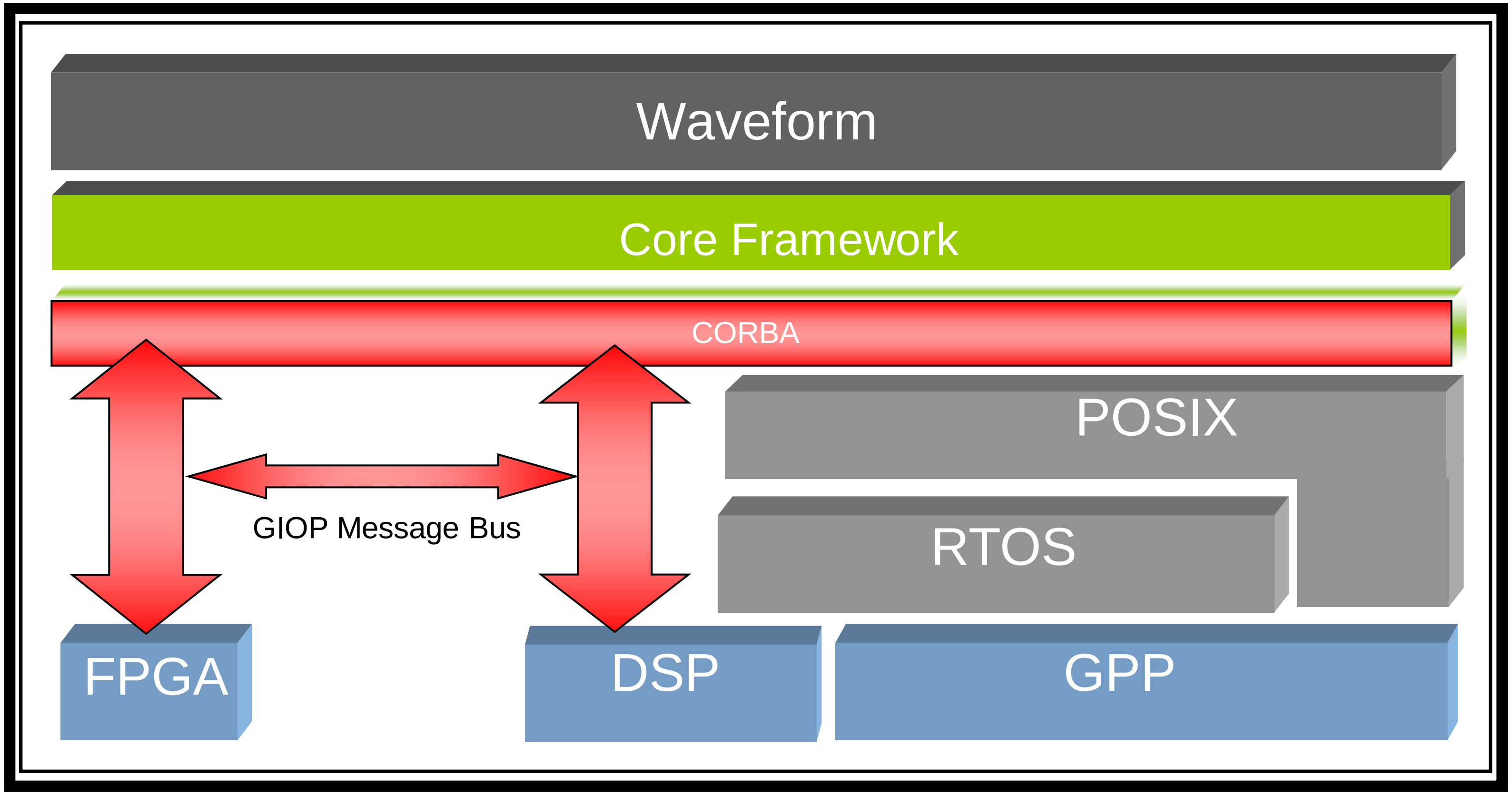
- Using an ORB, SCA compliance is maintained and overhead is reduced
- Now only requires single direct call for both outgoing and incoming messages – 2 calls instead of previous 4



🟥 = SCA Provides Port

⬛ = SCA Uses Port

Proposed Problem Solution



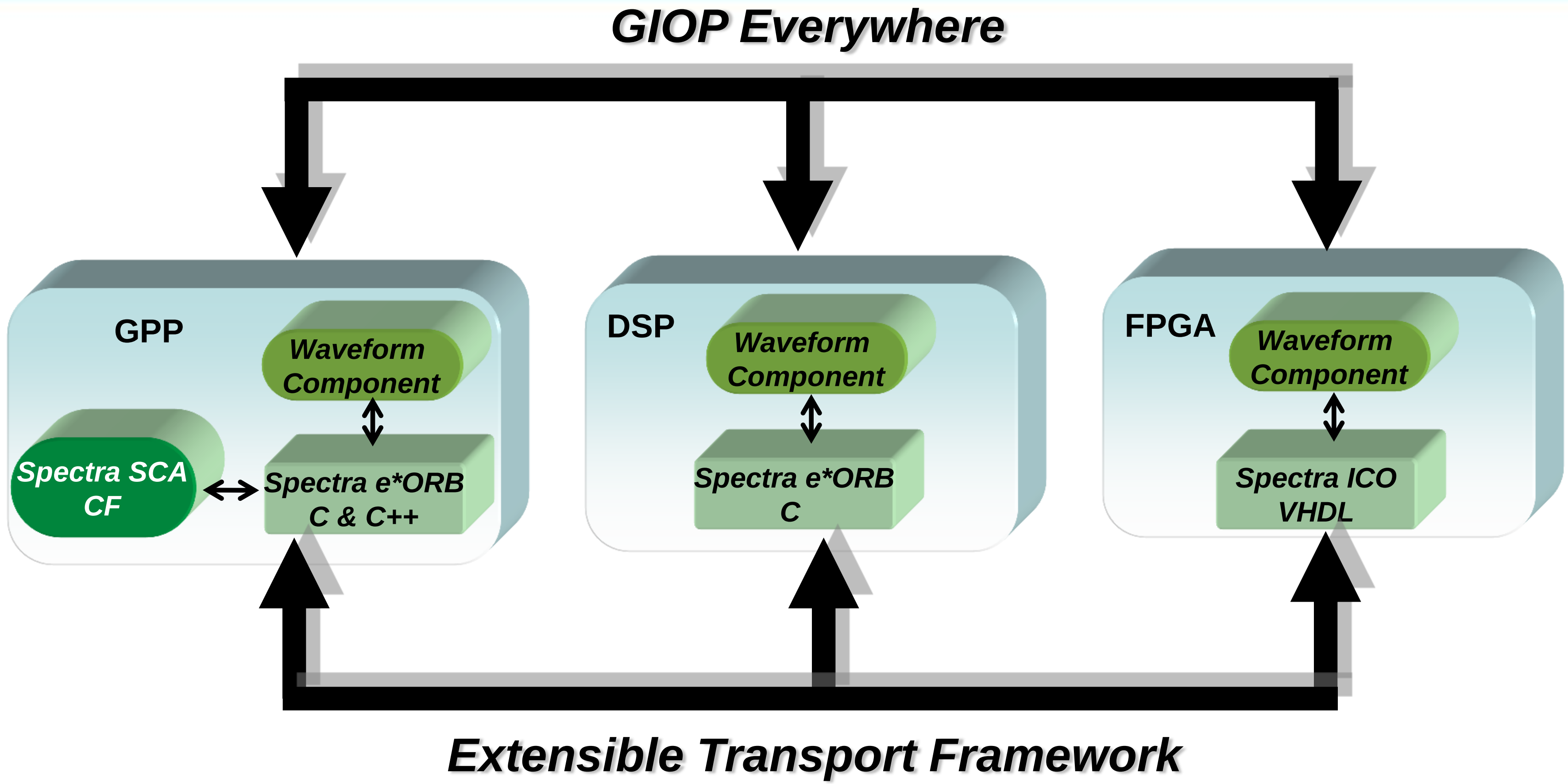
Standardized CORBA interfaces across signal processing chain

PRISMTECH

Powering Netcentricity

CORBA Everywhere

2nd Generation SCA/FPGA Connectivity – CORBA Everywhere



By leveraging CORBA a standards-based, high-performance, low-footprint, fully-interoperable COTS middleware solution that can be deployed across multiple processor types, including GPP, DSP, & FPGA environments

Benefits of CORBA Everywhere

- Once all SDR processors (GPP, DSP and FPGA) are CORBA enabled, a number of potential benefits can be realised:
  - Reduce overall system complexity and improve time-to-market for new waveform applications and also legacy waveform porting
  - Support waveform component location transparency making it much easier to re-locate waveform components across processors
  - Eliminate the need for proprietary communication protocols reducing complexity and improving waveform portability
  - Remove the need to use adaptor patterns in combination MHAL, therefore reducing communication latency and improving throughput

PRISMTECH

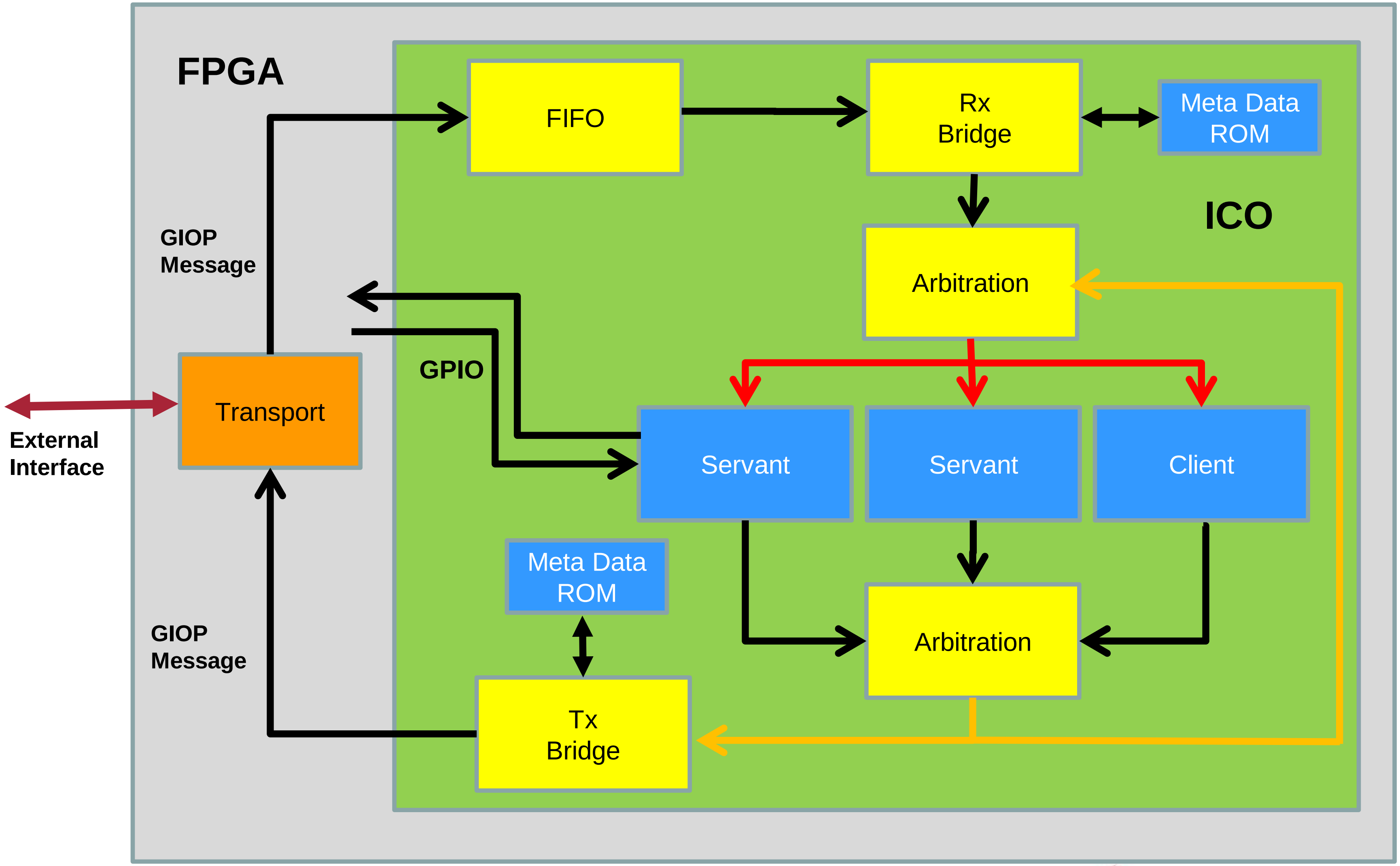
Performance Critical Middleware

Spectra IP Core ORB (ICO)

ICO Joint Industry Project (JIP)

- Spectra IP Core ORB (ICO) is a second generation high performance messaging solution for FPGAs based on CORBA's GIOP protocol – a hardware ORB
- Development jointly with key partners including:
  - **PrismTech** – lead implementer and the commercial organisation that will take ICO to market
  - **Selex** – providing end user requirements, use cases and technical validation – end user for ICO
  - **Altera** – providing FPGA hardware and development tools support + FPGA expertise
- Development started in summer 2009
- ICO V2 released in spring 2011

ICO Architecture



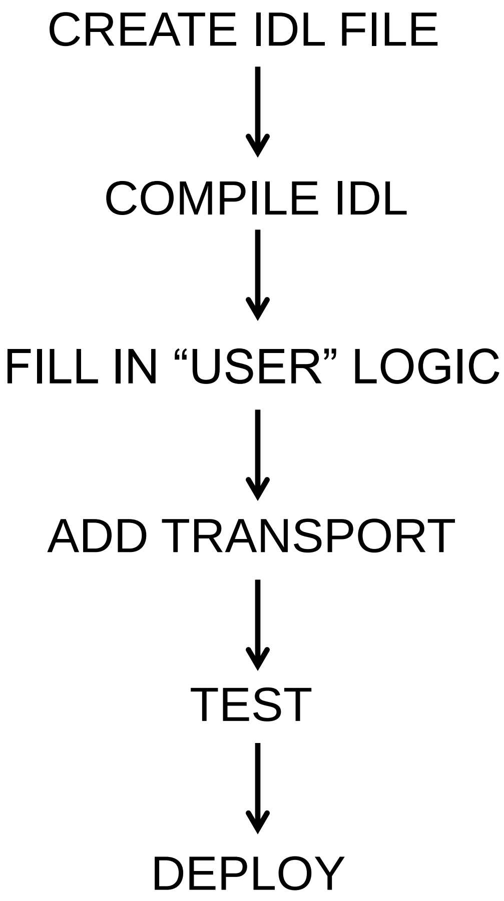
IDL to VHDL Mapping

- Direct mapping of CORBA primitive types to VHDL
- The mapping requires that a bus-based architecture is used
- The bus must support the concept of data and addressing
- The mapping defines a protocol called Bus Interoperability Protocol (BIOP)
- GIOP can be converted to BIOP and vice-versa

Bus Interoperability Protocol - BIOP

- Bus-based communication mechanism
- The protocol is designed to support requests and replies between hardware entities
- Data exchanged in the form of messages
- Three message types – Request, One-way Request and Reply
- Messages consist of a header, and optionally message data
- Request/reply data is placed on the bus with an address offset from the target entity's base address
- The offset is a constant generated according to the IDL-VHDL language mapping
- Operation parameter and reply data passed in GIOP CDR encoding order

Design Flow



CREATE IDL FILE

COMPILE IDL

FILL IN “USER” LOGIC

ADD TRANSPORT

TEST

DEPLOY

Example IDL

```
module AnalogDigital
{
   interface DAC
   {
      void send_data(in unsigned longval);
   };

   interface ADC
   {
      long read_data ();
   };
};
```

Example servant –DAC Interafce

```
case AddressOffset(adr_i(AddressBusLow'range)) is
               when ADC_read_data_request =>
                 v.request_id := unsigned(dat_i);

               when ADC_read_data_replyaddr =>
                 v.reply_addr := dat_i(AddressBusHigh'range);

               when ADC_read_data_request_end =>
                 case r.reply is
                   when normal =>
                     v.state := ADC_read_data_reply_state;
                   when others =>
                     v.state := request_state;
                 end case;

               when DAC_send_data_request =>
                 v.request_id := unsigned(dat_i);

               when DAC_send_data_replyaddr =>
                 v.reply_addr := dat_i(AddressBusHigh'range);

               when DAC_send_data_val =>
                 -- Modify the following line as needed.

                 null;
               when DAC_send_data_request_end =>
                 case r.reply is
                   when normal =>
                     v.state := DAC_send_data_reply_state;
                   when others =>
                     v.state := request_state;
                 end case;

               when others =>
                 null;
             end case;
```

Example servant – ADC Interface

```
when ADC_read_data_return_state =>
           v.address := mkaddr(r.reply_addr, ADC_read_data_return);
           if bus_grant = '1' then
             v.state := ADC_read_data_reply_end_state;
             -- Modify the following line as needed.
             v.data := (others => '0');
             v.we := '1';
           else
             null;
           end if;
end process;
```

PrismTech

Powering Netcentricity

ICO v2

ICO v2 Key Features

- ICO v2.0 provides support for the following key features:
  - Supports GIOP version 1.0 protocol
  - Processes incoming CORBA requests
    - One way operations
    - Two way operations
  - Support for CORBA clients and servers
    - Clients can be internal to the FPGA written in VHDL or external to FPGA(e.g., on a GPP or DSP) implemented by a conventional software application
    - Servants implemented on FPGA in VHDL
    - No arbitrary restriction on the number of clients and servers that can be supported on the FPGA

ICO v2 Key Features

- IDL compiler support
  - Supports IDL to VHDL language mapping and will auto generate VHDL equivalent of CORBA stubs and skeletons allowing ICO to be easily connected to servants implementing waveform logic
    - Based on CORBA 3 grammar, but only supporting a subset of data types and constructs
      - Simple data types - Char, Octet, Boolean, Short, Unsigned Short, Long, Unsigned Long, Long Long, Unsigned Long Long, String
      - Enumerated Types
      - Complex data types
        - Struct
        - Sequence
      - CORBA exceptions support
        - User exceptions
        - System Exceptions

ICO v2 Key Features

- Pluggable and open transport interface allows user-defined custom transports to be plugged into ICO
- Written in pure VHDL and completely portable across FPGA devices
- Available on both Altera and Xilinx FPGAs, including:

FPGA Model	Board	Tool Chain
Altera Stratix II	Altera Eval	Modelsim ALTERA 6.5b + Quartus II v9.1
Altera Stratix III	Altera Eval	Modelsim ALTERA 6.5b + Quartus II v9.1
Altera Stratix IV	Altera Eval	Modelsim ALTERA 6.5b + Quartus II v9.1
Altera Cyclone II	Altera Eval	Modelsim ALTERA 6.5b + Quartus II v9.1
Altera Cycone III	Altera Eval	Modelsim ALTERA 6.5b + Quartus II v9.1
Xilinx Spartan 3	Xilinx Eval	ISE 13
Xilinx Spartan 6	Monsoon	ISE 13
Xilinx Virtex Pro IV	Pro 4600	Modelsim Xilinx Edition III + ISE 12

🟨 = available now

□ = planned

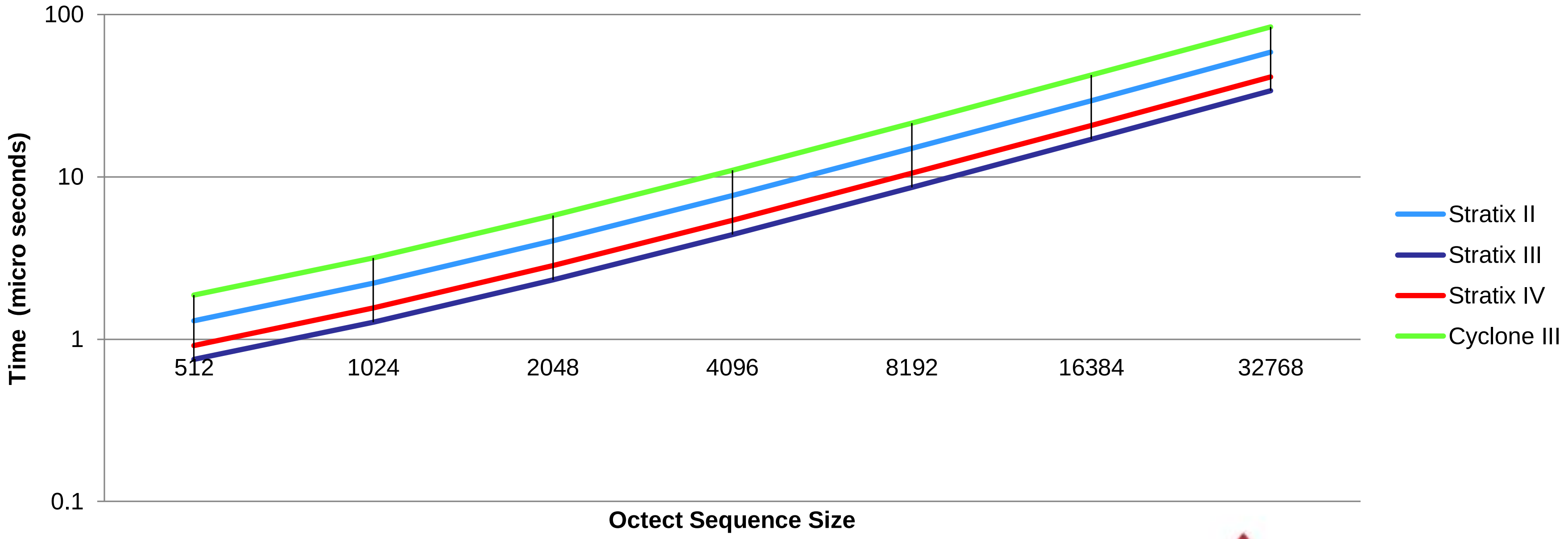
On-going Work

- ICO v2.1 will add support for the additional features and capabilities:
  - Full outgoing CORBA request support (e.g., to allow client requests from FPGA-GPP or FPGA-DSP)
  - Additional IDL data types:
    - Object (to support dynamic endpoints and passing of object references)
    - Arrays
    - Any (of basic types)
      - Primitive values
      - Strings
      - Sequence of primitive values
    - Unions
  - ICO v2.1 targeted for release in summer 2011
- Spectra CX v3.3 will support graphical modelling and code generation for DSP and FPGA components – target release date late summer 2011

ICO v2 Performance

Request Type	Octet Sequence Size	ICO v1 Stratix II Time (µS)	ICO v2 Stratix II Time (µS)	ICO v1 Stratix III Time (µS)	ICO v2 Stratix III Time (µS)	ICO v1 Stratix IV Time (µS)	ICO v2 Stratix IV Time (µS)	ICO v1 Cyclone III Time (µS)	ICO v2 Cyclone III Time (µS)
IN	512	4.1154	1.2993	2.6714	0.7503	3.4656	0.915	7.1478	1.8666
IN	1024	6.7488	2.2081	4.3808	1.2751	5.6832	1.555	11.7216	3.1722
IN	2048	12.8706	4.0257	8.3546	2.3247	10.8384	2.835	22.3542	5.7834
IN	4096	24.5442	7.6609	15.9322	4.4239	20.6688	5.395	42.6294	11.0058
IN	8192	47.8914	14.9313	31.0874	8.6223	40.3296	10.515	83.1798	21.4506
IN	16384	94.5858	29.4721	61.3978	17.0191	79.6512	20.755	164.2806	42.3402
IN	32768	187.9746	58.5537	122.0186	33.8127	158.2944	41.235	326.4822	84.1194

ICO v2 Message Processing Times



ICO v2 Footprint

	ICO v1 Stratix II	ICO v2 Stratix II	ICO v1 Stratix III	ICO v2 Stratix III	ICO v1 Stratix IV	ICO v2 Stratix IV
Logic Utilization						
Combinational ALUTs	2422	1812	2609	1812	2393	1812
Dedicated logic registers	2176	1531	2289	1531	2201	1531
Total Registers	2176	1531	2289	1531	2201	1531
Total pins	22	137	22	137	22	137
Total virtual pins	0	0	0	0	0	0
DSP block 9-bit elements	0	0	0	0	0	0
Total PLLs	0	0	0	0	0	0
Total DLLs	0	0	0	0	0	0

	ICO v1 Cyclone III	ICO v2 Cyclone III
Total Logic Elements	4925	3429
Total Combinational Functions	3576	2639
Dedicated logic registers	2457	1532
Total Registers	2457	1532
Total pins	22	137
Total virtual pins	0	0
DSP block 9-bit elements	0	0
Embedded Multiplier 9-bit elements	0	0

```
interface Performance
{
   typedef sequence<octet> OctetSeq;

   void setLength (in long seqLength);
   void testOctetSeqIn (in OctetSeq inSeq);
   void testOctetSeqOut (out OctetSeq outSeq);
   void testOctetSeqInout (inout OctetSeq inoutSeq);
   OctetSeq testOctetSeqRet ();

   void shutdown ();
};
```

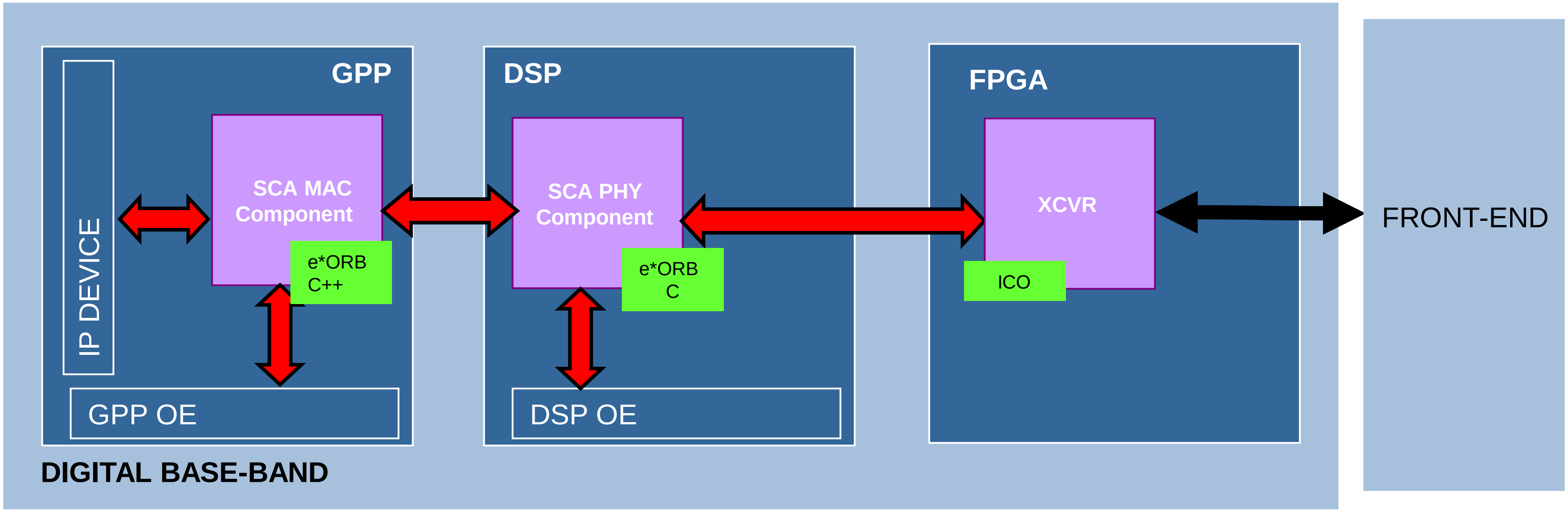
PrismTech

Powering Netcentricity

Case Study

Euler Project

- Euler - European Software Defined Radio for Wireless In Joint Security Operations
- Major European project that will focus on SDR technologies for emergency and security operations
- PrismTech are developing a base station to support the Euler Wimax derived waveform
- CORBA everywhere is being deployed across signal processing chain on base station – ICO is providing a CORBA interface between the modem and XCVR hardware

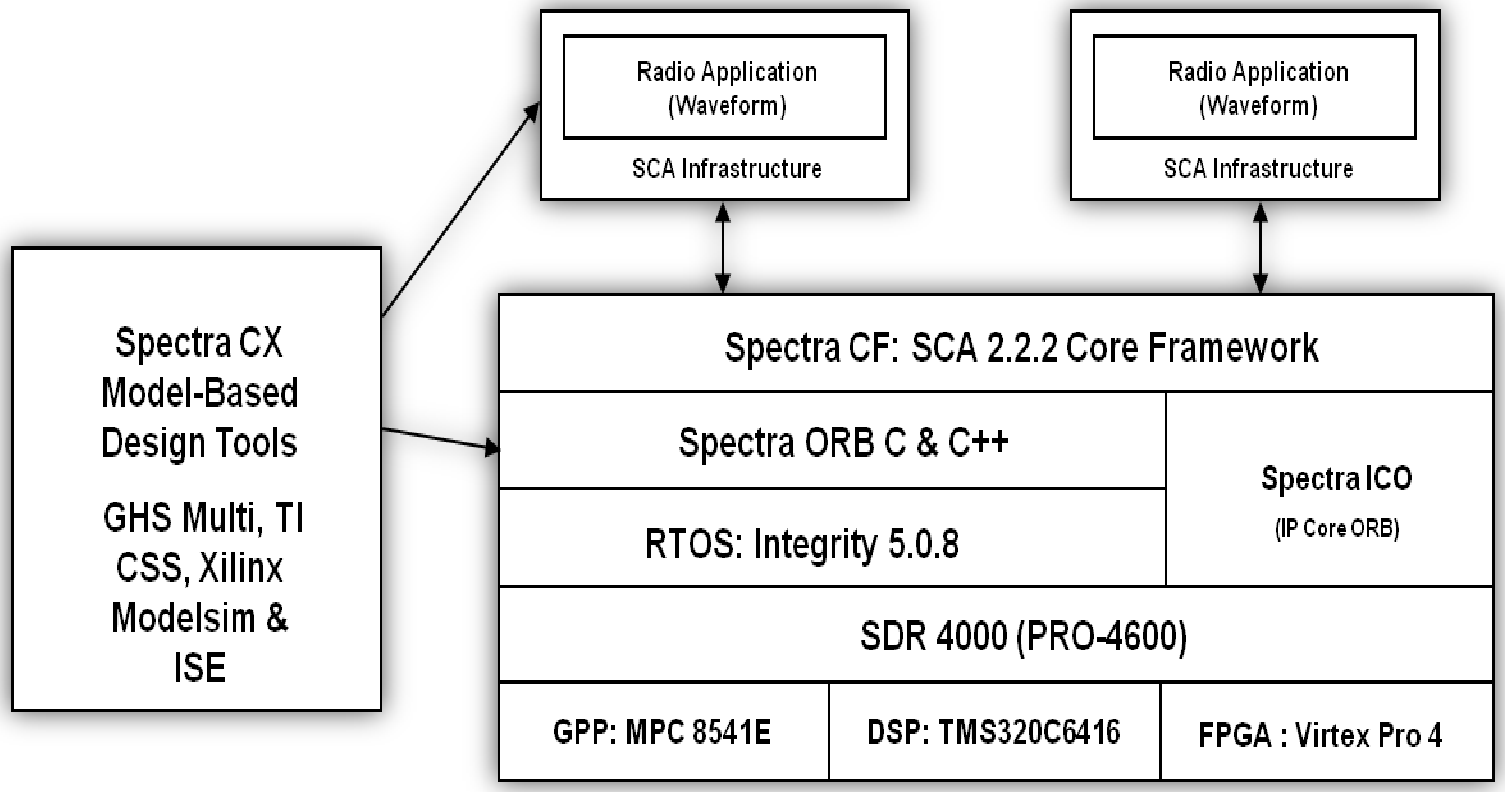


🟥 =CORBA Communications

Euler SDR4000 Development Platform Configuration

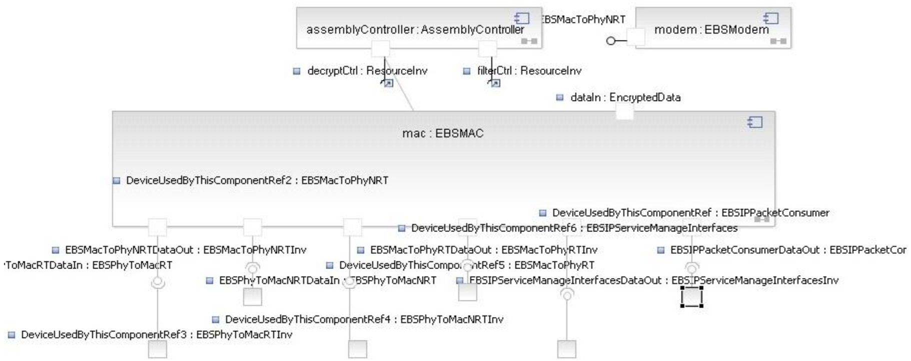
Host Development Tools

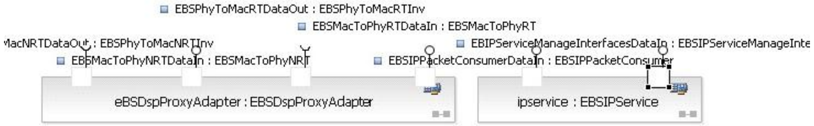
Target Radio Platform



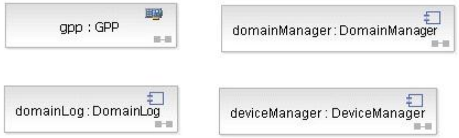
Euler SCA Models

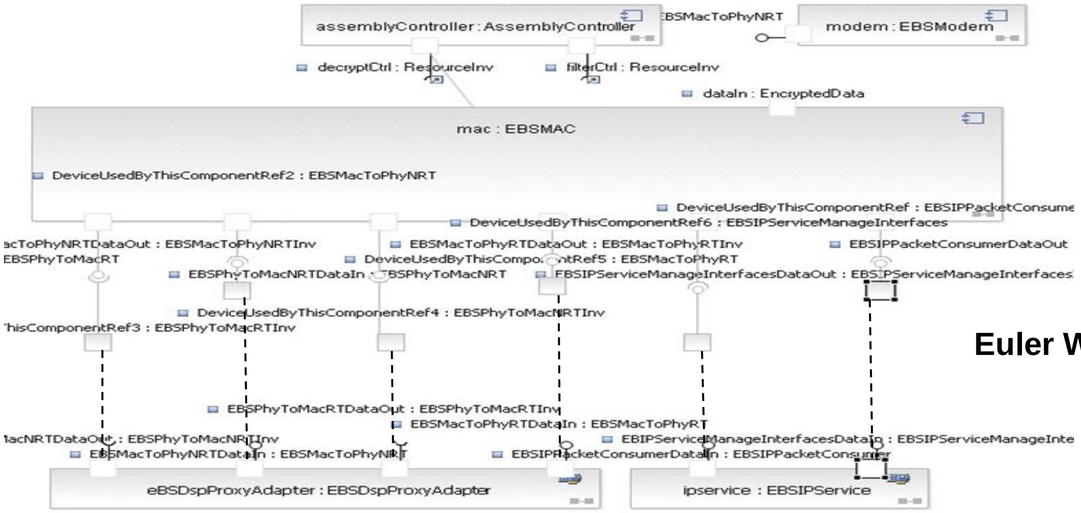
Euler Waveform (EWF) Model


assemblyController : AssemblyController
EBSMacToPhyNRT
modem : EBSModem
decryptCtrl : ResourceInv
filterCtrl : ResourceInv
dataIn : EncryptedData
mac : EBSMAC
DeviceUsedByThisComponentRef2 : EBSMacToPhyNRT
DeviceUsedByThisComponentRef : EBSIPPacketConsumer
DeviceUsedByThisComponentRef6 : EBSIPServiceManageInterfaces
EBSMacToPhyNRTDataOut : EBSMacToPhyNRTInv
EBSMacToPhyRTDataOut : EBSMacToPhyRTInv
EBSIPPacketConsumerDataOut : EBSIPPacketCor
ToMacRTDataIn : EBSPhyToMacRT
DeviceUsedByThisComponentRef5 : EBSMacToPhyRT
EBSPhyToMacNRTDataIn : EBSPhyToMacNRT
EBSIPServiceManageInterfacesDataOut : EBSIPServiceManageInterfacesInv
DeviceUsedByThisComponentRef4 : EBSPhyToMacNRTInv
DeviceUsedByThisComponentRef3 : EBSPhyToMacRTInv


EBSPhyToMacRTDataOut : EBSPhyToMacRTInv
EBSMacToPhyRTDataIn : EBSMacToPhyRT
MacNRTDataOut : EBSPhyToMacNRTInv
EBIPServiceManageInterfacesDataIn : EBSIPServiceManageInte
EBSMacToPhyNRTDataIn : EBSMacToPhyNRT
EBSIPPacketConsumerDataIn : EBSIPPacketConsumer
eBSDspProxyAdapter : EBSDspProxyAdapter
ipservice : EBSIPService

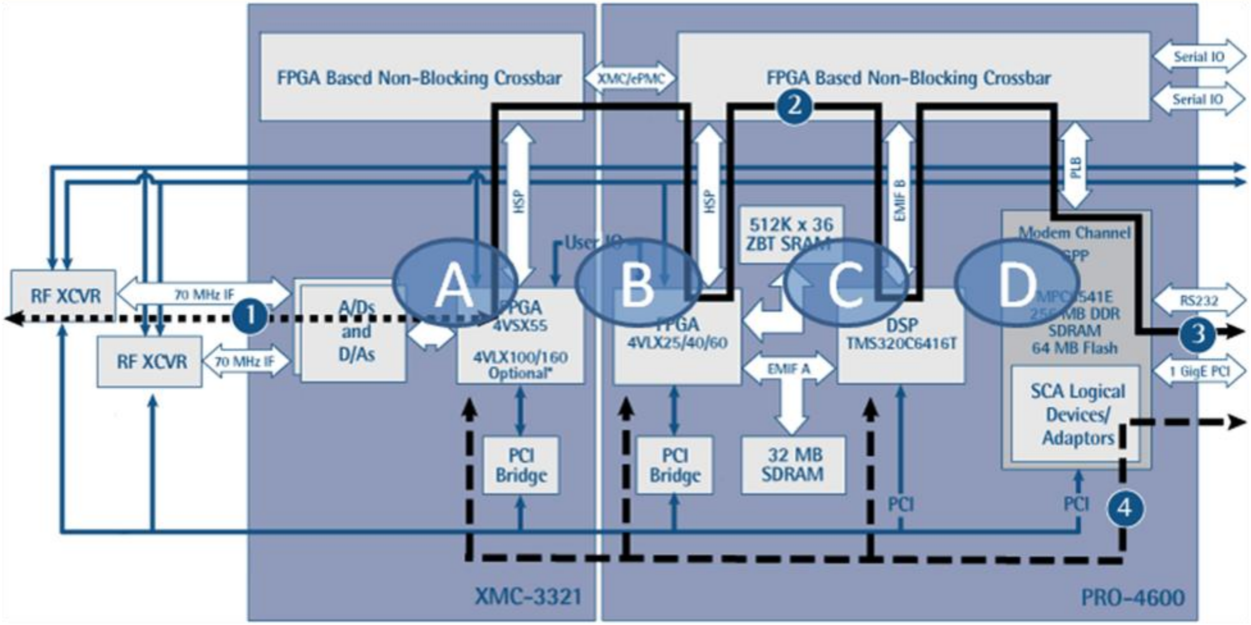
Euler SDR400 Platform Model




assemblyController : AssemblyController
EBSMacToPhyNRT
modem : EBSModem
decryptCtrl : ResourceInv
filterCtrl : ResourceInv
dataIn : EncryptedData
mac : EBSMAC
DeviceUsedByThisComponentRef2 : EBSMacToPhyNRT
DeviceUsedByThisComponentRef : EBSIPPacketConsume
DeviceUsedByThisComponentRef6 : EBSIPServiceManageInterfaces
acToPhyNRTDataOut : EBSMacToPhyNRTInv
EBSMacToPhyRTDataOut : EBSMacToPhyRTInv
EBSIPPacketConsumerDataOut
EBSPhyToMacRT
DeviceUsedByThisComponentRef5 : EBSMacToPhyRT
EBSPhyToMacNRTDataIn : EBSPhyToMacNRT
EBSIPServiceManageInterfacesDataOut : EBSIPServiceManageInterfaces
DeviceUsedByThisComponentRef4 : EBSPhyToMacNRTInv
hisComponentRef3 : EBSPhyToMacRTInv
EBSPhyToMacRTDataOut : EBSPhyToMacRTInv
EBSMacToPhyRTDataIn : EBSMacToPhyRT
lacNRTDataOut : EBSPhyToMacNRTInv
EBIPServiceManageInterfacesDataIn : EBSIPServiceManageInte
EBSMacToPhyNRTDataIn : EBSMacToPhyNRT
EBSIPPacketConsumerDataIn : EBSIPPacketConsumer
eBSDspProxyAdapter : EBSDspProxyAdapter
ipservice : EBSIPService

Euler Waveform Assembly Model

SDR4000 Radio Hardware Platform Devices



Device (Processing Unit)		Rating		Operating System
A	User FPGA – Xilinx Virtex-4 XC4VSX55 (Dalton) FPGA	Array	128 x 48	n/a
		Logic cells	55296	
		Block RAM (kb)	5760	
		CLB Slices	24576	
		RAM (kb)	384	
		DSP Slices	512	
B	User FPGA – Xilinx Virtex-4 XC4VLX60 (Tesla) FPGA	Array	128 x 52	n/a
		Logic cells	59904	
		Block RAM (kb)	2880	
		CLB Slices	26624	
		RAM (kb)	416	
		DSP Slices	64	
C	Instruments DSP TMS320-C6416T	4800 MIPS at 600MHz, Fixed Point , 32 MB SDRAM (shared with FPGA "B")		DSP-BIOS (TI) Code Composer Studio v3.1
D	Freescale MPC8541E PowerQUICC III	1850 MIPS at 800 MHz (estimated Dhrystone 2.1), 256 MB of 264 MHz DDR SDRAM and 64 MB Flash		Greenhills Integrity v508

Euler Transceiver Interface Details

- Euler XCRV API based on SDR Forum Transceiver Facility Specification v1.0.0

Standard XCVR IDL

```
interface transmitcontrol
  {
    void createtransmitcycleprofile (
      in time           requestedtransmitstarttime,
      in time           requestedtramsmitstoptime,
      in unsigned short requestedpresetid,
      in frequency      requestedcarrierfrequency,
      in analoguepower  requestednorminalrfpower);
  };


interface transmitdatapush
  {
    void pushbbsamplestx (
      in bbpacket thepushedpacket,
      in boolean  endofburst);
  };

interface receivecontrol
  {
    void createreceivecycleprofile (
      in time           requestedreceivestarttime,
      in time           requestedreceivestoptime,
      in unsigned long  requestedpacketsize,
      in unsigned short requestedpresetid,
      in frequency      requestedcarrierfrequency);
   };

interface receivedatapush
  {
    void pushbbsamplesrx (
      in bbpacket thepushedpacket,
      in boolean  endofburst);

  };
```

SDR4000 XCVR IDL

```
interface Transmit
  {
      oneway void createTransmitCycleProfile(
        in Time requestedTransmitStartTime,
        in Time requestedTransmitStopTime) ;

      oneway void pushBBSamplesTx(
        in BBPacket thePushedPacket) ;
  };

interface Receive
  {
      oneway void createReceiveCycleProfile(
        in Time requestedReceiveStartTime,
        in Time requestedReceiveStopTime ) ;

      oneway void pushBBSamplesRx(
        in BBPacket thePushedPacket ) ;

  };
```

Standard IDL modified to minimize call latency and to remove unused parameters

Future Standardization

- SCA 2.2.2 assumes that an FPGA is a non-CORBA capable Computational Element (CE)
- SCA Next will standardize a CORBA profile for DSP (Lightweight profile based on CORBA/e) reflecting availability of lightweight ORB implementations for DSPs
- Through the SCA Next working group at the WINNF the ESSOR consortium have recommended that SCA Next formalises IDL subset for use with FPGAs
- Unlikely that additional SCA standardization for an FPGA CORBA profile until will happen without OMG standardisation first
- PrismTech with support from our ICO customers are in discussions with the OMG on new areas of FPGA CORBA standardization, specifically:
    - An IDL to VHDL language mapping – describing the subset of IDL language that is supported and its mapping to VHDL + BIOP
    - A CORBA profile for FPGA – describing the subset of CORBA functionality that should be supported by a compliant implementation

Summary

- ICO supports direct access to SCA components running on H/W
- Enables vision of SCA architectural consistency across all aspects of the SDR
- Eliminates the need for complex hardware abstraction layer protocols improving application portability
- Helps reduce time to market for new applications by simplifying FPGA integration task
- CORBA message processing is executed directly in hardware - 100s x faster than in S/W
- Eliminates the need for S/W proxies/adapters (i.e., MHAL Device) - reducing call latency

PRISMTECH

Thank you for Listening

- For additional information on PrismTech's Spectra products and services:
  - E-mail:
    - info@prismtech.com
  - Website:
    - www.prismtech.com/spectra
  - Today's Presenter:
    - Andrew Foster, Spectra Product Manager
    - Email: awf@prismtech.com

Thank You