Third Time's The Charm: Designing & Building RDLC3

http://eecs.berkeley.edu/~gdgib/
Greg Gibeling (gdgib@berkeley.edu)
Advisor: John Wawrzynek
1/16/2008

Quick Introduction to RDL

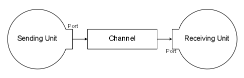
- The "RAMP Description Language" (RDL)
  - Hierarchical structural netlisting langauge
  - Describes message passing distributed event simulations
  - System level: contains no behavioral spec.
- Tradeoffs
  - Costs
    - Use of the RDF Target Model (sort of)
    - Area, time and power to implement this model
  - Benefits
    - Abstraction of locality & timing of communications
    - System debugging & power tools
    - Determinism, sharing and research
  - Goal: trade costs for benefits as needed

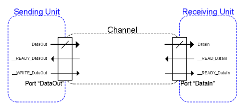
RAMP Architecture

- 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, CaLinx2+, S3, DE2
    - Emulation – Matlab, ModelSim
    - Software – C/C++, Java

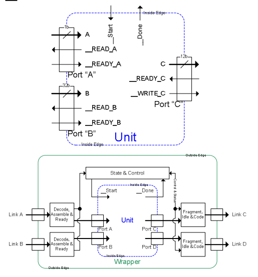
RDF Target Model (1)

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





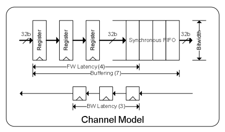
RDF Target Model (2)

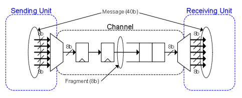


- Inside edge
  - Ports connect units to channels
    - FIFO/SR signaling
    - Hardware or Software
  - Target cycle control
    - __Start & __Done
    - __Enable
    - Can vary from unit to unit
    - **Host Level Time Sharing**

RDF Target Model (3)

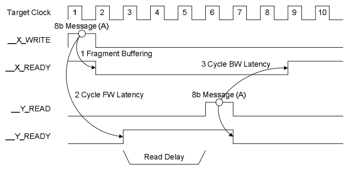
- Channel Params
  - Only used for timing accurate simulations
  - Bitwidth (CBFC/SR/Wire)
  - Latency
    - FW & BW (CBFC)
    - FW (SR)
  - Buffering (CBFC)
- Fragments
  - Smaller than messages
  - Indivisible message piece, which can be carried by a channel
  - May never exist in implementation



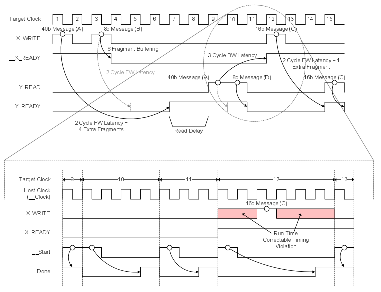


RDF Target Model (4)

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



RDF Target Model (5)



RDF Target Model (6)

- Busses
  - A result of I/O pin & board limitations
  - Not used in new designs (HyperTransport, PCIe, AMBA AHX)
  - Not implementable on FPGAs
  - Conclusion
    - Busses do not scale & are not distributed
    - The target model adds research and design value
    - Adapt busses in the short term, avoiding busses 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 dataflow 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)
  - Scaling
    - 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

2007 (1) – RDLC2

- **RAMP Blue in RDLC2 (Beta)**
  - By Alex Krasnov, Jue Sun & Greg Gibeling
  - Main Issues
    - Xilinx MicroBlaze
    - EDK Integration with RDL
  - RDLC2 2007.8.13
    - Mostly minor changes (a few bugs, a few scaling issues)
    - Added some significant features (ResetParam & Arrays, Regexs)
- Board Level Links
  - BEE2 Support/Interchip Link
    - Apparently been working quite well since ISCA/FCRC
  - BEE2 TestBed
    - Developed by Tracy Wang
    - Tested the interchip link
    - Will shift to BEE3 very soon
    - Basis of automated link reliability testing

2007 (2) – RDLC3

- Automated Mapping
  - Complete the theoretical work
  - NP complete problem
  - Similar to PAR in FPGAs, but at a higher level
  - Currently unpublished, waiting on implementation
- Framework Code for RDLC3
  - Any sufficiently large software project needs it's own libraries
  - Avoid the issues that plagued RDLC2 at an algorithmic level
    - Parameters, multi-board support, more complex links
    - RDLC2's internal design is very brittle and complex
    - RDLC3's internal design is general & flexible
  - Finally winding down
- Greg's Brain Scattered
  - Too many little projects
  - Teaching CS61C absorbed fall semester
- **Finished RDLC3 Architecture**

2008

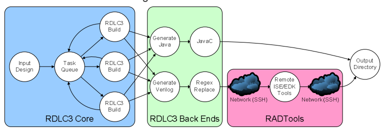
- RDLC3 Released
  - New compiler core (parameterization, cross-platform support)
  - Support for complete channel timing model
  - Automated Mapping
  - Better platform abstraction (virtualization & multiplexing of resources)
- RADTools Released
  - FPGA/Computer System "Loader"
  - Integrates for monitoring & debugging
- BEE3 Support Package (for RDLC3 & RADTools)
  - Abstraction of resources
    - Links: InterChip, Ethernet, XAUI
    - Memory
  - Loading & debugging support
- Example Projects
  - CounterExample
  - Simple RISC processor
  - Network Router w/Timing Model
  - EECS150: General Media Project

Design Sharing

- Design Sharing Goals
  - Share individual models
  - Share libraries, back-ends, links, etc…
  - Allow for independent validation
- Solution
  - Switch to an RDL unit as a packaging unit
    - Similar to PCores or Java classes
    - Protection
    - Namespace Isolation
    - Attached scripts & implementation files
  - Integrated repository access
    - Could have RDLC3 access an HTTP, CVS or SVN repository

RDLC3 Architecture

- RDLC3 Core
  - Lex, Parse, Parameterization, Mapping, Output Selection
  - Allows plugins to significantly modify build process
    - Hardware generators
    - Regex replacement
    - Compiling software for processor units
  - Dataflow Style
    - Integrates with RADTools for distributed build (standard build environments)
    - Allows multi-threading

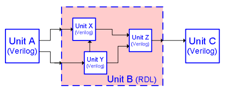


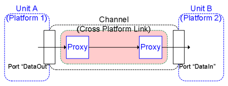
Virtualization & Sharing

- Units
  - One unit implementation supports multiple units
  - Wrapper interface plugins
    - Not just start/done
    - Multiple unit implementation timing interfaces
    - Pipelining of target cycles
    - Time sharing of unit implementation
    - Enhanced functional/timing split
- Links
  - One link supports multiple channels
  - Host level: simple message multiplexing
  - Target level: enable unit implementation sharing

RDL MetaProgramming

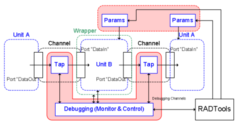
- RDL in RDL
  - Shared development of host & target networks
  - **Simplifies Compiler!**
    - Target & Host are similar
  - Recursive debugging
  - Zero-Delay Links
- Generating RDL
  - Inline Scripting
    - Limited to mathematical expressions at first
    - Could be generalized
  - Java Plugins
    - RDL Reflection Interfaces
    - Already done in FLEET
    - Adding better support



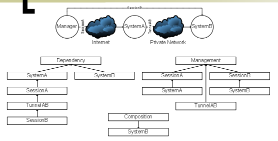


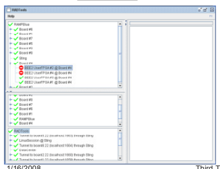
Debugging & Parameterization

- Untimed Channels
  - Untimed – No timing model
  - Optional – Needn't be used
  - Point of interaction between target & host
- Plugin defined protocols
  - Memory Mapped
  - Raw bits (GPIO)
  - Extensions to the RDL type system (Java objects or bits)
- Debugging
  - Can use host-level network or dedicated links
  - RADTools & R2
- Parameterization
  - Not all known until load time
  - MAC Address, Speed, etc…
  - Important for design minimization



RADTools

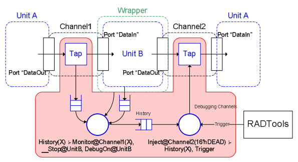




- Abstract Management
  - **Loading**
    - FPGA Programming
    - Load time parameterization
  - Debugging, Tracing & Monitoring
  - Uniform Interface
  - **Research resource sharing**
- Structures
  - Composition (Hierarchy)
  - Dependency (Platforms)
  - Management (Infrastructure)
  - Communication
- Framework
  - Based on service specific plugins
  - Integrated with RDLC3
  - Not restricted to RDL designs
  - Failure management
  - Class project with Nathan Burkhart & Lilia Gutnik

R2 – Debugging & Monitoring

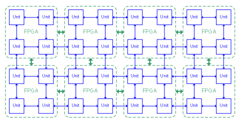
```
stream Stream0(Int, Int);
stream Stream1(Bool);

watch Stream0;
watch Stream1;

Stream1(true);
Stream0(0, 1);
Stream0(2, 3);
Stream1(false);
```

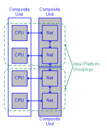


- Active Debugging
  - Datalog style declarative event language
  - Allows integrated monitoring & injection
  - Simple to specify & modify
  - Can support dynamic changes in debug rules
- The P2 Project
  - Used a language (overlog) to build P2P networks
  - Previously adapted to RDLC2
    - Ran on BEE2
    - Class project with Andrew Schultz & Nathan Burkhart
    - Will need rewrite

Automated Mapping (1)

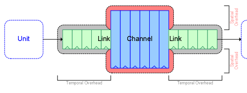
- Mapping
  - Units to platforms
  - Channels to links
- Graph Embedding
  - Minimize cost in compile time, run time and resources
  - Many requirements
  - Largely mechanical (easy) for a "good" design
- Hierarchical Analysis
  - Heuristic based
  - Reduces problem size
  - History of hierarchical partitioning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

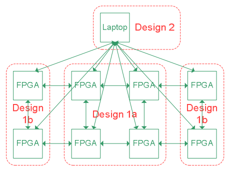




Automated Mapping (2)

- Differences: PAR/CAD
  - 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



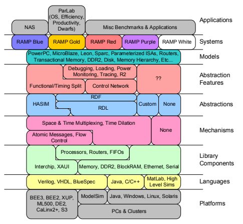


RDLC3 Promises

- Promises
  - RDLC3 Examples (ASPLOS/June): MIPS, Router, Base Examples
  - Full timing models (June)
  - Multi-FPGA platform support (ASPLOS)
  - RADTools (ASPLOS/June)
  - Platforms: BEE3 (ASPLOS/June), ModelSim (ASPLOS)
  - Links: Ethernet (June), InterChip, XAUI (June)
  - Languages: Verilog (ASPLOS) & Java (June)
  - **Support, Documentation & Development Help**
- Hopes
  - Platforms: CaLinx2+, S3, DE2, XUP, BEE2
  - Links: Serial, TCP/IP
  - Languages: VHDL, BlueSpec, C/C++
- Letdowns (I Call Not-It)
  - 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)

- Criticisms of RAMP
  - Definition of Terms
    - RAMP: Project, System…
  - 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 Access
  - Lets get the ball rolling…


NAS
ParLab (OS, Efficiency, Productivity, Dwarfs)
Misc Benchmarks & Applications
Applications
RAMP Blue
RAMP Gold
RAMP Red
RAMP Purple
RAMP White
Systems
PowerPC, MicroBlaze, Leon, Sparc, Parameterized ISAs, Routers, Transactional Memory, DDR2, Disk, Memory Hierarchy, Etc...
Models
Debugging, Loading, Power Monitoring, Tracing, R2
Functional/Timing Split
Control Network
??
Abstraction Features
HASIM
RDF
RDL
Custom
None
Abstractions
Space & Time Multiplexing, Time Dilation
Atomic Messages, Flow Control
None
Mechanisms
Processors, Routers, FIFOs
Interchip, XAUI
Memory, DDR2, BlockRAM, Ethernet, Serial
Library Components
Verilog, VHDL, BlueSpec
Java, C/C++
MatLab, High Level Sims
Languages
BEE3, BEE2, XUP, ML500, DE2, CaLinx2+, S3
ModelSim
Java, Windows, Linux, Solaris
PCs & Clusters
Platforms

RAMP: A Team Project (2)

- Recruiting
  - Who wants to use RDL?
    - I certainly do…
  - Anyone interested in helping to develop RDLC3?
    - Got 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/do-the-work to put them into RDL
    - Be a man: share your work with the RAMP project!