Hardware Observability Framework for Non-Intrusive Monitoring of Complex Embedded Systems

Roman Lysecky
Department of Electrical and Computer Engineering
University of Arizona
rlysecky@ece.arizona.edu

Students: Jong Chul Lee, Sachi Mahadevan

Embedded Systems Design Laboratory
http://www.ece.arizona.edu/~embedded
Software Observability

- dynamic instrumentation of software systems
  - DTrace [Solaris]
  - SystemTap [Linux]
- no overhead when not enabled
- software probes can monitor more than just instruction addresses
- secure execution of designer specified instrumentation code

*How we provide similar capabilities at the hardware and system levels?*
Introduction - Goals of System Observability

- **Goals of System Observability**
  1. monitor execution behavior of hardware circuit
     - arbitrary events, registers, or computational components
     - as non-intrusive as possible
  2. provide support for various types of hardware monitors – referred to as event probes
     - e.g., event probes, event and data probes
  3. allow arbitrary designer-defined software probe monitor in response to hardware probes
  4. provide integrated monitoring of software and hardware components
     - integration of *software event probes* within single environment
  5. NON-INTRUSIVE
     - should not impact/perturb the performance of the system
Traditional/Existing Hardware Monitoring - JTAG

- JTAG Scan Chain Hardware Analysis
  - Scan chains typically built-in into ICs
  - Could be utilized for dynamic analysis, but incurs significant overhead
  - Pros
    - readily available without additional hardware
  - Cons
    - interrupts HW execution to scan all/selected registers
    - must be periodically performed to detect events
  - [Leatherman and Stollen, IEEE Potentials 2005]

HW execution is interrupted

JTAG Scanning of All Registers

HW Execution

SW Analysis of Scan

JTAG Scanning ...

HW execution is interrupted

Must periodically scan to detect changes

JTAG scan requires 10’s of milliseconds
Traditional/Existing Hardware Monitoring - Export to IOs

- IO Access to Internal Registers
  - internal registers/signals of interest can be connected to pins at boundaries of HW circuit
  - pros
    - does not interrupt HW execution
    - readily available without additional hardware
  - cons
    - wires – additional wires needed to connect internal registers
      - can impact area, delay, power, and cost of circuit
    - SW must continuously monitor registers to detect events

- [Abramovic et al., DAC 2006 ]
- [Vermeulen and Goel, IEEE Design&Test 2002]
- [Shutlz et al., CF 2007]
**Dynamic Assertion Checkers**

- Automatically synthesize assertion checkers from specification used for validation/verification
- Property Specification Language (PSL) commonly used for specifying assertions

**Pros**
- Automated tools for creating hardware checkers
- Can be used for dynamic assertion testing
- Hardware for assertion closely integrated with logic

**Cons**
- Only indicates occurrence of assertion, no support for events or data capture
- Requires additional support for online monitoring

- PSL [Accellera, 2004]
- FoCs [Abardanel et al., CAV 2000]
- [Borrionne et al., FDL 2005]
- [Morin-Allory et al., 2008]
- [Boulé and Zilic, ACM TODAES 2008]
Hardware Observability

- **Hardware/System Observability**
  - hardware probes inserted directly into hardware to detect probe events
  - observability engine performs all interfacing with hardware probes and triggers designer-defined SW code if enabled
  - pros
    - does not interrupt HW execution
    - *event-driven* HW monitoring – i.e. no unnecessary polling or periodic scanning
    - minimal impact on hardware circuit
  - cons
    - area requirements of observability engine

![Diagram of hardware observability](image-url)

- Hardware analysis is triggered by designer-defined probes
- Analysis can be disabled for HW probes

**HW Probe/SW Analysis**

**HW Probe**
Hardware Observability

- **Hardware Observability Framework**
  - hardware observability interface (HWOI)
  - interface to individual hardware IP cores
  - monitors internal signals to detect hardware event probes
  - tracks hardware event probes, timing, and
Hardware Observability

- **Hardware Observability Framework**
  - hardware observability bus (HWOBus)
  - dedicated bus for hardware observability monitoring
  - allows for observations without affecting system bus
  - optional hardware
Hardware Observability

- Hardware Observability Framework
  - hardware observability engine (HWOEngine)
  - implements designer-specified software probe monitors
  - APIs defined for basic hardware probe configuration and event processing
Hardware Observability

- Hardware Event Probes (HEPs)
  - designer-defined hardware event probes (HEPs) are integrated within HW components
  - HEPs point defines an event or register that a designer would like to monitor
Hardware Observability

- Hardware Observability Interface (HWOI)
  - eventflags register
  - stores status of all hardware event probes
  - eventmask register
  - enables/disables firing of hardware event probes
- Hardware Observability Interface (HWOI)
  - interrupt is generated whenever at least one
Hardware Observability

- Hardware Observability Interface (HWOI)
  - state-based definition of HEP with blocking
Experimental Setup

- **Experimental System Configuration**
  - **goal**: evaluate hardware observability for monitoring system bus performance and overflow conditions
  - single core microprocessor system with several hardware coprocessors and common peripherals
    - FIR coprocessor: 13-tap fixed point finite impulse response filter
Experimental Setup

- Experimental System
  - hardware event probes
  - FIR computational overflow
  - bus transaction initiation
  - bus transaction acknowledge (granting access to bus to coprocessor)
  - bus transaction completion

- SW probe monitors
- report overflow condition for
Experimental Results - Case Study

- Effects of bus priority between hardware IP cores
- Effects of overlapping bus transactions, cache writebacks, and peripheral execution

**Graph:**
- X-axis: Burst Transaction Period (cycles)
- Y-axis: Bus Transaction Wait Time
- The graph displays variations in wait times for burst transactions at different transaction periods, highlighting specific points of overlap and priority effects.
Experimental Results - Area (LUTs+FFs)

Hardware Event Probes

- HWO Engine
- HWO Bridge
- HEPs

Hardware Event Probes
**Experimental Results - Comparison with PSL Assertion Checkers**

- **PSL Specification for Asserts (nearest equivalent to HEPs)**
  
  \[
  \text{assert (cap_input > trig_value) @rising_edge(Clk); assert (p0_rd_req -> next_a [0 to 10] (p0_rd_rdy)) @rising_edge(Clk); \ldots assert (p0_rd_req -> next_a [2 to 3] (p0_burst_done)) @rising_edge(Clk);}
  \]

![Diagram showing comparison between Bridge/Interface and HEPs/Assertions](image)
Current Work - System Observability

- Integration of hardware & software observability
  - use real-time trace/debug ports provided by many processors
  - developing HWOI for software applications
Future Work - Are we there yet...No!

- developing extensions to existing formal specification for defining event probes - build upon existing methods
- property specification languages (PSL)
Thanks.