Third Time’s The Charm: Designing & Building RDLC3

http://eecs.berkeley.edu/~gdgib/
Greg Gibeling (gdgib@berkeley.edu)
Advisor: John Wawrzynek
1/10/2008

Quick Introduction to RDL

- The “RAMP Description Language” (RDL)
  - Hierarchical structural netlisting language
  - Describes message passing distributed event simulations
  - System level: contains no behavioral spec.

- Tradeoffs
  - Costs
    - Use of the RDF Target Model (sort of)
    - Area, time and power to implement this model
  - Benefits
    - Abstraction of locality & timing of communications
    - System debugging & power tools
    - Determinism, sharing and research
  - Goal: trade costs for benefits as needed

RAMP Architecture

- RAMP Design Framework (RDF)
  - Restrictions on how systems are built
  - Not to be confused with RDL
- Target
  - The system being emulated
  - Must conform to the target model
  - Emulation, not implementation
- Host
  - The system hosting the emulation
  - May include multiple platforms
    - Hardware – BEE3, BEE2, XUP, ML500, CalLinx2+, S3, DE2
    - Emulation – Matlab, ModelSim
    - Software – C/C++, Java

RDF Target Model (1)

- Units
  - RDF: 10,000+ Gates
    - Processor + L1
    - Router/Switch
    - Implemented in a “host” language
- Channels
  - Unidirectional
  - Point-to-point
  - FIFO semantics
  - Delay Model

RDF Target Model (2)

- Inside edge
  - Ports connect units to channels
    - FIFO/SR signaling
    - Hardware or Software
  - Target cycle control
    - __Start & __Done
    - __Enable
    - Can vary from unit to unit
  - Host Level Time Sharing

RDF Target Model (3)

- Channel Params
  - Only used for timing accurate simulations
  - Bitwidth (CBFC/SR/Bit)
    - Latency
    - FW & BW (CBFC)
    - BW (SR)
    - Buffering (CBFC)
- Fragments
  - Smaller than messages
  - Indivisible message piece, which can be carried by a channel
  - May never exist in implementation
**RDF Target Model (4)**

- Simple CBFC Example
  - Channel parameters <8, 2, 1, 3>
  - A single 1 bit Message

**RDF Target Model (5)**

- Distributed development

**RDF Target Model (6)**

- Buses
  - A result of I/O pin & board limitations
  - Not used in new designs (HyperTransport, PCI, AMBA AVX)
  - Not implementable on FPGAs
  - Conclusion
    - Buses do not scale & are not distributed
    - The target model adds research and design value
    - Avoid buses in the short term, avoiding buses by design in the long term
  - 0 latency, 0 buffering channels (Wires)
  - Does not fit RDF, RDL allows it
  - Requires additional assumptions for cross-platform support or debugging tools
  - Possible severe performance reduction
  - System as a unit
  - Unit as a system

**RDF Target Model (7)**

- RDF: Non-universal Model
  - Lossy channels, multicast networks and busses modeled as units
  - No global reset
  - Emulation & abstraction are not free
  - Time, area and power are all spent
  - Particularly noticeable for DSP or control-free systems
  - Existing Systems
    - Can be used as a single unit
    - May be split, but this will require design changes

**RDF & RDL**

- RDF
  - The RAMP Design Framework
  - Requires latency insensitivity
  - System composition
  - Distributed development
  - Performance based research

- RDL
  - The RAMP Description Language
  - More general, easier to implement
  - Support for CBFC/SR/Wire channels
  - Support for 0 latency, 0-delay
  - Fixed latency designs (e.g. DSP perhaps)

**RDF/RDL & HASIM**

- Identical Base Models
  - Based on dataline semantics
  - RDL SR Channel + HASIM A-Port
  - Effectively distributed event simulators

- Differences
  - Generalization & Sharing
    - RDL Supports multiple languages
    - RDL Designed to cover software & hardware
    - RDL Includes CBFC, SR and Wire channels (A-Port + SR)
    - HASIM Published (Soon)
  - Soaking
    - RDL designed for multiple FPGAs & boards
    - Automatic mapping & resource abstraction

- Conclusions
  - HASIM is more focused, RDL is very general
  - RDLC3 should “play well” with HASIM
Avoid the issues that plagued RDLC2 at an algorithmic level

Any sufficiently large software project needs its own libraries

Too many little projects

Main Issues

Abstraction of resources

FPGA/Computer System "Loader"

Simple RISC processor

Better platform abstraction (virtualization & multiplexing of resources)

Support for complete processor units

New compiler core (parameterization, cross-platform support)

NP complete problem

EECS150: General Media Project

Integrates for monitoring & debugging

RDLC2 2007.8.13

Dataflow Style

Loading & debugging support

Teaching CS61C absorbed fall semester

BEE2 Support/Interchip Link

Automated Mapping

Teaching CS61C absorbed fall semester

BEE2 TestBed

Automated Mapping

Network Router w/Timing Model

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

Automated Mapping

BEE2 TestBed

AutomatedMapping
**RDL MetaProgramming**

- RDL in RDL
  - Shared development of host & target networks
  - Similithia Compiler!
  - Target & host are similar
  - Recursive debugging
- Generating RDL
  - Inline scripting
  - Limited to mathematical expressions of RDL
  - Fully parameterized
  - Java plugins
  - RDL-reflection interface
  - Already done in FLEET
- Adding better support

**Debugging & Parameterization**

- Untimed Channels
  - Untimed – no timing model
  - Optional – needn’t be used
  - Piece of inspiration between target & host
- Plugin-defined protocols
  - Memory Mapped
  - Raw bus (SPI)
- Extensions to the RDL type system (Java objects on disk)
- Debugging
  - Core level tool level network or RDL levels
  - RADTools & R2
- Parameterization
  - Not all known until load time
  - MAC Address, speed, etc.
  - Important for design minimization

**RADTools**

- Abstract Management
  - Location
  - Data structures
- Loading
  - File programming
  - Load-time parameterization
- Debugging, Tracing & Monitoring
  - Uniform Interface
- Research-based sharing
- Structures
  - Composition (Hierarchy)
  - Dependency (Platforms)
- Management (Infrastructure)
- Communication
- Framework
  - Built-in various specific plugins
  - Integrated with RDL2
  - Not overlaid in RDL designs
  - Failure management
- Class project with Butzin
  - Lisa Gutnik

**R2 – Debugging & Monitoring**

- Active Debugging
  - Datatype style declarative
  - Allows integrated monitoring & simulation
  - Simple to specify & modify
  - Can support dynamic changes in debug rules
- The R2 Project
  - Used a language (overload) to build PDP networks
  - Previously adapted to RDL2
  - Now in R2
  - Class project with Dominik
  - Schmitt & Nathan Butzin
  - Will need rewrite

**Automated Mapping (1)**

- Mapping
  - Units to platforms
  - Graph Embedding
    - Minimize cost in compile time, run time and resources
    - Many requirements
    - Largely mechanical (easy) for a ‘good’ design
  - Hierarchical Analysis
    - Heuristic based
    - Reduces problem size
    - History of hierarchical partitioning papers

**Automated Mapping (2)**

- Differences: PARCAD
  - Performance is flexible
  - Scale is vastly different
  - Channels are heavy-weight
  - Requires fast turn-around
  - Requires design minimization (30 Hour PAR)
- NP Complete
  - Currently have an IP formulation
  - Simplex solver (for now)
  - Needs testing
RDLC3 Promises

- Promises
  - RDLC3 Examples (ASPLOS): MIPS, Router, Base Examples
  - Full timing models (June)
  - Multi-FPGA platform support (ASPLOS)
  - RADTools (ASPLOS)
  - Platforms: BEE3 (ASPLOS), ModelSim (ASPLOS)
  - Links: Ethernet (June), InterChip, XAUI (June)
  - Languages: Verilog (ASPLOS), Java (June)
- Support, Documentation & Development Help
- Hopes
  - Platforms: Calinix2+, S3, DE2, XUP, BEE2
  - Links: Serial, TCP/IP
- Simple Suggestions
  - The Many Meanings of RAMP
  - The Website
  - Planning - Let the world know what we intend to do!
  - People - Who is everyone?
- Let down.
  - Integration: EDK, Eclipse, etc...
  - Host level networking & memory controllers (I’ll need these though)
  - A complete RAMP system (too ambitious)

RAMP: A Team Project (1)

- Critics of RAMP
  - Definition of Terms
  - Long-term vision
  - Cooperation & integration
- Simple Suggestions
  - The Many Meanings of RAMP
  - The Website
  - Planning - Let the world know what we intend to do!
  - People - Who is everyone?
  - Central/Shared CVS or SVN
  - Let’s get the ball rolling...

RAMP: A Team Project (2)

- Recruiting
  - Who wants to use RDL?
    - I certainly do...
  - Anyone interested in helping to develop RDLC3?
    - Get some Berkeley undergrads
- Discussion of RDL & Features
  - Missing features?
    - In particular, ones which haven’t been mentioned
  - Design concerns?
    - Does the model match what you want?
    - “Never ignore the possibility that you may be completely wrong”
  - Projects which could be examples?
    - I’ll help to the work to put them into RDL
    - Be a man: share your work with the RAMP project!