Computer Systems, Inc.
MERCURY

FPGAs & Software Components

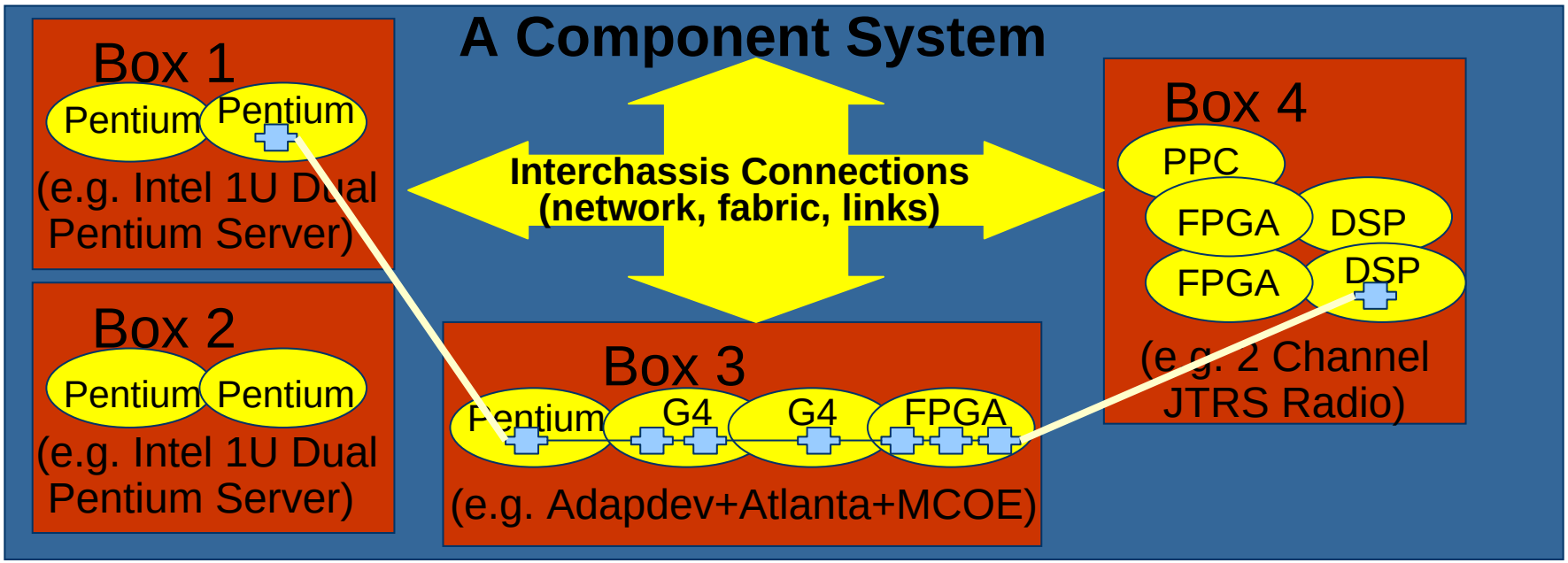
Graham Bardouleau & Jim Kulp
Mercury Computer Systems, Inc.

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

The Ultimate Performance Machine

Goals

- FPGAs can now be used as scalable processing resources in heterogeneous multicomputers, not just I/O enhancers or glue logic.
- Many applications need multiple processor types for “best fit” (power, weight, etc.).
- We must enable FPGAs to be “full peers” in the multicomputer, without undue tax on FPGA resources.



FPGAs & Software Components

Approach

- **Our approach has two thrusts:**
  - **Component programming models at application level and component level, building on standards.**
    - **How to write applications, as a set of components**
    - **How to write components, as building blocks for apps**
  - **Infrastructure elements that enable a common control model, and common communication model between peer processors of all types, including the "middleware" for FPGAs**
    - **How components are managed**
    - **How components communicate with each other**

FPGAs & Software Components

Application Programming Model

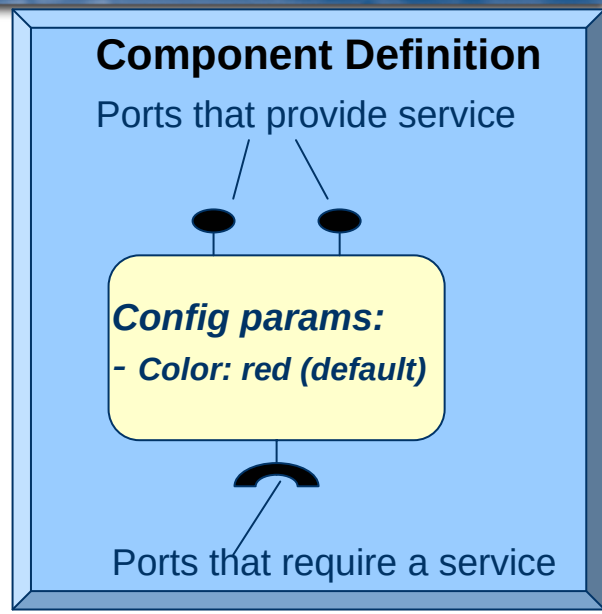
- Enable all processing resource types to be easily integrated (and changed/inserted).
- Support real world, flexible mixing of GPPs, DSPs, FPGAs.
- The Component Software model does it.
  - A hardware-ish way of building software, usable for FPGAs
  - Application building blocks that can have different implementations (even different source code), for different processor types
- Standards are established for this (OMG and JTRS).
- We build on this heterogeneous model to embrace FPGAs.

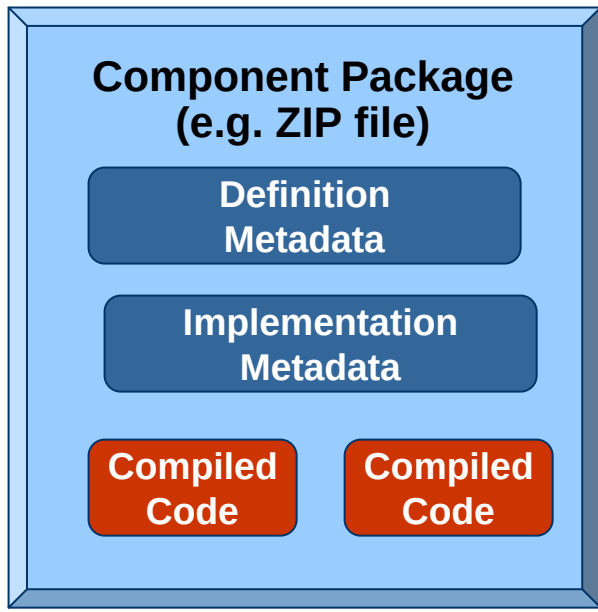
What's a Component?

- A (software/FPGA) package which offers services through interfaces.
- A reusable part that provides the physical packaging of implementation elements.
- An independently deliverable package of software that can be used to build applications or larger components, or be an application itself.
- ***A unit of software that is pre-built, packaged, self-describing, which can be individually deployed or updated or replaced in the field. It can be sent as an email attachment.***
- ***A well behaved DLL on steroids?***

What's a Component?

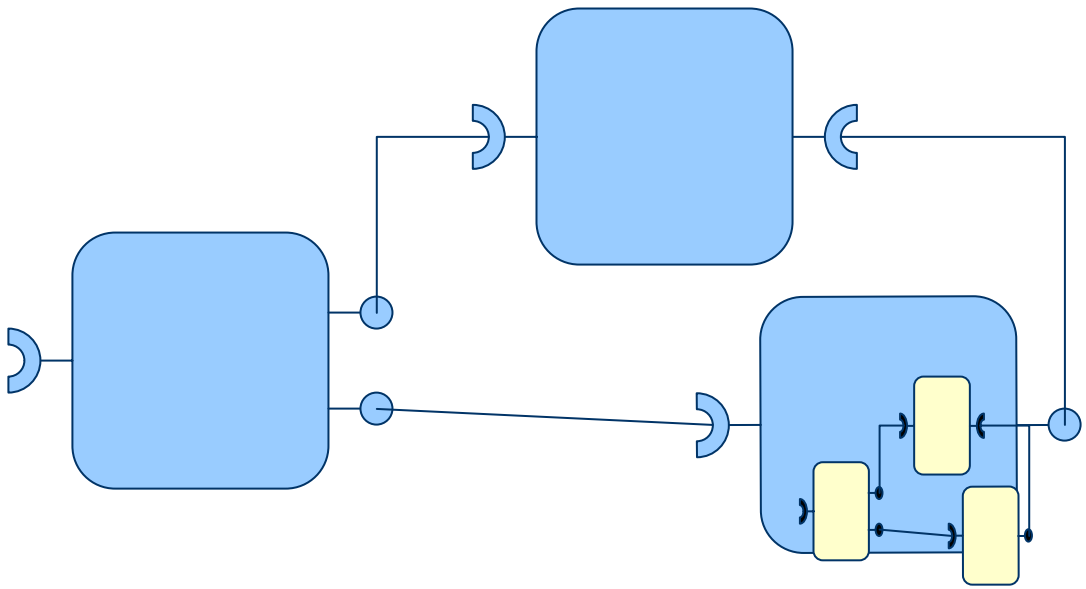
- Defined for its "users" by:
  - Ports that *provide a service* via an interface/protocol (component acting as server)
  - Ports that *require (use) a service* via an interface/protocol (component acting as client)
  - Configuration (instantiation) parameters.
  - An overall functional behavior
- Packaging (e.g. zip archive) of compiled code files (e.g. DLLs) and descriptive metadata (e.g. XML).
- Metadata allows tools and runtime environments to know how to use, configure, run them, after it is compiled and packaged.





What's an Application?

- An application's functionality is created by using components as parts in an ***assembly***, and wiring together their required and provided ports.
- Assemblies can be used as components in higher level assemblies, enabling an application to be used as a component in a new application.
- Assemblies are described in metadata (usually XML), *not* code.



FPGAs & Software Components

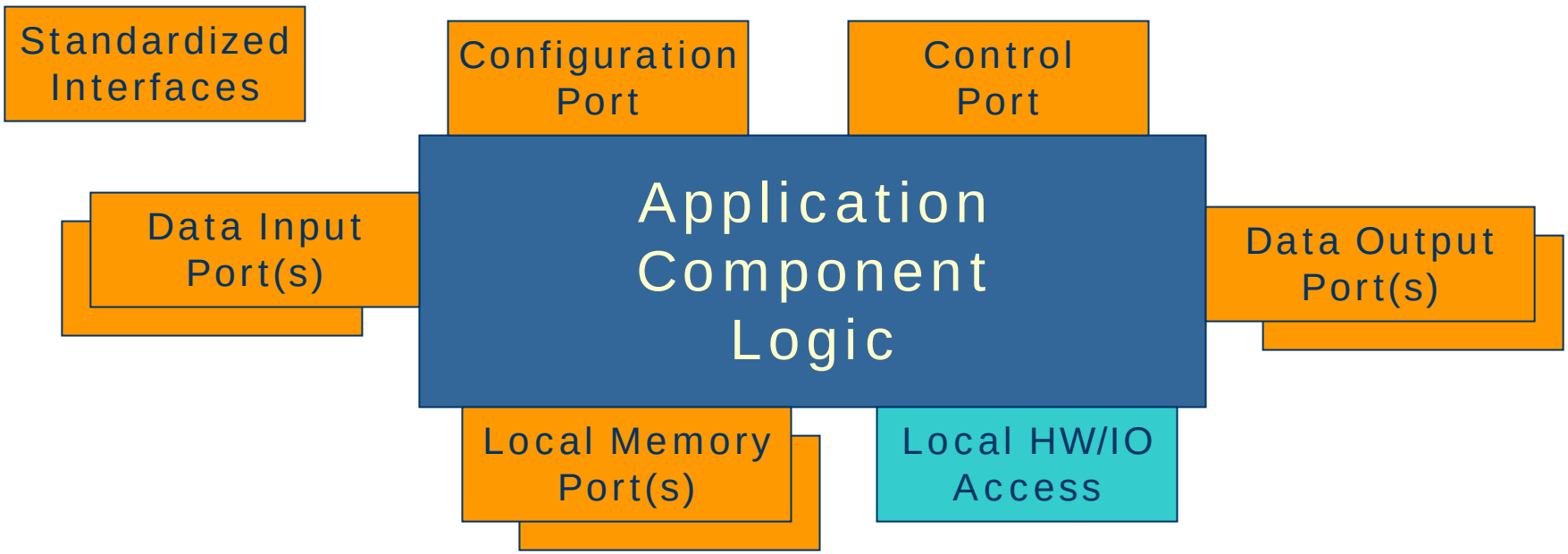
FPGA Component Model

- **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.**

FPGAs & Software Components

FPGA Component Model

- **Exposed interfaces for the VHDL designer**
  - **Local memory (scratch, LUT, or comm buffers)**
  - **Data ports for communicating with other components (FIFO style or randomly addressable comm buffers)**
  - **Runtime configuration parameters (scalars)**
  - **Execution control (start/stop/reset etc.)**
  - **Local FPGA resources or I/O (generally not portable)**



Infrastructure Elements

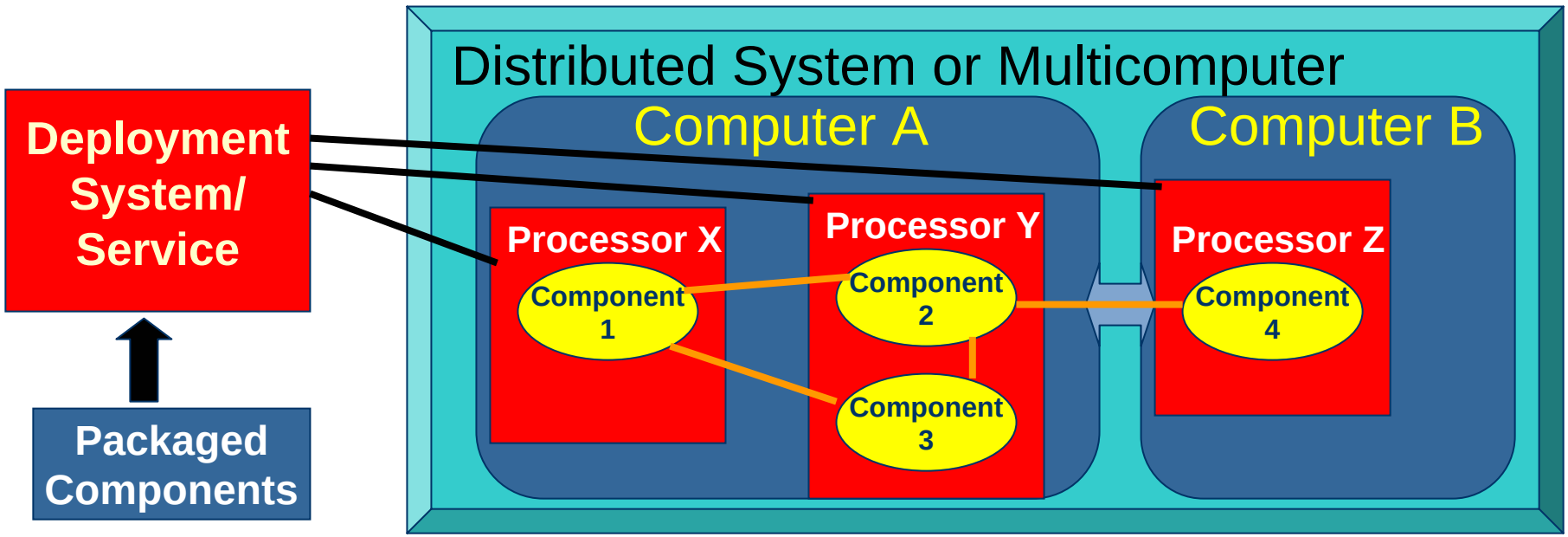
How to "bring FPGAs into the first world"?

- **A common control model and mechanisms that can work across processor classes:**
  - **Load, initialize, configure, start, stop, connect, etc.**
  - **Top level server manages a collection of processors, assuming they can all run and connect components.**

Infrastructure Elements

How to "bring FPGAs into the first world"?

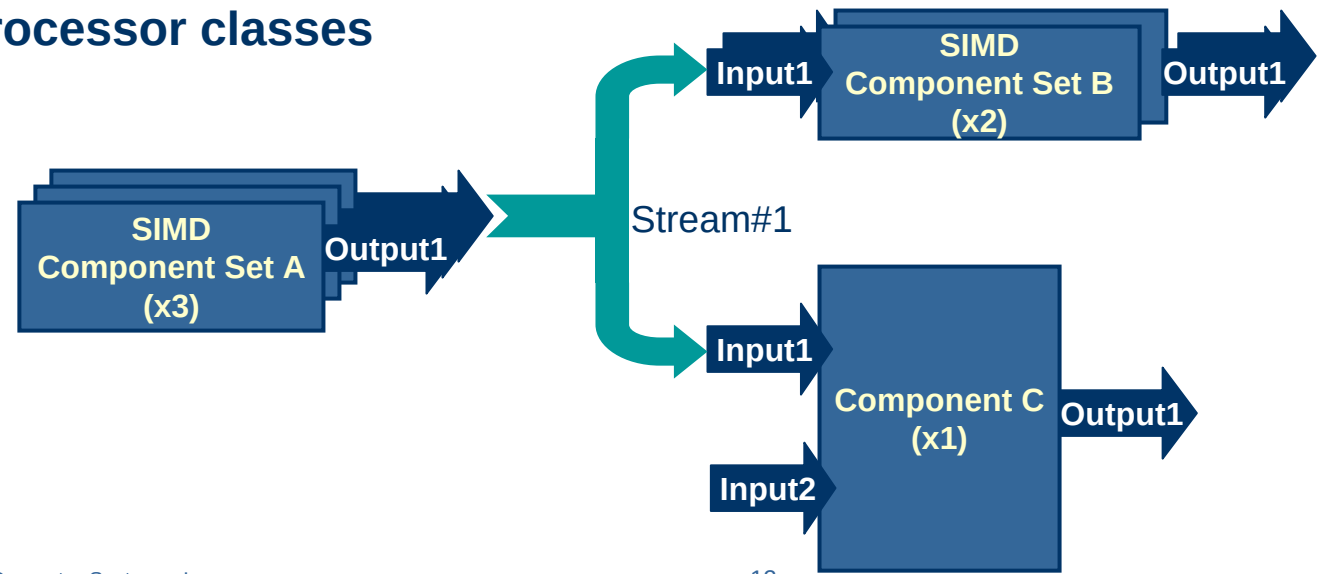
- **A control & deployment mechanism that works across processor classes:**
  - **Load, initialize, configure, start, stop, connect, etc.**
  - **Top level service manages a collection of processors, that can all run and connect components.**
  - **Each processor is self-managed or managed by proxy (FPGA).**



Infrastructure Elements

How to "bring FPGAs into the first world"?

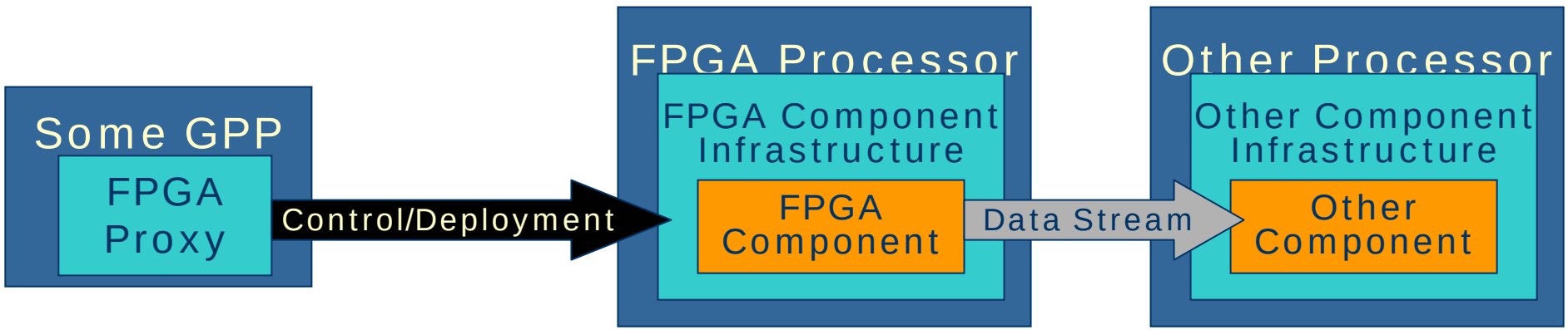
- **A data movement and synchronization model that can be supported locally on all processor classes, including FPGAs, with no central control at runtime.**
  - **Streaming data flow**
  - **Data reorg (striping/partitioning)**
  - **Request/response messaging**
  - ***Interoperable between processor classes on a fabric***
  - **Based on current standards, extended to cover a broader set of processor classes**



FPGA Infrastructure Elements

Outside-the-FPGA support software

- **The FPGA driver and proxy code to treat FPGAs as "computers than can load and run code that talks to others."**
- **Implement the common component control and deployment model for FPGAs by proxy.**
  - **Loading FPGA programs**
    - **Partial loading still a challenge with today's FPGA technologies**
  - **Configuration, control, and communication *setup*, via touching on-chip infrastructure elements**
  - **Does *not* participate in data flow or synchronization**



FPGA Infrastructure Elements

On-chip infrastructure

- **Hardware abstraction (like an OS)**
  - **Memory technology**
  - **Fabric/Bus attachment technology, with DMA**
  - **I/O technology**
- **Component abstraction (like middleware)**
  - **Configuration (runtime parameters)**
  - **Execution control**
  - **Communication with other components, local or remote**
  - **How FPGA components are written (in VHDL)**