Middleware Challenges for Reconfigurable Computing

Cray User Group Conference

Albuquerque, New Mexico May 16-19 2005

Contents

Introduction

Koan (koh' on) - "A Zen teaching riddle. Classically, koans are attractive paradoxes to be meditated on; their purpose is to help one to enlightenment by temporarily jamming normal cognitive processing so that something more interesting can happen."

www.dict.org

CUG 2005 – Albuquerque, NM

Introduction

- Incarnation of a prior company Progress Forge
  - VIVA Training, RC Integration
- Primary Office located in Columbus, Ohio

- Focused on RC integration tools and services
- Platform Lab Technical Operations

CUG 2005 – Albuquerque, NM

Introduction

Question: "This reconfigurable stuff looks great but *how do I get data into it*?"

Purpose: Accelerate the realization of benefits from Reconfigurable Computing by making integration Fast, Simple & Painless

The Promise of RC

- History
  - Prototyping
  - Embedded Systems
  - Application Acceleration
  - Complete Computational Systems
- FPGA Technology Advancement
- Promise of RC within High Performance Computing
- Barriers

The Promise of RC - History

- Prototyping
  - Save Time/Money on ASIC generation
- Embedded Systems
  - Flexibility
- Application Acceleration
  - Accelerate Computationally Intense Algorithms
- Computational Systems
  - Tightly Coupled, Larger RC Space, Faster Architecture

The Promise of RC - Technology

- FPGA Technology Advancement
  - Real Estate
  - Clock Speed
  - Power Usage
  - Heat Generation
  - Added Features (Multipliers, Block RAM, I/O, PPC Cores)

The Promise of RC - Computing

- Extremely Good at Computationally Intense Tasks
  - Hardware vs. Software
  - Parallel and Pipelined Implementations
  - Fast Clock Speeds
  - Strong I/O speed potentials

The Promise of RC - Barriers

- Development Tools
  - HDL's still rule the roost
  - Higher-Level Tools are emerging
    - Impulse-C, Confluence, Handel-C, Mitrion-C, VIVA, etc.
- Success Stories
  - “Few” and Scattered
  - Difficult to Verify

The Promise of RC - Barriers

- Lack of Standards
  - Hardware is proprietary
  - Cores are proprietary
  - Becoming an industry focus (OpenFPGA, OCP-IP)

- RC Integration Challenges
  - No Products or Support, Rip & Tweak
  - Roll your own, for each RC platform, Core and Dataset...

RC Integration Challenges

- Vendors leave off at the Bus or API
- Wide diversity of implementation models
  - Bus Architectures
  - Core Interfaces
  - No Standards
- Cores support no high-level functions (nor should they)

RC Integration Challenges

- An RC integration layer is needed that is:
  - Platform Independent
  - Extensible
  - Easy to Use
- Integration Layer Should Provide “Common” Middleware Functions
  - Hardware Interfaces
  - RC Applications
  - Host Applications
  - Data Access
  - Security, Logging, Scheduling, etc...

Reconfigurable Computing Management System (RCMS)

- GOALS
  - Interface with wide range of RC platforms
  - Support diversity of traditional systems
    - Host Applications
    - Data Interfaces (Databases & Files)
  - Platform Independent
  - Networking
  - Security
  - Minimize Performance Impact
  - Support Development Process

CUG 2005 – Albuquerque, NM

RCMS Goals

- Interfaces with Wide Range of RC Platforms (Process Interfaces)
  - Hardware Architectures
    - Proprietary – Vendor Specific
    - Embedded vs. Complete
  - Application Specific – Every Core is Different
    - Data Transfer
    - Data Types (Apps, OS's, Cores, Networks)
    - Synchronization (Vendor AND Application Specific)

RCMS Goals

- Support diversity of traditional systems
  - Application Integration (Process Interfaces)
    - Host Applications
    - Operating Systems
    - Pluggable
  - Data Interfaces
    - Easy & Efficient
    - Pluggable

RCMS Goals

- Platform Independent
  - Java (J2SE v1.5)
    - 32 & 64 bit versions
    - Linux & Windows (Solaris & OS X)
    - Console, GUI & Remote Command Interface
  - XML – Commands, Environment and Dataflow Definitions
  - Networking
    - TCP/IP
    - Pluggable (CANBus, ModBus)

RCMS Goals

- Security
  - Support for Encrypted Data Streams (AES)
  - Encrypt Environment Information
- Minimize Performance Impact
  - Small Footprint, Efficient Code
  - Multi-threaded, Scalability
  - Tweak Java Environment
  - Compressible Data Streams
- Support Development Process
  - Version Control, Scheduling, OS Scripting, Logging

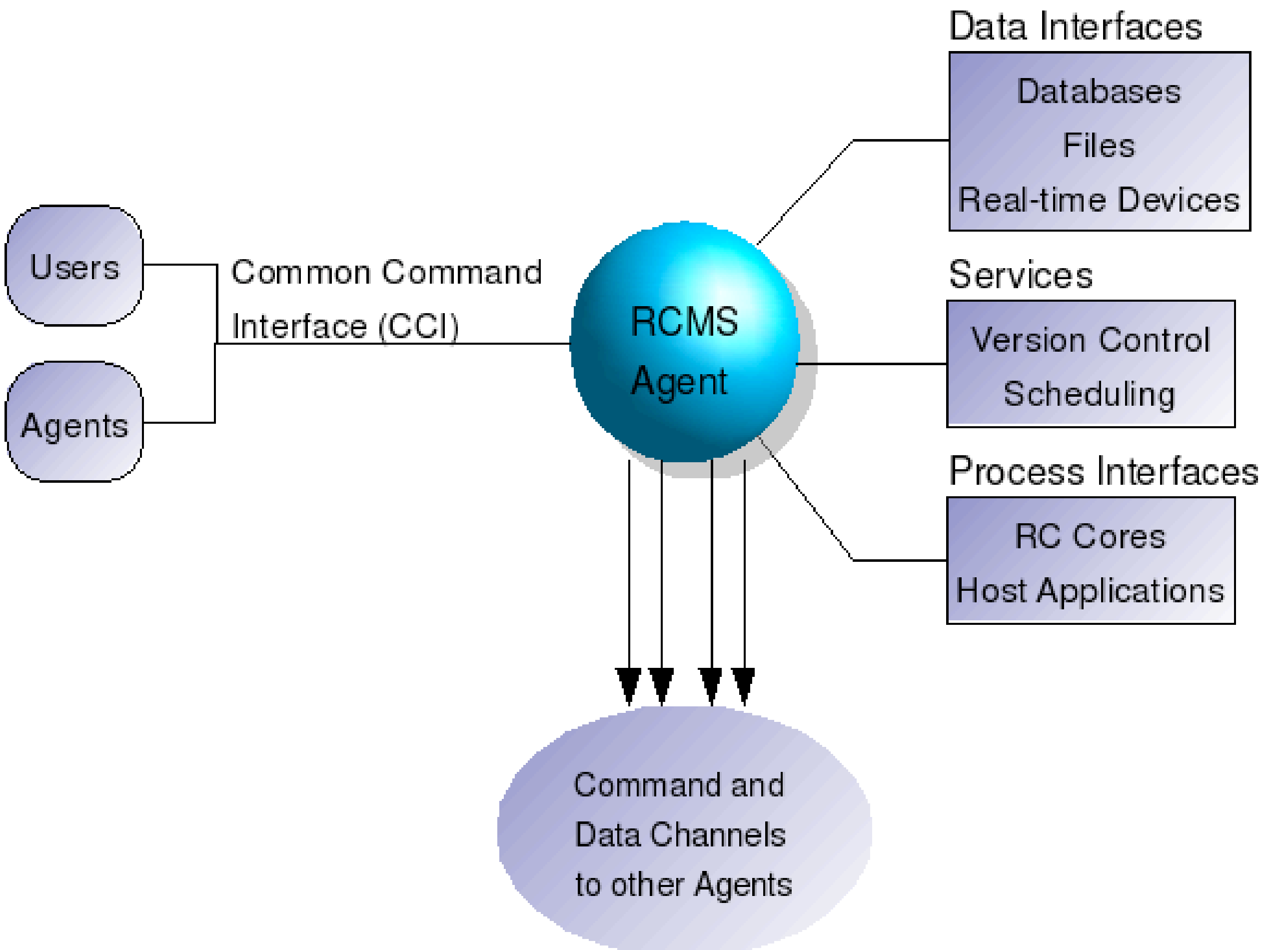
RCMS Goals

- Pluggable
  - Process Interface
    - RC Platforms/Cores
    - Host Applications
    - Locate, Open, Configure, Push Data, Pull Data, Close
  - Data Interface
    - Databases
    - Files and RT Controls
    - Open, Query, Insert, Update, Delete, Execute, Close

RCMS Approach

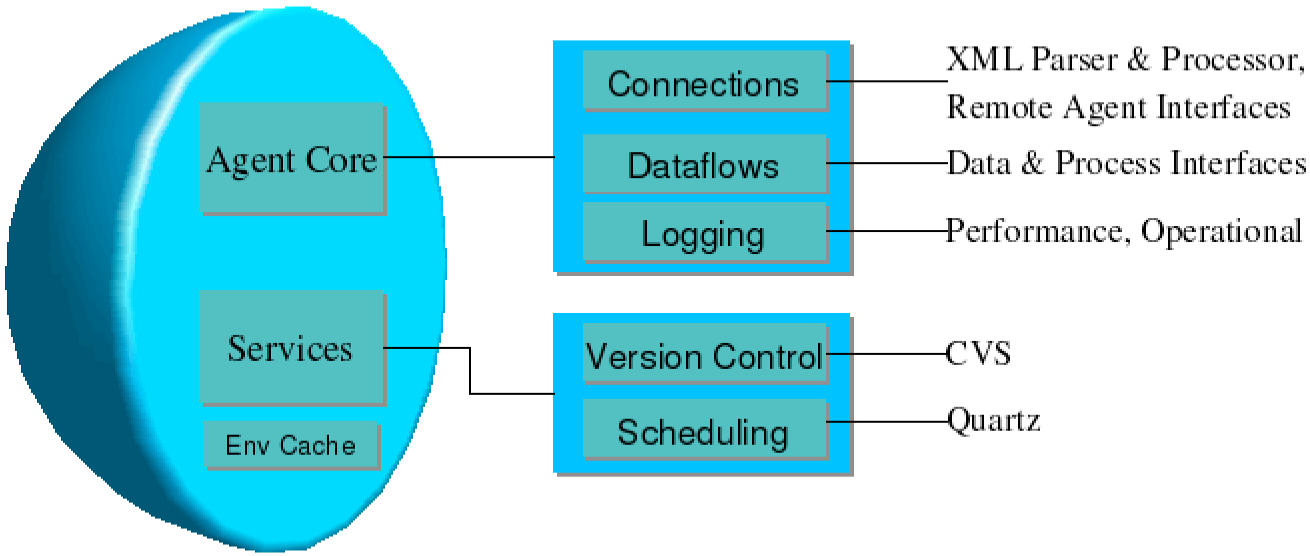
- Approach
  - **Agents** manage **Dataflows** across an **Environment**
  - Agents – Primary RCMS Operators, *they get stuff done!*
  - Dataflows – Data and Process Interfaces (XML)
    - Data Interfaces – Source & Sink Data
    - Process Interfaces – Manipulate Data
  - Environments (XML)
    - Define Agents – Who lives in the neighborhood?
    - Define Assets – What do we have access to?
      - RC Platforms, File Systems, Databases, Services, Applications

RCMS – Agent Overview

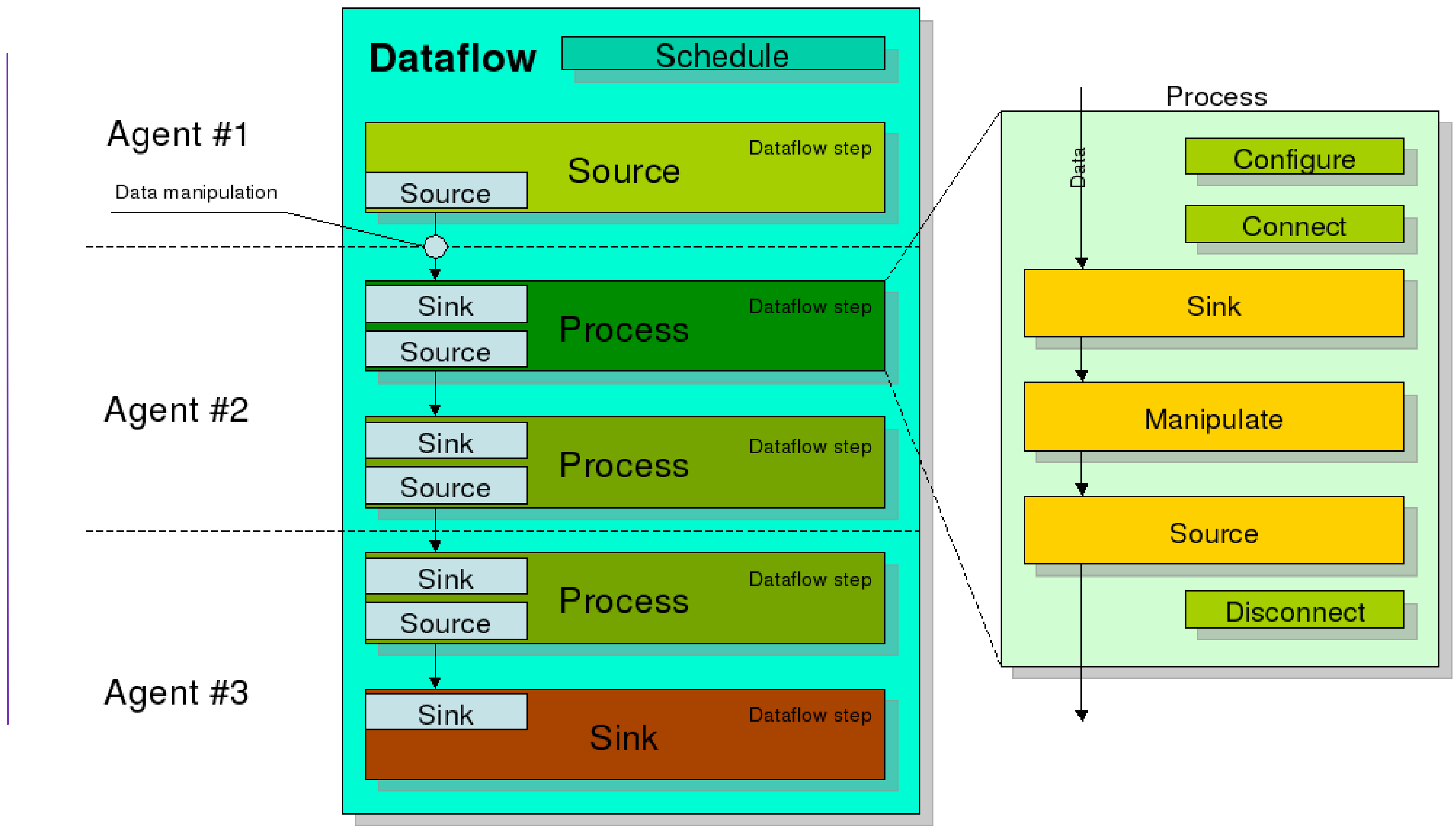


CUG 2005 – Albuquerque, NM

RCMS – Agent Internals



RCMS Approach



CUG 2005 – Albuquerque, NM

RCMS - XD1 Process Interface

- The XD1 Process Interface
  - The Development Journal
  - Approach - Wrap XD1 API in Process Interface with JNI
  - Data Push & Pull methods different from accelerator architectures
    - No bulk transfer write() or read() methods
    - Uses memory sharing between Host and Core
    - No means of concurrent data writes synchronization
    - Flexible but “we haven't seen this before”

RCMS - XD1 Process Interface

- Approach - Two Core Communications Functions
  - Small transfers and cores with no support for double-buffering
    - One time value approach, Data transferred using AIR Registers
    - User Defines starting address
    - Data Sourced/Sinked with incrementing address
  - Quad-buffer to handle more complex scenarios
    - Extension of double-buffer approach in MTA
    - Provide for concurrent Reads/Writes
    - First word in buffers defines state control
    - Buffer Addresses, length and other data is specific to the Core (and App) from dataflow to AI registers

RCMS - XD1 Process Interface

- Cray XD1 interface for RCMS, config params:
  - device: FPGA device node (/dev/ufp0)
  - core-file: the name of the core file to load into FPGA
  - register: comma-separated entries used to program FPGA registers. address-value pair
  - register-range: an address-filename entry, to load data into several registers at once.
  - source-protocol: type of data transport protocol to use for sourcing data.
  - source-buffer1-offset: address of first source buffer. Applicable only if source-protocol is DoubleBufferedSharedMemory.
  - source-buffer2-offset: address of second source buffer. Applicable only if source-protocol is DoubleBufferedSharedMemory.
  - source-offset: address of starting register for ApplicationInterfaceprotocol type.
  - source-chunk-length: how many words to source at a time. Defaults to 8.
  - source-data-length: how many words of data to source.
  - sink*: equivalent parameters for sinking data

RCMS - XD1 Process Interface

- Issues
  - No EOD (End Of Data) marker, the XD1 interface cannot determine when the Core has finished sourcing data.
    - This has been resolved by specifying the "source-data-length" parameter.
    - New enhancements to this protocol will provide for obtaining data transfer status information from the Core.

RCMS - XD1 Process Interface

- XD1 Interface Results
  - It Works!!
  - Tested on Cray XD1 located at OSC-Springfield
  - Currently tweaking for performance
- Next Steps
  - Test with diversity of cores
  - Need user input for wider assortment of Core Interfaces

Reconfigurable Computing Management System (RCMS)

- RCMS Status
  - RCMS v1.0 Released March 2005
    - Support for Nallatech Dime-II Platform
    - Few Features but Proved the architecture to be sound
  - RCMS v1.1 Release – May 2005
    - Support for Cray XD1
    - Met all expressed goals
    - Added GUI

The Future

- Standards Development through OpenFPGA
- Automated Environment Discovery
- Dataflow Templating
- Security Integration (LDAP, PAM)
- Strong EDA Support
- Cluster and Grid Integration
- Controls Integration (JME, CANBus, ModBus)
- Acceleration On-Demand via Networked RC Resources!

Conclusion

- The performance benefits of RC are beginning to be realized
- Development and Integration Tools are evolving
- RCMS fills the middleware gap in RC technologies
- Standards are desperately needed
- XD1 Interface Project was a success
- RC Technology should be ubiquitous and invisible

Q & A

Koan Corporation

Suite 244

1275 Kinnear Rd.

Columbus, Ohio

43212

www.koancorporation.com

614.390.0932

tyler.reed@koancorporation.com

614.595.1098

CUG 2005 – Albuquerque, NM