# An Approach to Verification of Many-Core Systems Using the Virtual Platform

Victoria Mitchell Altera Corporation



MEASURABLE ADVANTAGE™

#### Topics

- Introduction
- Tools
- Setup
- Findings



Previously, on this program...

**Extending the life of the Virtual Platform:** 

- Earlier Verification
- Architectural Decisions
- Design Verification (visibility)
- DFX Methodology

... not just early enablement of software, but true simultaneous engineering

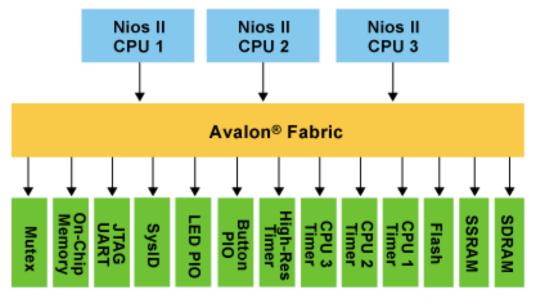


#### Why Many-Core?

#### 1. Parallelization of tasks

#### 2. Using cores as repeatable templates of custom logic

- Modify functionality over time
- Faster to design
- More automatic and reproducible
- Ultimately more configurable





#### Why Extend The Virtual Platform?

- 1. Cost effective to leverage investment
- 2. Develop once, use in many applications
- 3. Improves overall quality of HW
- 4. Improves overall quality of system

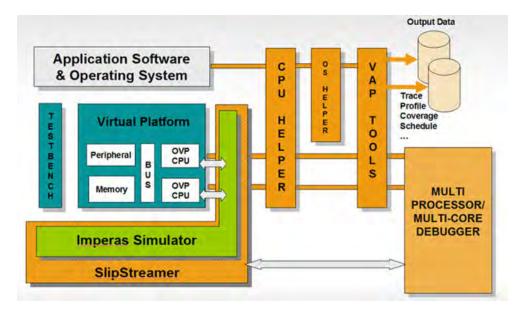
Enabled by unique capability of the VP

- Instrumentation of platform: models, peripherals
- Intercepting simulation with hosted functions
- Non-intrusive
- Verify the full system



#### **Tools Used: Processor Simulation and Interception**

#### Translates instructions to host native





- Dynamically builds translation lookup
- Peripheral models execute in quantum measure of time



#### **How Binary Intercepts Work**

- Dynamic loadable modules
- APIs are registered to events

#### Examples of events:

- simulation construction
- model enumeration
- before or after an instruction morph
- after 1-N instructions
- when address is executed
- when data address range is accessed
- programmers view events

- May be opaque or transparent
- What you can do with intercept APIs
  - inspect memory
  - drop into debugger of simulation, or code
  - alter translation
  - change processor state
  - evoke other APIs
  - add/remove other API callbacks



#### Imperas M\*SDK and VPA API

Operating System Bare Metal Apps & Middler Platform (e.g. Drivers) Processor Trace coprocessor registers Trace TLB trace exceptions Trace modes Trace service calls Trace service calls Trace hypervisor calls Trace secure monitor calls Trace MT/MP extensions Trace system calls Trace timer	Multi Processor Debug Address space introspection Virtual2physical mapping Print CP registers TLB dump Break on exception Break on mode Break on register change Break on instruction	Bus connectivity view Peripheral register view Peripheral src debugger Processor freeze control Trace peripheral access Memory coverage Shared memory checks	Break on line Break on function call Elf introspection Unlimited HW breakpoints Memory region watchpoints Trace source line Trace context Trace functions Line Coverage	Trace console Trace execve Trace scheduler Trace tasks Trace module loads Trace printk
Trace cache instructions Trace SIMD extensions Trace instruction Trace register change	Instruction coverage Instruction profiling Fault Injection Cache analysis		Function profiling Heap checks Stack checks Malloc checks Semaphore checks	

#### Use cases include

- Drivers
- Firmware
- Assembly libraries
- OS porting and bring up
- Hypervisors

#### Tool features

- Multiprocessor, multicore, multithread, multi-everything
- Non-intrusive
- Low overhead (high performance sim)
- User extendable



#### **Our Platform Setup**

#### Matrix network of NIOS-II soft cores

- 1 "master"
- 2 N cores
- many topologies

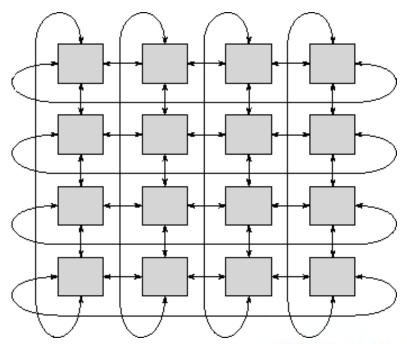
## NIOS-II OVP model for nodes

## Node functionality

- initialization
- address negotiation
- data packet handling

## Platform construction/ assembly

specify # NIOS and topology





#### **Platform With Direct Verification**

## Platform runtime w/ code embedded in NIOS-II SW

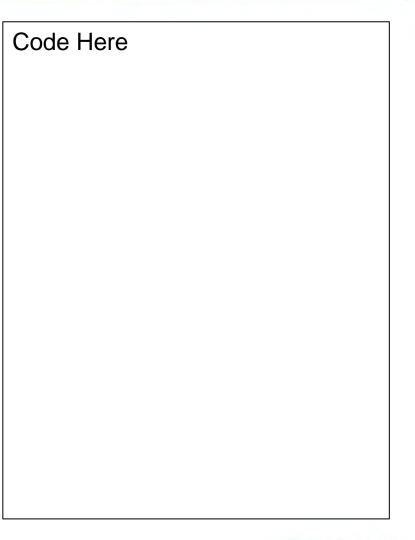
- Sendmsg() function
- Acknowledge() function
- Uses printf()
- Compile-time switches to enable
- Changes execution path and size of the code

Code Here



#### **Platform With Intercept Library**

- Same platform, but with "production-ready" NIOS firmware
- Intercept sendmsg() and acknowledge()
- Registered callback at memory access





#### **Positive Feedback**

- Use of intercepts eliminate need to change NIOS-II microcodes
- Validation engineers hook into intercept functions where they would normally write code for a directed test
- Intercept functions have simple CLI that is scriptable onto test bench
- AV is confirmed for addressing, topology, transactions
- DV validates on the system level
- eSW focuses on production code





#### **Improvements Realized**

#### **Speed of Design**

- FW starts with test-bench setup
- Closed-loop architecture verification

#### **Test Coverage**

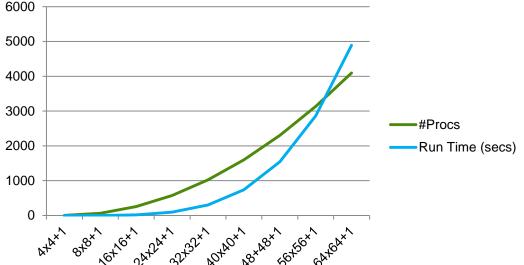
- RTL and FW in sync
- 100% production
  microcode coverage



#### **Still Work to be Done**

# 1. Separate instances of the intercept library per processor

- Memory impact
- Duplication and sync
- 2. Execution speed vs number of cores



3. Enhance  $A^{A^{*}} \otimes A^{B^{*}} \otimes A^{B^{$ 



#### **Extending the Software Virtual Platform:**

- Doesn't replace timing analysis or characterization of the design, but ...
  - Improvements are well worth the negligible extra effort
- We plan to continue to use thru product development lifecycle of complex, many-core systems





## **Thank You**



MEASURABLE ADVANTAGE M

© 2014 Altera Corporation-Public

ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www altera com/legal