RAMP: Architecture, Language & Compiler

ramp.eecs.berkeley.edu
Greg Gibeling
gdgib@berkeley.edu
1/10/2006

Outline

- Introduction to RAMP
- RAMP Architecture
- Target & Host Models
- Tools & Toolflow
- RAMP Description Language
- Status & Future Work

Introduction to RAMP (1)

- FPGAs as a research platfom
  - About ~25 CPUs can fit in Field Programmable Gate Array
  - 1000-CPU system from ~ 40 FPGAs?
  - FPGA generations every 1.5 yrs; 2X CPUs, 2X clock rate
- HW research community does logic design (gateware)
  - Create out-of-the-box, massively parallel system
    - Runs full OS
    - Allows OS, compiler and application developement
  - Gateware: Processors, Caches, Coherency, Ethernet Interfaces, Switches, Routers, …
  - E.g., 1000 IBM Power cache-coherent supercomputer

Introduction to RAMP (2)

- A framework for system simulation
  - Massively parallel (digital hardware) systems
  - Orders magnitude performance enhancement
  - Leverage existing designs
  - Allow community development
    - Share designs, validate experiments, etc…
- Flexible, cross platform designs
  - Requires proper structure
    - Support for automatic debugging
    - Automatic glue logic/code generation
  - Based on the “target model”

RAMP Architecture (1)

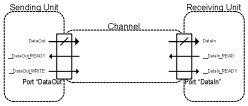
- Target
  - The system being simulated
    - Actually only a model of the system being simulated
    - Can be a cycle accurate simulation model
  - Must conform to the RAMP target model
- Host
  - The system doing the simulation
  - May include multiple platforms
    - Hardware – BEE2, XUP, CaLinx2
    - Software – Java, C, C++

RAMP Architecture (2)

- Fundamental Model
  - Message passing
  - Distributed event simulator
  - Message passing system generator
    - Cross platform
    - Shared development effort
    - Easy to develop, debug and analyze
  - Similar Work
    - Click
    - Petri Nets
    - Process Networks

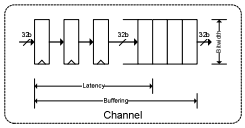
RAMP Target Model

- Units communicate over channels
- Units
  - 10,000+ Gates
    - Processor + L1
  - Implemented in a "host" language
- Channels
  - Unidirectional
  - Point-to-point
  - FIFO semantics
  - Delay Model



Target Model – Channel

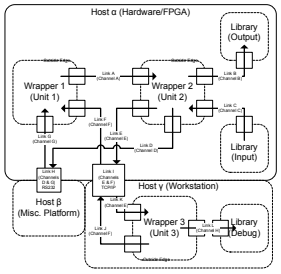
- Channel Params
  - Only used for timing accurate simulations
  - Bitwidth
  - Latency
  - Buffering
- Fragments
  - Smaller than messages
  - Convey the simulation time through idles



Target Model - Debugging

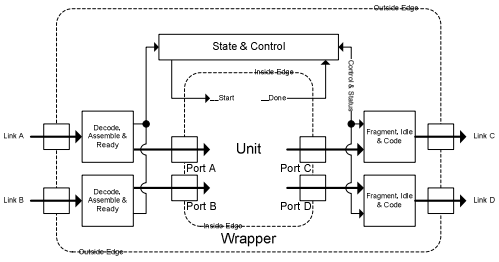
- Monitoring
  - All communication is over channels
    - Can be examined and controlled
  - Target time can be paused or slowed
- Injection
  - Makes developing test benches easy
    - Simply inject a sequence of messages
  - Cross platform comm. is hidden by RDLC
    - No simulation/reality mismatch!
    - Test benches can be written on another platform (GUI)

Host Model



- Cross platform
  - Units implemented in many languages
  - Library units for I/O
  - Links implement channels
- Links
  - Any communication
  - Not well defined

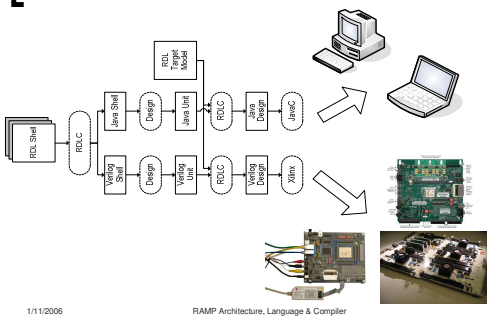
Host Model – Wrapper



RAMP Toolflow (1)

- Development Steps
  - Unit Implementation
    - RDL unit descriptions
    - RDLC generates shell code in a specific language (Verilog, Java…)
    - Researcher adds implementation code
  - RDL target design
    - Includes Mapping
    - RDLC generates complete implementation code
    - Includes all links, instantiates all unit shells

RAMP Toolflow (2)



RDL (1)

- "RAMP Description Language"
  - General message passing system description language
  - Compiler includes back-end extensibility
    - Can integrate with existing designs in many languages
- Does NOT include unit functionality
  - RDL is a "plumbing" language

RDL (2)

- Hierarchical Namespaces
  - Allows for communal development
- Target Constructs
  - Channels, Messages and Port types
  - Units include instances, inputs, outputs and connections
- Host Constructs
  - One platform per board or computer
  - Platforms include an implementation language
  - Hierarchy allows for, eg. A board with many FPGAs
- Mappings
  - Hierarchy allows for "compile one, run many"
  - Allows specific units and channels to be precisely mapped

State of the Project

- **Working hardware implementation!**
  - Compiled RDL to Verilog
  - Tested on a simple FPGA board
  - Java, BEE2, XUP should be done before Feb 1, 2006
- RDL & RDL Compiler
  - RDL is semi-stable
    - Some advanced features are in flux
    - Ready for use!
  - Working compiler, written in java
  - Powerful parser & output generators
    - Easily extensible
    - Software (Java) back end almost complete
- Documentation: http://ramp.eecs.berkeley.edu

Future Work

- RDL & RDLC Features
  - Language Features
    - Generated code
    - Port arrays
    - Compile time parameters
  - Significant additions to back end
    - Languages, platforms, links
    - Debugging Code
- Documentation
  - Architecture, Language & Compiler Technical Report
  - Complete compiler internals documentation
- Project Overhead
  - Source Control
  - Community Website

Summary

- RAMP
  - Complete emulation of ~1000 processor systems using FPGAs
- RDL
  - System netlisting language
- RDLC (The Compiler)
  - Cross platform system generator
  - Working!