RAMP: Architecture, Language & Compiler

http://ramp.eecs.berkeley.edu
Greg Gibeling, Andrew Schultz & Krste Asanovic
gdgib@berkeley.edu
2/12/2006

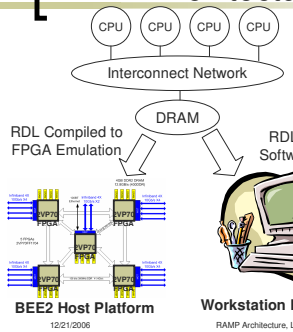
Outline

- RAMP Architecture
- Target Model
- Host Model
- Tools & Toolflow
- RAMP Description Language
- Status & Future Work

RAMP Architecture (1)

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

RAMP Architecture (2)



Target System: the system being emulated

• Structure in RAMP Description Language (RDL)

Host Platforms: systems that run the emulation

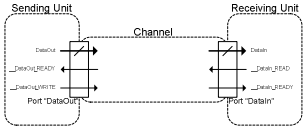
• Describe behavior of each leaf unit in favorite language (Verilog, VHDL, Bluespec, C/C++, Java)

RAMP Architecture (3)

- Fundamental Model
  - Message passing
  - Distributed event emulator
  - Message passing system generator
    - Cross platform
    - Shared development effort
    - Easy to develop, debug and analyze
  - Similar Formalisms
    - Petri Nets, Process Networks
    - Research: Click, P2, Ptolmey, Metropolis, GasP, etc….
    - Many of these can be built in RDL
      - P2 (Click) and GasP are being ported

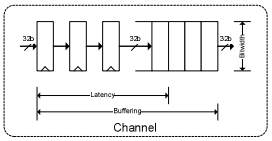
RAMP Target Model (1)

- 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

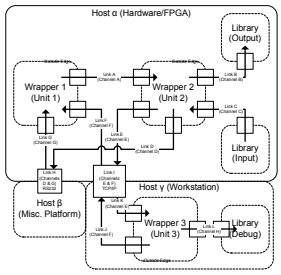
- Channel Params
  - Only used for timing accurate simulations
  - Bitwidth
  - Latency
    - Forward
    - Backward
  - Buffering
- Fragments
  - Smaller than messages
  - Convey the simulation time through idles



Target Model - Debugging

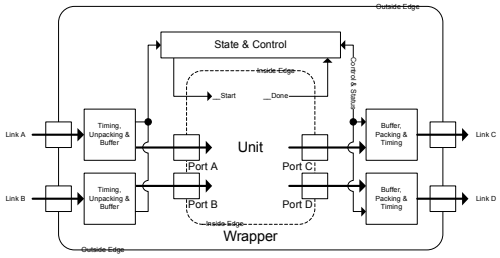
- Monitoring
  - All communication is over channels
    - Can be examined and controlled
  - Real time can be paused or slowed down
    - Target cycles are completely subjective
- Injection
  - Makes developing test benches easy
    - Simply inject a sequence of messages
  - Cross platform comm is hidden by RDLC

Host Model



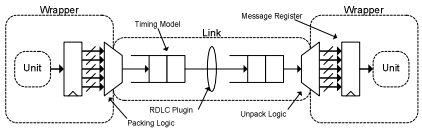
- Cross platform
  - Units implemented in many languages
  - Library units for I/O
  - Links implement channels
- Links
  - Any communication
  - Less defined

Host Model – Wrapper



Host Model - Link

- Typically Three Components
  - Packing & Unpacking
  - Timing Model
    - For cycle accurate simulations
  - Physical Transport
    - This is the only part a new link writer MUST build



RDL (1)

- "RAMP Description Language"
  - General message passing system description language
  - Two kinds of units
    - Structural – Instantiates and connects units
    - Leaf – Represents a preexisting block of verilog or java
- Compiler
  - Generates verilog & java (for now)
  - Extensible through plugins
    - Links, other toolflows, external signals

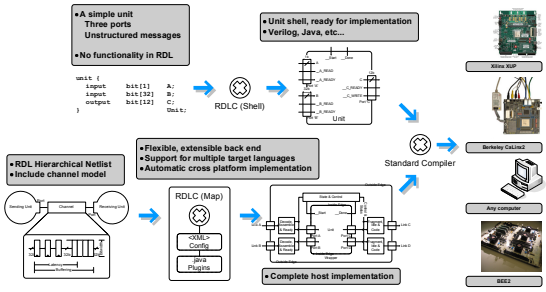
RDL (2)

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

RDLC 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

RDLC Toolflow (2)



State of the Project

- Working hardware implementation!
  - Compiled RDL to Verilog
  - Tested on CaLinx2, XUP, Digilent S3 and ModelSim SE
- Working software implementation
  - Compiled RDL to Java
  - Should be released by 2/15/2006
    - Testing and adding type checking delayed this significantly
- RDL & RDL Compiler
  - RDL is stable (Some advanced features are in flux)
  - In use
    - At graduate architecture class/seminar at Berkeley
    - RAMP Blue
    - Course project for distributed systems

Future Work

- RDL & RDLC Features
  - Language Features
    - Generated code, port arrays and compile time parameters
  - Significant additions to back end
    - Languages, platforms, links
    - Debugging automation
- Documentation
  - Architecture, Language & Compiler Technical Report
  - Complete compiler internals documentation
  - Example and Tutorials
- Automated Testing
  - Regression tests for the compiler
  - Automated test code generation for links and units