OpenCPI - Open Component Portability Infrastructure:
A software framework to simplify complexity & enable code portability of real-time systems

AFRL Safe & Secure Systems and Software Symposium
June 2, 2009

John Scott
jscott@mercfed.com
240.401.6574
Agenda

• CPI Overview
  – Motivations, targets, applications
  – Concepts, models, architecture

• Key tasks
  – Experiments for assessment of key dimensions of suitability
  – OSS issue analysis and delivery of prototype OSS site/repository

• OSS effects and values
Problem to Solve

ISR embedded market is vertically integrated

- Difficult to change/deploy/modify/swap/upgrade sensors and compute resource to match demand (need) or supply (technology)
- High costs of program-specific solutions
- Limited standardization
- Slow to incorporate new ideas
- Lack of cooperation among platforms
- Limited community re-use of algorithms
Critical S3 Military Assumptions

Opportunities

- Moore’s law will continue (multi-core architectures, optical interconnects, etc.)
- Sensors will improve (higher resolution, larger aperture)
- Networking technology will improve (ad hoc MANETs, gateways, optical links)
- Funding for back-end comms, storage, and analysis (GIG, DCGS, agency) will continue

Challenges

- COTS is giving way to “Open” and the role of standards or managed ecosystems is in flux
- Bandwidth will remain scarce (civilians, jammers, platform proliferation)
- Sensor data rates will increase (more signals or pixels/sec, more platforms)
- Time will become even more critical (targets flee into cities, rough terrain, vegetation, little a-priori knowledge)
- Targets will become more amorphous in character
Our Approach to S3

• Address a problem space with a concrete solution

• Challenges:
  – "networked heterogeneous distributed systems"
  – "reliability, robustness, security, interoperability, and real-time operation"
  – "a social framework for participation, discussion"
  – "capture and reuse of domain knowledge and expertise through an open, community-driven, technically focused shared infrastructure"
  – "specific domains, but have the potential for expansion to other domains"
  – "support collaboration, encourage community-development"
  – "open frameworks, and have a clear path to industrial acceptance"

• Solution:
  – Use the best practices and value of OSS and its collaboration model
  – Migrate (from proprietary) and establish, as OSS, a development, integration and runtime framework (OpenCPI) that can support and integrate multiple real embedded applications and technologies.
Mission Needs & Technology drive Software Complexity

“Software is the central ingredient of the information age.” - DSB Task Force

Changing/uncertain mission req'ts combined with technology proliferation lead to more complex & costly development/integration/maintenance
Component Portability Infrastructure (CPI) is a real-time embedded (RTE) middleware framework that simplifies programming of heterogeneous processing applications requiring, in a system or across a tech refresh, a mix of processing and interconnect technologies.

- Designed for heterogeneous component-based applications
  - FPGA, GPP, DSP, GPU, Multicore processing, peer model, interoperable
  - Intra-chip, interchip, interboard, network interconnects within applications
  - Component basis means sharing and reuse of sub-application functions
  - The reality of complex, SWAP-sensitive, deployed applications

- Open standards compliant
  - US Government’s Software Communications Architecture (SCA)

- Industry standards used or supported
  - CORBA
  - POSIX
  - OCP

- TRL Level of 6, moving to 7 under current programs/projects
OpenCPI is...

- Hardware Abstraction Layer
- Real-time Middleware
- Real-time Virtualization
Where OpenCPI comes from

Component-based Application Model

Heterogeneous Computing Challenges

Wideband Datalink Requirements

DoD Open Systems, SDR mandates

Mercury FDK (FPGA) Experience+IP

Mercury/DARPA PCA Program

Mercury/JTRS JPO-funded Study/specification
Where OpenCPI can be used: application areas

- Application areas using advanced technologies to meet performance and SWAP requirements.
- Characterized by rapid change and/or component-level re-use
- Tech refresh demands tech changes
Where OpenCPI can be used: technology

- Mixing and scaling heterogeneous processing and interconnect technologies
- Close enough "to-the-metal" for high performance and low SWAP
- OS/RTOS-agnostic

CPI

Typical GPP (PPC/Pentium)

DSP (e.g. TI)

FPGA (using OCP standard) (Xilinx/Altera)

Specialized multi-core (e.g. IBM Cell, GPU)

Homogeneous many-core (Intel, Freescale)
OpenCPI: Can support multiple control frameworks

- Application and system management approaches are factored from how apps are written
- Compliant with standard frameworks, but also can be lean and simple, without runtime “management”

CPI

SCA
(Software Radio)
(from DoD/JTRS)

CCM
(CORBA Component Model)
(from OMG/ISO)

Standalone XML-driven
(CCM-based)
SW/Firmware Component Interface Pattern

Management Interface

\[ e.g. \text{start/stop/configure} \]

Input Ports:

- Messages/data from other components to this component.
- These ports are “input” or “provider” or “server” or “consumer”

Requests from component for local services

Component

Output Ports:

- Messages/data from this component to other components.
- These ports are “output” or “user” or “client” or “producer”

- Building blocks for heterogeneous component-based applications
- Same pattern/model for any processing technology
  - GPP/DSP/FPGA/GPU/Multicore/Cell
- Models for all technologies must interoperate in a system
CPI Summary

• **Infrastructure to support component-based applications**
  – Control plane & Data Plane support
  – FPGA IP & Software/Middleware/Drivers
  – Setup/control/management & Runtime/execution
  – Compliant with DoD/JTRS SCA control/deployment system

• **Targeting the challenges of heterogeneous systems**
  – FPGA-capable, for VHDL/Verilog application components
  – *Peer* model with GPP and DSP software (components)
  – Reuse/Portability/Interoperability within and across technologies

• **Driven initially by high bandwidth data link applications**
  – Applications are waveforms
  – Waveforms are assemblies of connected components
  – Most components must be in FPGAs, nxGB/s data flow

• **Driven by DoD communications requirements.**
  – Open standards and interfaces
  – SDR/SCA Compatibility
  – Waveform portability model:
    ▪ DoD owns/provides waveforms to primes/platforms/terminals.
    ▪ Upper limit of component granularity

• **Applicable to other applications: SIGINT, RADAR, CIED, Net-centric ISR**
Key Tasks (1/2): experiments: prototyping/assessment

• 1.1 Processor & interconnect technology agility and supportability
  – Candidate selected is a new component implementation model for modern GPU processor technology, via industry-standard OpenCL (Nvidia/ATI/Intel/Apple etc.)
  – Recent & different processor technology entering embedded defense applications now

• 1.2 Platform/system supportability
  – Candidate selected is commonly available, inexpensive, modern PC platform supporting inexpensive development boards for FPGA, GPU, Cell.
  – Lowest possible barrier to entry for experimentation and community

• 1.3 Application domain suitability (ease-of-use, natural implementation models etc.)
  – Candidate selected is airborne persistent surveillance
  – Multiple recent programs seeking to do more complicated and flexible processing and exploitation in airborne platforms
Key Tasks 2/2: transition to OSS & community development

• 2.1 Analysis of OSS issues for OpenCPI
  1. Determine appropriate governance and OSS licenses
     ▪ Enable/ensure widest usage/exploitation
  2. Examine OpenCPI business and community model
     ▪ Enable/ensure widest usage/exploitation
  3. Resolve ITAR/Classification issues
     ▪ Enable effective usage for ITAR and classified contexts
  4. Determine best methods for how to scale OpenCPI community in Phase 2
     ▪ Prioritize/determine best early launch enhancements/efforts

• 2.2 Construction/Delivery
  1. Derive/extract open code base from proprietary baseline
  2. Deploy OpenCPI community collaboration website/OSS repository
  3. Deliver OSS roadmap for OpenCPI in embedded defense apps
Task 1.1: Processor & interconnect technology agility

• Add a new component authoring model to the existing set:
  – "SCA" for C++/CORBA/POSIX/JTRS compliance
  – "RCC" for lean C/DSP/Microcontroller/Multicore
  – "RPL" for FPGA-based computing and I/O

• Latest entrant on embedded processor tech scene: GPUs
  – "GPGPU": General Purpose use of Graphics Processors
  – Until recently, no API or language with vendor independence and portability.

• Now there is OpenCL
  – A trademark of Apple Inc. defined by Khronos Group:
    ▪ A "member-funded consortium focused on the creation of royalty-free open standards".
    ▪ Same group that standardizes OpenGL for graphics apps.
  – An control API and computational language+API for GPUs
    ▪ Also attempts to address other technologies, but is focused on GPUs.
1.1: OpenCPI Generic Component Model

- Building blocks for heterogeneous component-based applications
- Same *pattern/model* for any processing technology
  - GPP/DSP/FPGA/GPU/Multicore/Cell
- Models for all technologies must *interoperate* in a system
1.1 Authoring models for Component Implementations

- **Components in general**
  - **“Waveform” Components**
  - **Simpler Interfaces** originally from SCA 3.1/CP289
- **General Purpose Processors**
  - PPC/Pentium
  - CORBA CORBA Interfaces
  - CORBA APIs
  - OO Languages
  - POSIX
  - SCA 2.x
- **Specialized Hardware Processors**
  - Resource-Constrained C-language
  - -or- RTL-Programmable Language
- **SCA Components**
- **SHP Components**
  - Generalized SCA Component Model
  - Used to model Waveforms
  - Common interface definitions (IDL)
  - Same modeling tools (e.g. Zeligsoft)
- **RCC Components**
  - DSP
  - RISC (small mem)
  - Cell SPU
  - Microcontroller
  - C-language APIs
- **RPL Components**
  - FPGA
  - ASIC
  - VHDL/Verilog APIs
  - OCP interfacing
  - RTL-Programmable Language
- **GPU Components**
  - NVidia
  - ATI
  - Intel/Lararbee
  - Based on OpenCL

Like-for-like Portability by Recompiling Source Code for GPP, RCC, or RPL
Processor Technology Agility: add GPUs

- Use recent industry standard to stay independent of GPU vendor
  - OpenCL spec version 1.0 recently released
  - Abstracts vendor-specific details but no more
  - Usable for various multicore architected
  - Required to fully exploit GPU power
  - Thus requires additional model (added to GPP, DSP, and FPGA)

OpenCPI is in this category

Appropriate Interface for OpenCPI

Vendor-Specific

 AMD Graphic Positioning OpenCL

Applications (Game Computing, Video Computing, Scientific Computing, Productivity)

Tools, Libraries, Middleware (ACML, Brook+, Cobra, RapidMind, Havok etc.)

Industry Standard Interfaces (OpenCL, DirectX 11®, etc.)

Compute Abstraction Layer (CAL)

AMD Stream Platform
Platform/system supportability: add easiest platform

- Add support for a new, but non-embedded development platform
  - Maximize ease of experimentation and adoption
  - Minimize barriers to trying technology, developing code
- Use low cost widely available modern technology
  - PC Platform (Intel x86 and PCI Express-based)
  - Support self-hosted development
- Enable true heterogeneous development
  - Room/slots for FPGA add-in development board
  - Room/slots for GPU add-in cards
  - Room/slots for Cell add-in cards (not supported in this project)
- Enable data streaming performance for ISR and comms apps
  - With cheap/COTS platform, show performance potential
Modern, low-cost, powerful development system

- Latest desktop technology that supports high speed slots for other technologies
- 3 high speed slots for other processors: GPU, FPGA, Cell
1.2: Choices for OpenCPI reference platform

OpenCPI Reference Platform, COTS Component Specification

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>By</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>2009-03-09</td>
<td>ssiegel</td>
<td>Draft</td>
</tr>
<tr>
<td>0.02</td>
<td>2009-04-20</td>
<td>ssiegel</td>
<td>Added “rationale” text</td>
</tr>
<tr>
<td>0.03</td>
<td>2009-04-22</td>
<td>ssiegel</td>
<td>Added closed chassis option</td>
</tr>
<tr>
<td>0.04</td>
<td>2009-05-01</td>
<td>ssiegel</td>
<td>Name change, minor edits</td>
</tr>
</tbody>
</table>

This document specifies the COTS components used for the OpenCPI FPGA Reference Platform (OC-FRP). The parts and vendors specified herein have been selected for their ubiquity and low-cost. For most of the items listed, component substitutions may be made; however the burden of testing is then upon the user. Pricing is non-binding, quantity-one, guidance only.

Development and Target Computer, Hardware Components

It is possible to use one computer as both a development machine and a target. Table 1 below lists components central to the performance of the system.

Table 1 – FPGA Reference Platform, Hardware Components

<table>
<thead>
<tr>
<th>Line</th>
<th>#</th>
<th>Manufacturer</th>
<th>Part Number</th>
<th>Cost</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td><a href="http://www.evga.com">www.evga.com</a></td>
<td>132-BL-E798-A1</td>
<td>$300</td>
<td>X58 Motherboard</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td><a href="http://www.intel.com">www.intel.com</a></td>
<td>Corei7 920</td>
<td>$280</td>
<td>Nehalem 2.66GHz x4 Processor</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td><a href="http://www.corsair.com">www.corsair.com</a></td>
<td>DDR3 3x2GB</td>
<td>$190</td>
<td>6x2GB (12GB) DDR3 1600 Memory</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td><a href="http://www.evga.com">www.evga.com</a></td>
<td>01G-P3-1280</td>
<td>$335</td>
<td>GTX260 240-core GPU</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td><a href="http://www.enermax.com">www.enermax.com</a></td>
<td>EMD625AWT</td>
<td>$150</td>
<td>625W MODUS2+ Power Supply</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td><a href="http://www.westerndigital.com">www.westerndigital.com</a></td>
<td>WD10EADS</td>
<td>$105</td>
<td>1TB SATA Disk Drive</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td><a href="http://www.antec.com">www.antec.com</a></td>
<td>P182</td>
<td>$120</td>
<td>ATX Mid Tower Computer Case</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td><a href="http://www.xilinx.com">www.xilinx.com</a></td>
<td>HW-V5-ML555-G</td>
<td>$2200</td>
<td>Xilinx ML555 V5 Dev Kit</td>
</tr>
</tbody>
</table>

Notes:
1. Cost estimated, quantity one, as observed online Q1-2009
1.2 Platform/system support: task status

- Platform (includes all SW and tools) defined, documented
- Platform procured, installed, operational.
- OpenCPI core code base built/installed.
- FPGA reference platform support:
  - Tools/build process established up to operational FPGA "hello world".
  - OpenCPI control plane implemented/tested to spec.
- Remaining to be done:
  - Ensure/test trouble-free procurement-to-"hello world" for a new experimenter on the reference platform.
  - OpenCPI Data plane support for PCIe
  - Driver adaptation for control and dataplane.
  - Show mixed technology component-based examples.
  - Final document for procurement and usage (platform-specific).
Task 1.3: Domain suitability: Persistent surveillance

- Recent powerful trend to perform more and complex processing in SWAP-constrained airborne platforms
  - Multiple sensors
  - Airborne processing, exploitation, storage
  - Low latency tactical mixed with forensic
- We will assemble an example app from open-source algorithms and data sources.
Key Tasks 2/2: transition to OSS & community development

• 2.1 Analysis of OSS issues for OpenCPI
  1. Determine appropriate governance and OSS licenses
     ▪ Enable/ensure widest usage/exploitation
  2. Examine OpenCPI business and community model
     ▪ Enable/ensure widest usage/exploitation
  3. Resolve ITAR/Classification issues
     ▪ Enable effective usage for classified contexts
  4. Determine best methods for how to scale OpenCPI community in Phase 2
     ▪ Prioritize/determine best early launch enhancements/efforts

• 2.2 Construction/Delivery
  1. Derive/extract open code base from proprietary baseline
  2. Deploy OpenCPI community collaboration website/OSS repository
  3. Deliver OSS roadmap for OpenCPI in embedded defense apps
Vision: OpenCPI Productivity & Community

Integrated Productivity

- Integrated verification, simulation, and runtime instrumentation
- Software and hardware solutions optimized for different stages of development and deployment to minimize time-to-field

Open Performance

- Open, Linux-based, real-time operating & middleware environment
- Open, high-performance libraries and middleware
2.1.1 OSS Licensing for CPI

Things to keep in Mind:

- Intellectual Property
- Copyright
- Trademark

Assumptions for OSS Licensing

1. Must be GPL compliant, to enable maximum usage of other OSS
2. Do not create new or 'special' Government-license
3. Needs to 'fit' into the real-time embedded community
4. Seek to lower friction of using OpenCPI: need to foster greatest use as possible, leading to growing and scaling community
2.1 OSS Basics: IP, Copyright and Trademark

Copyright:
- Copyright: owner *has an exclusive right to do* certain things
- Copyright grant is an affirmative license to copy, modify, or distribute the software owner by the licensor.

Patents:
- Patents: owner *has a right to exclude others from doing* certain things
- Intellectual Property & Patent Laws
- Patent grant is an affirmative license to practice patents necessary to make, sell or offer for sale, or import the software, but only to the extent of patent claims actually owned or controlled by the licensor.

License
- Gives people permission to use someone else's IP
- Puts bounds and conditions around use of IP
- Allows IP owners recourse if conditions not met (conditional use)
- A license can be a contract (differing views)

Trademark
- Provides exclusivity over a name
OSS Licenses

Permissive (or Academic)
- Unfettered access to code
- Can be added into proprietary applications
- No requirement for downstream sharing of source code
- Examples: BSD, MIT, Apache

Partially Closable
- Proprietary applications can use unmodified version of the library in a closed source, proprietary licensed product. If they make changes to LGPL'ed library they must distribute the modified source code corresponding to that library along with the binary application.
- LGPL projects: Jboss, OSSIM

Reciprocal
- Requires licensee of code to reciprocally apply the same open source license to any code derived from the originally license code
- Each binary distribution contain also include full source code to the application
- Examples: GPL & Linux
How to Pick an OSS License

Do you want to relinquish any control over how your code is used and distributed?

• **NO:** put it under the BDS/MIT
• **YES:** Copyright it, and ask: **Do you want to allow people to use your code in non open-source programs?**
  - **NO:** release it under the GPL
  - **YES:** If somebody uses your code in their program and sells their program for money, do you want some of that money?
    • **YES:** Dual-license or don't release the source at all and use a closed-source license.
    • **NO:** Use a "commercial-friendly" license, and ask: **If somebody uses your code and improves it (fixes bugs or adds features) do you want to make them give you the improvements back so you can use them too?**
      - **NO:** Use a non-reciprocal license.
      - **YES:** Use a reciprocal license → LGPL
OSS Licensing for OpenCPI

Best candidate is LGPL

- Partially Closable License: proprietary applications can use unmodified version of the library in a closed source, proprietary licensed product. If changes made to LGPL'ed library, modified source code corresponding to that library along with the binary application must be passed along.
- Derivative works must be made available under LGPL
- Primarily used for software libraries
- LGPL projects: Jboss, OpenOffice.org, Mozilla
2.2.3 Development in Constrained Environments

- More specialized application code
- CPI Codebase (Public)
- ITAR/FOUO
- SECRET
- TS/SCI

Bias to make open, Justify exceptions

Future setup: Use forge.mil, etc.
S3: How-to Plan

1. Community, S3 must be about how to grow and evolve effective communities for producing military software
   - Software is an instantiation of knowledge around a defense capability
   - Email lists (all here invited to join, send to jscott@mercfed.com)
   - Web site, [www.OpenCPI.org](http://www.OpenCPI.org)
     - TO Launch June 5!
     - Email: jscott@mercfed.com for notice!
   - Code repository open July 1

2. Demonstration:
   - Reference Platform, w/Bill of Materials for how-to
   - New Application
   - New processor technology
OSS Community

Differing Technologies

Technical Level

Proprietary Because of its Innovation

Open Source

Proprietary
Simple, but no OS sol’n yet

Proprietary
Simple, but no OS sol’n yet
As OpenCPI Technology Mature, S3 grows

Companies provide services by providing improved technology

Reduced areas for companies to provide proprietary lock-in

Technical Level

Differing Technologies

New Proprietary

Proprietary

New Open Source

Original Open Source
OpenCPI Impact

• **Greater utilization, lower cost of ownership**
  - Underlying software and hardware infrastructure supports open, plug-plot software capabilities reducing development, maintenance, and logistics
  - Capability to on-load processing lessens the need for forward-deployed exploitation cells, reducing staffing and security costs

• **Value to DoD, utility and ROI on Government investment:**
  - Faster time to change/update functionality to deliver capability
  - Faster time to adapt and exploit new technologies, especially in embedded space integration risk reduced due to robust units of deployable technologies
  - *Better re-use of sub-application components in embedded ISR.*

• **Ultimately increased competition and sharing leading to lower costs.**
Community: real-time signal processing for embedded, heterogeneous systems for Comms, xxxINT (Signals, Communications, Electronics, etc. - Intelligence) and EW/CIED in defense ISR systems.

DoD is slated to spend Billions in R/D in coming years on ISR-associated platforms

CPI Usage:

- DARPA PCA-program funded infrastructure work in dynamically morph-able hardware/software systems
- JTRS JPO-funded standardization on SDR-compatible component models for DSP, FPGA and ASIC processing
- XMidas SIGINT application demonstration on field deployable platforms with FPGA integration
- HDR-RF SATCOM Modem early access demonstration of CPI
- EW/CIED applications (JCREW/I2WD)
Backup
Open Component Portability Infrastructure (OpenCPI)

Description:
An Open Source Software (OSS) software framework to simplify complexity & enable code portability of real-time systems

- Middleware for Waveform-Ready Processing Platforms
- Real-time middleware for embedded systems
- Improved waveform code portability with standards-based interfaces
- Increased interoperability using container technology
- Quicker time to evolve and deploy

Capability Gaps: (1) portability/interoperability of ISR capabilities and (2) modular, open-service oriented architecture to insert new applications quickly across multiple platforms.

Schedule:
- Funding: OSD-DDR&E
- Program Manager; AFRL Rome Labs
  - AFRL ARC Facility as principal bench test facility
- Test and Evaluation
  - 6 months: Concept refinement and Technology integration
- Transition
  - Phase 2, scale community and increase functionality

Funding:
Phase 1, Convert to OSS $350k
Phase 2, Scale Effort (tbd) -
Total $350k
Critical S3 Military Assumptions

Opportunities

- Moore’s law will continue (multi-core architectures, optical interconnects, etc.)
- Sensors will improve (higher resolution, larger aperture)
- Networking technology will improve (ad hoc MANETs, gateways, optical links)
- Funding for back-end comms, storage, and analysis (GIG, DCGS, agency) will continue

Challenges

- COTS is giving way to “Open” and the role of standards or managed ecosystems is in flux
- Bandwidth will remain scarce (civilians, jammers, platform proliferation)
- Sensor data rates will increase (more signals or pixels/sec, more platforms)
- Time will become even more critical (targets flee into cities, rough terrain, vegetation, little a-priori knowledge)
- Targets will become more amorphous in character
OpenCPI Productivity Impact

How the proposed technology will improve the producibility of DoD software-intensive systems?

1. Technology
   - Tactical SOA
   - CPI brings can help mediate asynchronous needs with synchronous capabilities

2. Community
   - Centralizing the community around a common infrastructure will lead to better technology. Need to focus on what the community of developers and users need first then focus on tech
     - Freedom to use: Bottom up vs. Top-down
     - The OpenCPI community will need:
       - Test suites and data
       - Developer tools
       - Ability to find supporters of the technology
Open Technology Development & OSS

OTD Fostering collaboration across DoD on technology acquisition and development.

1. Open Standards, Open Data, Open Interfaces: enable systems and services to evolve

2. Use of best-of-breed Open Source Software (OSS)

3. Open Source Software Development Practices

Military OSS Projects:

- VSIPL
- LargeData JCTD
- TISC JCTD
- USAF FalconView (soon)
- OSSIM
- Delta 3D
- BRL-CAD
- Optics (BallForge.net)