FPGAs & Software Components

Graham Bardouleau & Jim Kulp
Mercury Computer Systems, Inc.

High Performance Embedded Computing (HPEC) Conference
September 29, 2004

The Ultimate Performance Machine
Introduction

- FPGAs can now be used as scalable processing resources in heterogeneous multicomputers, not just I/O enhancers.
- Many applications need multiple processor types for “best fit” (power, weight, etc.).
- We must enable FPGAs to be “full peers” without undue tax on the FPGAs resources.
- Our approach has two thrusts:
  - Component programming models at application level and component level, building on standards.
  - Infrastructure elements that enable a common control and communication model between peer processors, including the “middleware” for FPGAs.
Programming Models

• Application programming must enable all processing resource types to be easily integrated (and changed/inserted).
  • Component (software) model does this
  • Standards are established for this (OMG and JTRS)
  • Build on this heterogeneous model to embrace FPGAs

• Effective use of FPGA technology still requires writing VHDL, and sometimes special features/macros of specific FPGAs.
  • Define and enable standard VHDL interfaces for external interactions, enabling peering with other component types
  • Provide more portability and less dependency on choices of FPGA, fabric technology and peer processor types
Infrastructure Developments

- How to “bring FPGAs into the first world”? 
- A common control model and mechanisms that can work across processor classes:
  - Load, initialize, configure, start, stop, connect, etc.
- A data movement and synchronization model that can be supported locally everywhere
  - Streaming, data reorg, and request/response messaging
- The FPGA driver and proxy code to treat FPGAs as “computers that can load and run code that talks to others”
- On-chip lean infrastructure (IP) to enable it all to work