RAMP – RDL Bootcamp

Compiled & Edited: Eric Chung
Authors: Asif Khan, Eric Chung, Marghoob Mohiyuddin, Andrew Putnam, Greg Gibeling, Adrian Caulfield, Eriko Nurvitadhi, Zhangxi Tan, Hari Angepat, Shobana Padmanabhan, Muralidaran Vijayaraghavan, Sewook Wee

Austin, Texas
March 21-25th, 2007

Activities

- Day 1
  - RDL intro, tool setup
- Day 2
  - LEON intro, tool setup
  - Did not work "out-of-the-box"
- Day 3
  - Unit wrapping discussion and design
  - LEON self-contained in Modelsim (ok), in ISE (fail)
  - Stripped LEON to minimal system (1-cpu, bus, rom)
- Day 4
  - Continued unit wrapping discussion and design
  - Coding and testing of Master AMBA-to-Transport adapter (ok)
- Day 5
  - Documentation and retrospection

LEON

- "Out-of-the-box" issues:
  - Bad bitstreams (e.g., ISE8.2 ok, ISE8.1 fail)
  - On some setups, Makefiles unreliable
    - Would be nice to have "./configure" option

IP wrapping

- Goals:
  - Reusable AMBA IP units
  - Generic transport format for communication
  - Avoid modification of IP cores
  - Describable in RDL

Challenges (1/2)

- How to abstract away legacy IP iface behavior?
  - Reconciling atomic snoop-buses w/ distributed messages
  - Stateful multi-cycle transfers ←→ messages
  - General enough to connect PLB IP w/ AMBA IP?
- Tightly coupled interfaces
  - AMBA master and slave protocol on LEON3
  - Ex: DDR ctrl perf. optimization for AMBA (e.g., bus pred.)
- Message granularity
  - Encapsulate logical or physical bus transaction?
  - How to handle arbitrary burst sizes and types?

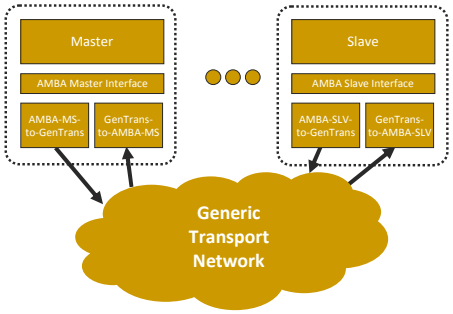
Challenges (2/2)

- Performance implications
  - Abstraction change not free
    - Bus transaction to message conversion costs cycles
  - Need to quantify better w/ workloads, experiments

Design: Generic Transport

- Message format for bus transactions
  - Converts bus transactions to messages
  - Ideally: standardize over multiple bus prots
  - First version: AMBA-specific
- Making AMBA compatible with Transports
  - Write conversion adapters **once** for master- and slave-side IP
  - Future in-house IP simply generate transports

Design: RAMP adapter model



Implementation

- Master-side AMBA-to-Transport converter
  - Single-target word transactions only
  - Burst-mode unsupported
  - FIFO interfaces for channels
  - 100L of Bluespec
- Testbed in Modelsim simulation
  - 1-CPU LEON with two-way channel to simple RAM model
  - Successfully fetches instructions and accesses RAM
- What's left:
  - Integrate with RDL
  - Slave-side AMBA-to-Transport converter (for DDR, etc)
  - Burst-mode, large-message support
  - CPU snoop, interrupt channels

Lessons learned

- Tool uniformity is essential
  - We had: diff OSes, diff tool versions, licensing
  - In the future: advanced setup of identical, working tools via VMs, remote server, etc.
    - Modelsim, Synplicity, ISE, Linux, Java, RDL, Licenses
- Clearer IP requirements necessary
  - What do researchers need?
  - What are basic building blocks?

Future collaboration

- Prerequisites for bootcamps
  - Get familiar with tools **ahead** of time
- RAMP white requirements
  - Need concrete specs for func & timing model
    - How many MIPS in the platform?
    - How many target & host processors in system?
  - Supporting software stack
  - Identify IP building blocks

Conclusion

- Wrapping of existing IP non-trivial
  - Requires design knowledge of IP interfaces (no way to cheat here)
  - How to balance engineering vs. research?
- Adapter model
  - Necessary to avoid re-coding individual IPs
  - Forward compatibility with new IPs
  - Simple assembly test cases OK
  - What's left to do: slave adapter, burst-mode support
  - Supporting other prots: PLB, Wishbone, OpenSPARC

Bootcampers

From left: Asif Khan, Eric Chung, Marghoob Mohiyuddin, Andrew Putnam, Greg Gibeling, Adrian Caulfield, Eriko Nurvitadhi, Zhangxi Tan, Hari Angepat, Shobana Padmanabhan, Muralidaran Vijayaraghavan (not shown: Sewook Wee)

ModelSim Libraries

- compxlib –s mti_se –arch virtex2p
- Setup
  - Right click in the libraries pane
  - Add Library
  - Add Mapping to Existing Library
- Libraries
  - xilinxcorelib_ver
  - unisims_ver
  - simprims_ver
  - All located in C:\Xilinx\Verilog\mti_se

Counter Example

- java –jar rdlc2.jar –gui
- Input File: CounterExample.rdl
  - From RDLC2 Examples ZIP file
- Output Dir: C:\Temp or similar
- Root Dynamic ID
  - ::Maps::ModelSim
  - ::Maps::XUP
- Enable BackEnd Plugins