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

Victoria Mitchell, Jeff DaSilva
Altera Corporation

ALTERA
MEASURABLE ADVANTAGE™

Topics

- Introduction
- Tools
- Setup
- Findings

ALTERA
MEASURABLE ADVANTAGE™

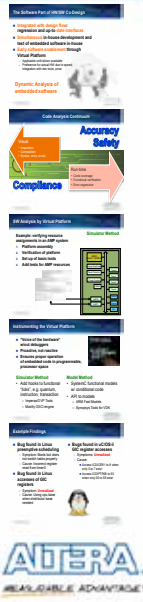
Introduction

Previously, on this program…

Extending the life of the Virtual Platform:

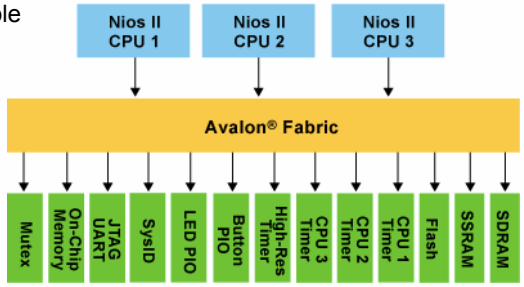
- **Earlier Verification**
- **Architectural Decisions**
- **Design Verification (visibility)**
- **DFX Methodology**

… 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**
   - Finite-state machine replacement
   - Modify functionality over time
   - Faster to design
   - More automatic and reproducible
   - Ultimately more configurable
   - Easier to test ?



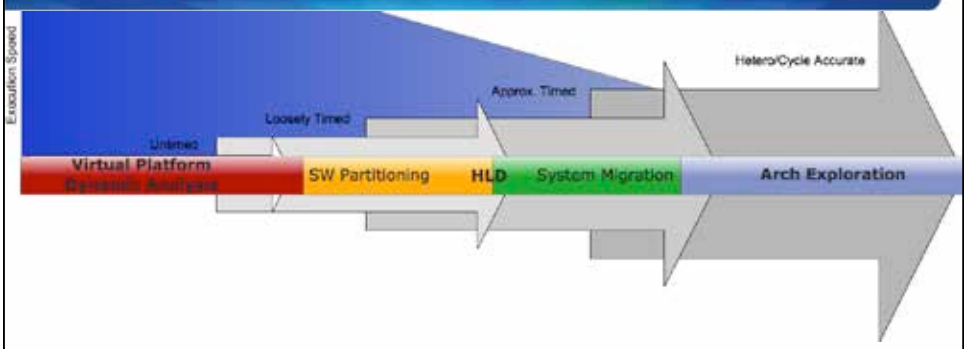
Why Extend The Virtual Platform?

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**

Virtual Platform Options on the System Modeling Spectrum



eSW, FW, DV, AV:
- Co-Design
- Internal eSW, FW
- Early Enablement
- SW, Design, Arch Verification

SW Exists:
- Profiling
- Acceleration Annotation

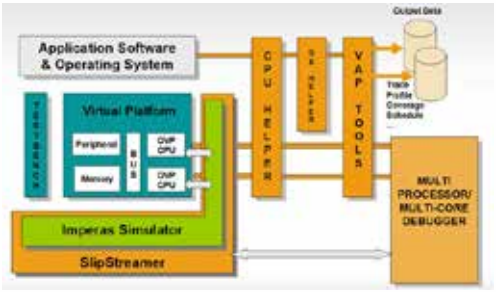
No SW:
- Timing Studies
- System Partitions

Hardware Spec:
- Architectural Design
- Verification to Constraints
- IC Specification

6

Tools Used: Processor Simulation and Interception

- **Translates instructions to host native**



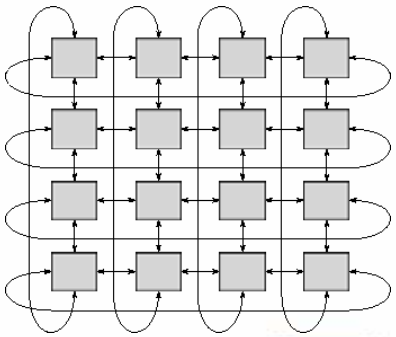
- **Dynamically builds translation lookup**
- **Peripheral models execute in quantum measure of time**

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

Our Many Core Platform Setup

- **Configurable matrix network of Nios II systems**
  - 1 "master"
  - 2 - N cores
  - many topologies
- **Nios II OSCI TLM 2.0 wrapped OVP model for nodes**
- **Node functionality**
  - initialization
  - address negotiation
  - data packet handling
- **Platform construction/ assembly**
  - specify # Nios II's 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 and machine state, too**

```
void SendMsg(int x,int y)
{
  packet[0] = cfHeader(x,y,MT_BLK_LEFT,3,SRC_SDM);
  packet[1] = LEFT_BLK_1;
  packet[2] = LEFT_BLK_2;
  packet[3] = LEFT_BLK_3;
  if (VERIF_MODE) {
    printf("-I:SDM: XMT BLK LEFT data packet to
node(%s)\n", getNodeAddrStr(x,y));
  }
}


void AckMsg(int thisId,int header)
{
  int header  = packet[0];
  int thisId  = getIdFromXY(header);

  if (VERIF_MODE) {
    printf("-I:LSM(%s): ACK BLK LEFT data packet\n",
           getNodeAddrStrFromHXY(getMsgXY(header)));
  }
}
```

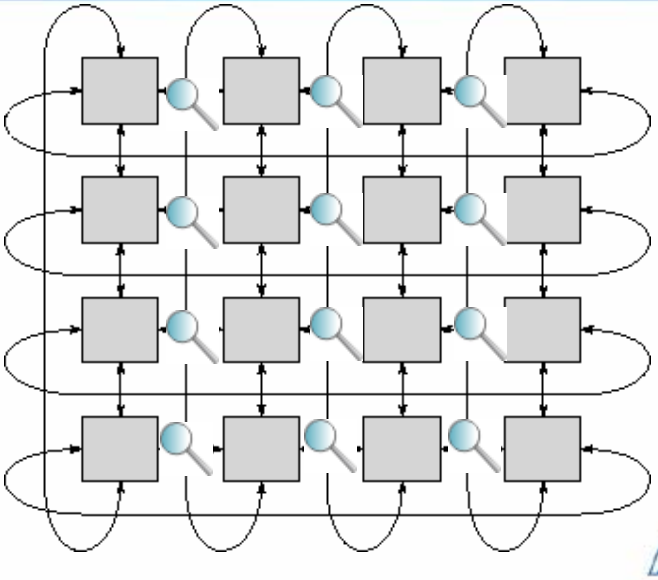
Platform With Intercept Library

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

```
static intercept_send_fn (_blkWrLeftXmit)
{
 vmiPrintf("-I:SDM: XMT BLK LEFT data packet
to node(%s)\n", getNodeAddrStr(x,y));}
}

static intercept_ack_fn(_blkWrLeftAck)
{
 vmiPrintf("-I:LSM(%s): ACK BLK LEFT data
packet\n",
}
```


Verify Transaction Packets between Sim and Virtual Model



Positive Feedback

- Use of intercepts eliminate need to change NIOS-II microcode
- 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

ALTERA. MEASURABLE ADVANTAGE™

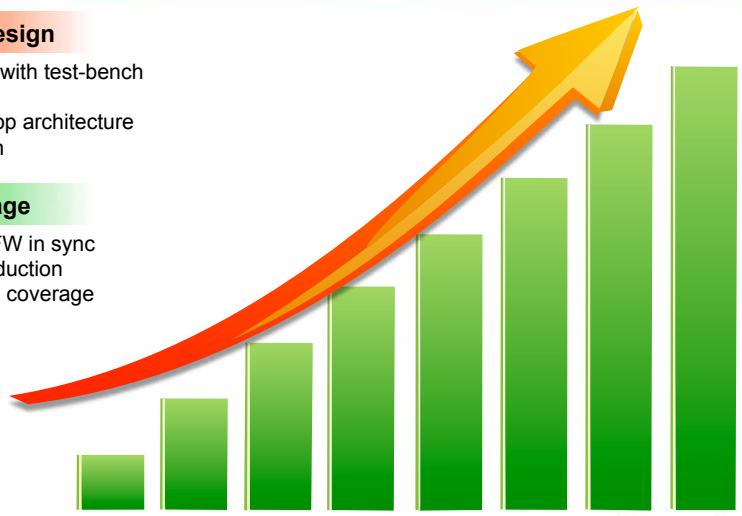
Improvements Realized

Speed of Design

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

Test Coverage

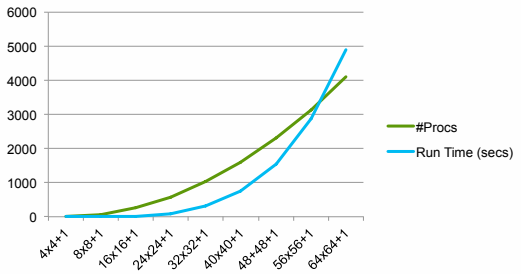
- RTL and FW in sync
- 100% production microcode coverage



ALTERA. MEASURABLE ADVANTAGE™

Still Work to be Done

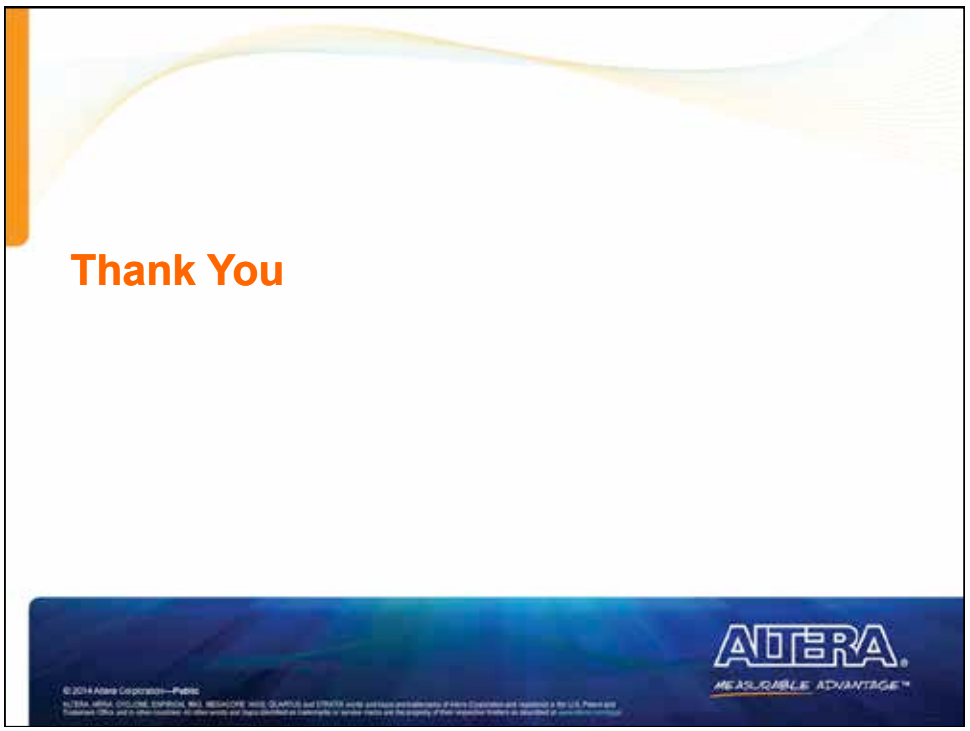
1. **Separate instances of the intercept library per processor**
   - Memory impact
   - Duplication and sync
2. **Execution speed vs number of cores**
3. **Enhance instrumentation to include FW profiling and performance**



Conclusions

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
ALTERA
MEASURABLE ADVANTAGE™