CoWare®
The ESL Design Leader

Debug and Test Techniques for Parallel Software using Virtual Platforms

MP-SoC 2008

Achim Nohl, CoWare Inc.

Agenda

- Introduction
- Defects in Concurrent Software
- Virtual Platform Based Debugging
- Triggering and Asserting Defects
- Reproducing and Tracing Defects
- Summary

Introduction

Languages and methodologies for parallel programming: *"Such languages and methodologies may eventually be forthcoming. But for now,* ***a great deal of legacy sequential software is being transformed into parallel code, however laborious that process may be****."* - electronicdesign.com 04/2008

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it." - Brian W. Kernighan

Consequence**: Debugging nightmare**

CoWare® The ESL Design Leader

Concurrent Software Defects

- **Defects in sequential programs are largely deterministic**
- **Concurrent programs have more defect modes than sequential ones**
  - Asynchronous interaction between multiple programs
- **Many defects unique to concurrent programs are rare probabilistic events**
  - Some defects require unlucky timing
  - Harder to trigger the defect
- **Changes in one program may cause bugs to emerge in another.**
  - Source of a problem is likely not located in the program or core where the problem is detected
  - Harder to trace the symptom to the cause of the defect

CoWare® The ESL Design Leader

Debugging using real Hardware

Debugging complex, time sensitive, concurrent program defects with technology that is known for:

- **Non-deterministic behavior ?**
  - Minimal timing changes of events have huge impact on overall system behavior
  - Defects are hard to reproduce

- **Limited controllability ?**
  - Debugging is intrusive -> Heisenbug
  - Time cannot be stopped globally
  - State provided by debuggers may be not coherent

- **Limited visibility ?**
  - Limited (in-consistent) exposure of platform registers and pins
- **Consequences !**
  - Not sufficient for multi-core software development

CoWare® The ESL Design Leader

Virtual Platform Debugging

A virtual platform is a (partial) model of the hardware SoC that can run real embedded software with simulation performance close-to-realtime.

- **Non intrusive global visibility and control**
  - *A fundamental characteristic of a virtual platform*
- **Global synchronous system stop**
  - No state change in any core or peripheral during global stop
  - Ensures consistent system state
  - Inspect system level program execution, registers, memories and signals (e.g. interrupt lines)
- **Synchronous stop is a must to debug concurrent software**
  - But more is required!
- **Challenge: Debug and test the flow of concurrent software**
  - Each step every core advances the program counter and changes the state of the platform
  - Can you keep the overview?
  - More debug automation required!

CoWare® The ESL Design Leader

Virtual Platform Debugging

Virtual Platform based debugging solutions enable a methodical process to debug parallel software defects

Debugging Process:

1. **Trigger a defect**
2. **Catch the defect**
3. **Reproduce the defect**
4. **Trace back symptom to defect cause**

Parallel software defects:

- **Synchronization**
  - Deadlocks
- **Shared memory communication**
  - Race conditions
  - Data corruption
- **Processor utilization!**
  - Starvation

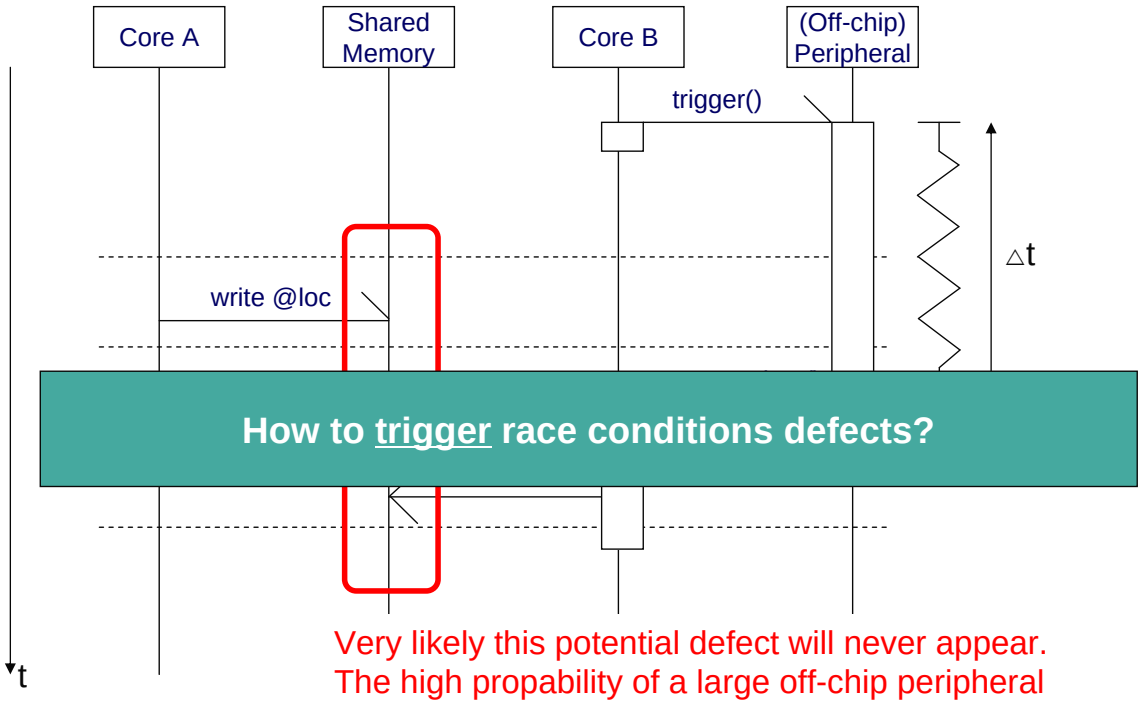
CoWare The ESL Design Leader

Triggering the Defect

- **Defect appears always**
  - Best case
- **Defect appears sometimes**
  - Not really the worst case
- **Defect appears never**
  - At least not during development and test
  - Only after product has been rolled out
  - Worst case
- **Concurrent software defects**
  - Probabilistic events can mask potential errors

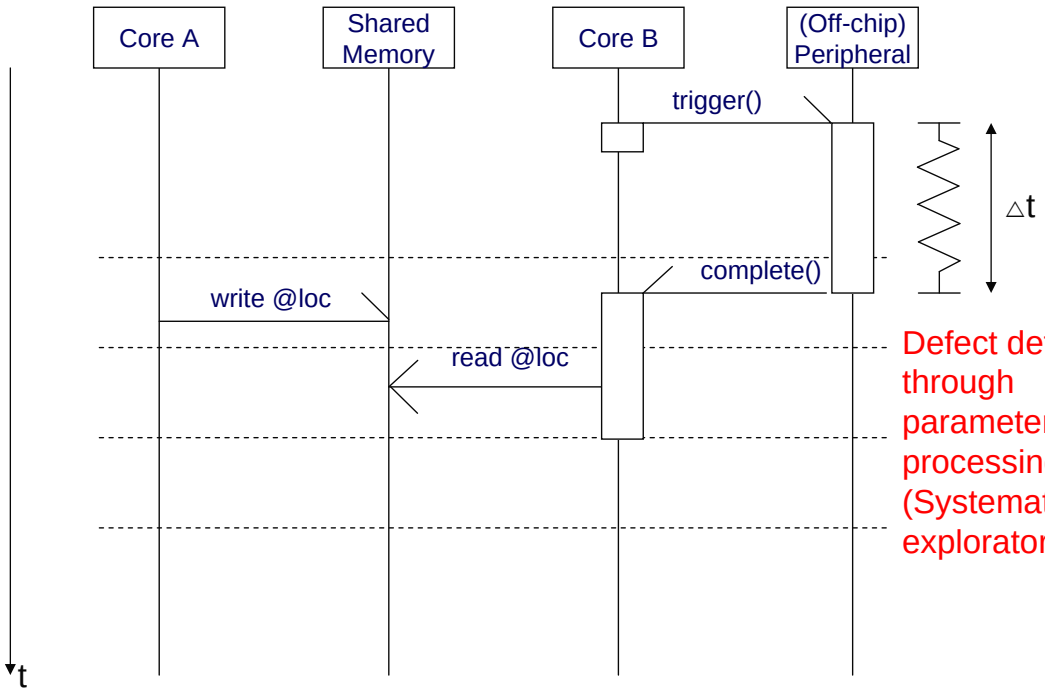
NEWS
A BMW trapped a Thai politician when the computer crashed. The door locks, windows, A/C and more were inoperable.
Responders smashed the windshield to get him out.

Example: Triggering Race Conditions



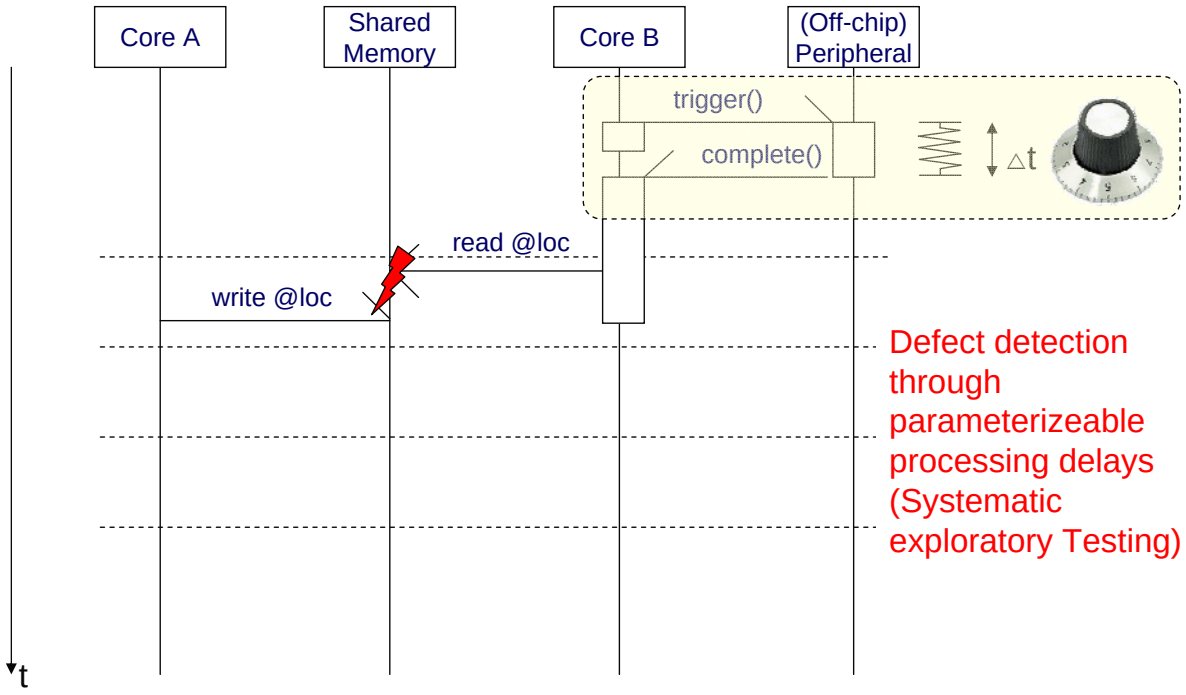
Very likely this potential defect will never appear.
The high propability of a large off-chip peripheral processing delay hides the defect.
A missing synchronization between read and write.

Example: Triggering Race Conditions



Defect detection through parameterizeable processing delays. (Systematic exploratory Testing)

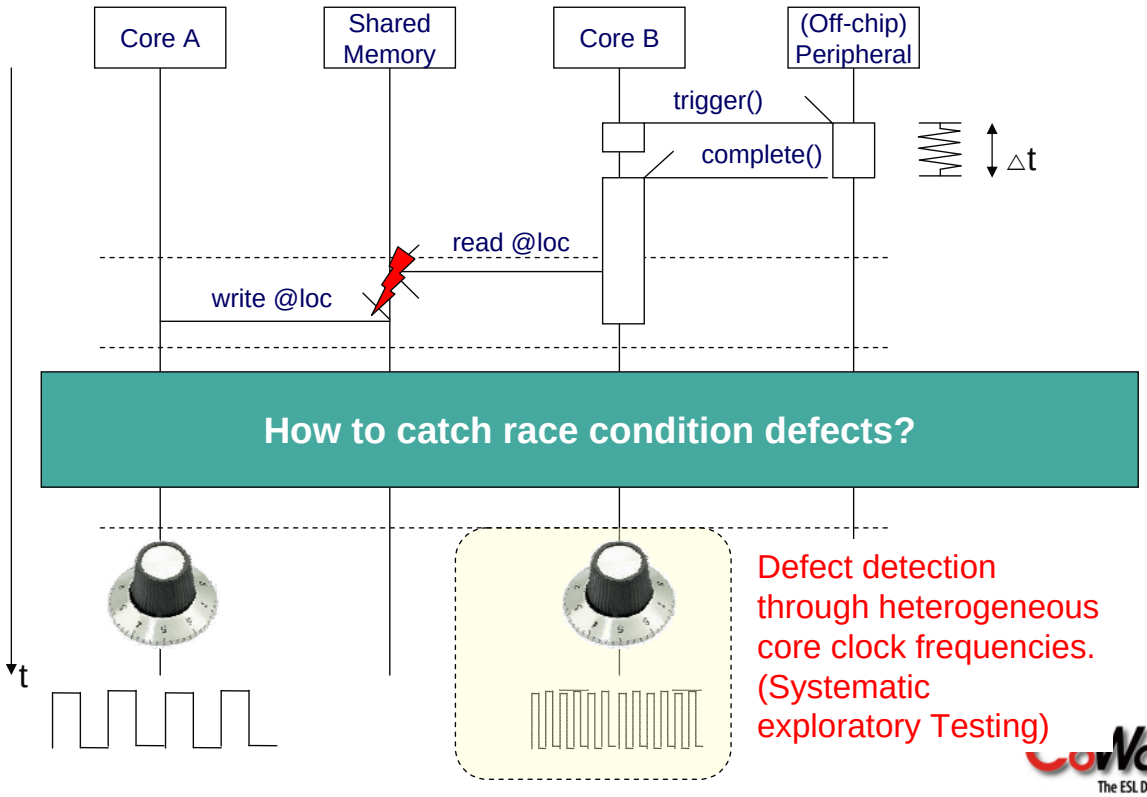
Example: Triggering Race Conditions



Defect detection through parameterizeable processing delays (Systematic exploratory Testing)

CoWare The ESL Design Leader

Example: Triggering Race Conditions

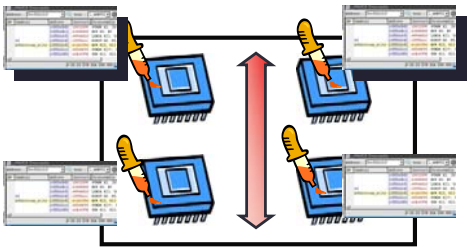


Defect detection through heterogeneous core clock frequencies. (Systematic exploratory Testing)

CoWare The ESL Design Leader

Asserting Defects

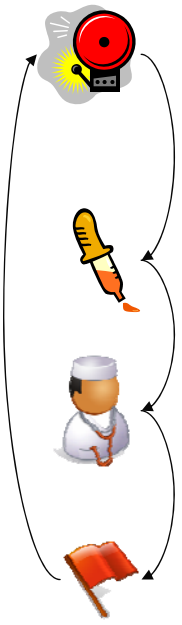
- **Software assertions are the best practice method to catch fault conditions**
  - Classical software assertions cannot assert the global platform state (apart from shared memory)
  - No assertions with inter-core state dependencies
- **System level software assertions required**
  - CoWare's Virtual Platform scripting framework allow for non-intrusive system level assertions



CoWare
The ESL Design Leader

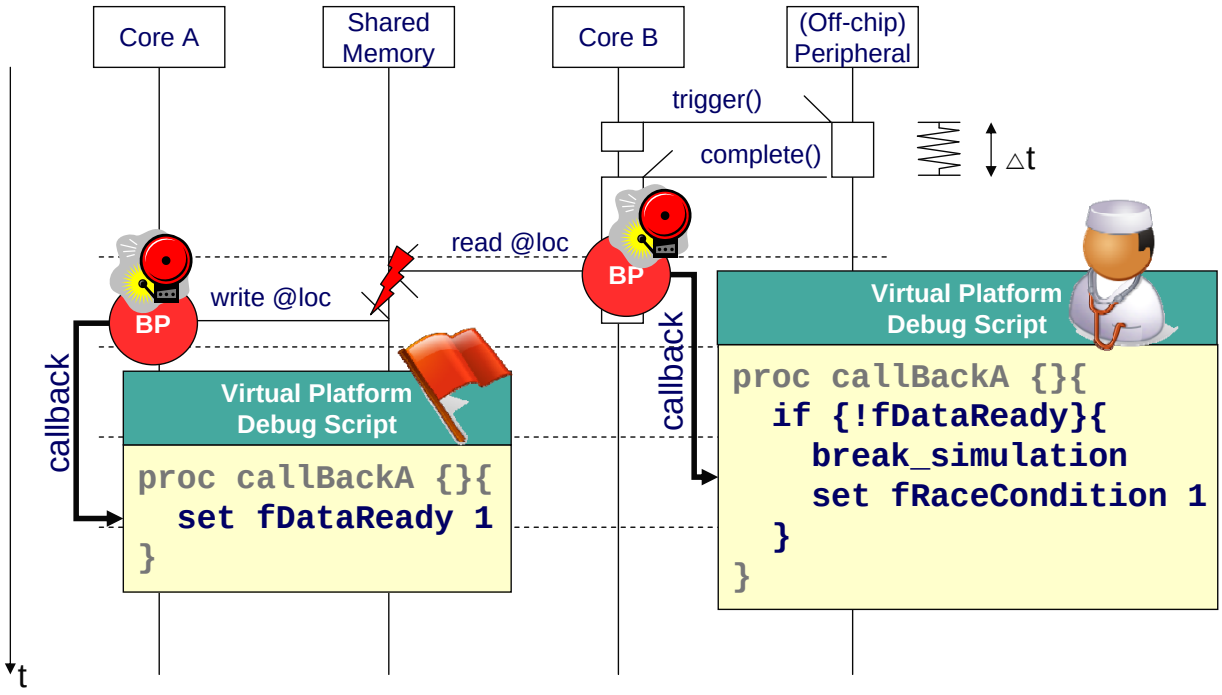
System Level Software Assertions

- **Notify and react on events**
  - Register, memory, pin access and change
  - Program control (e.g. Function call)
- **Inspect state**
  - Register, memory and pin values
- **Validate**
  - Assert correctness
- **Report**
  - Feedback assertion result
  - Stop or carry state to next assertion



CoWare
The ESL Design Leader

Asserting Race Conditions



CoWare The ESL Design Leader

Virtual Platform Debugging

Virtual Platform based debugging solutions enable a methodical process to debug parallel software defects

Debugging Process:

- ✓ Trigger a defect
- ✓ Catch the defect
- 3. Reproduce the defect
- 4. Trace back symptom to defect cause

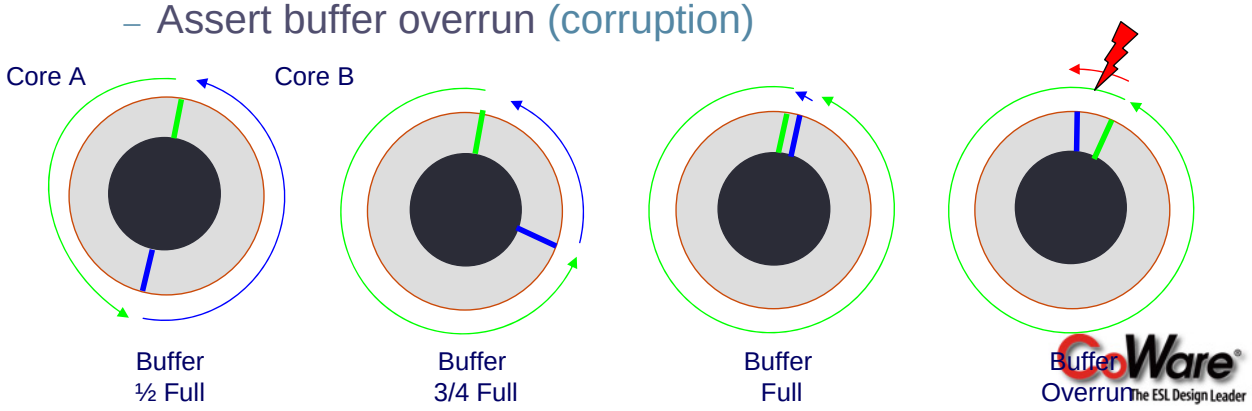
Parallel software defects:

- **Synchronization**
  - Deadlocks
- **Shared memory communication**
  - ✓ Race conditions
  - Data corruption
- **Processor utilization!**
  - Starvation

CoWare The ESL Design Leader

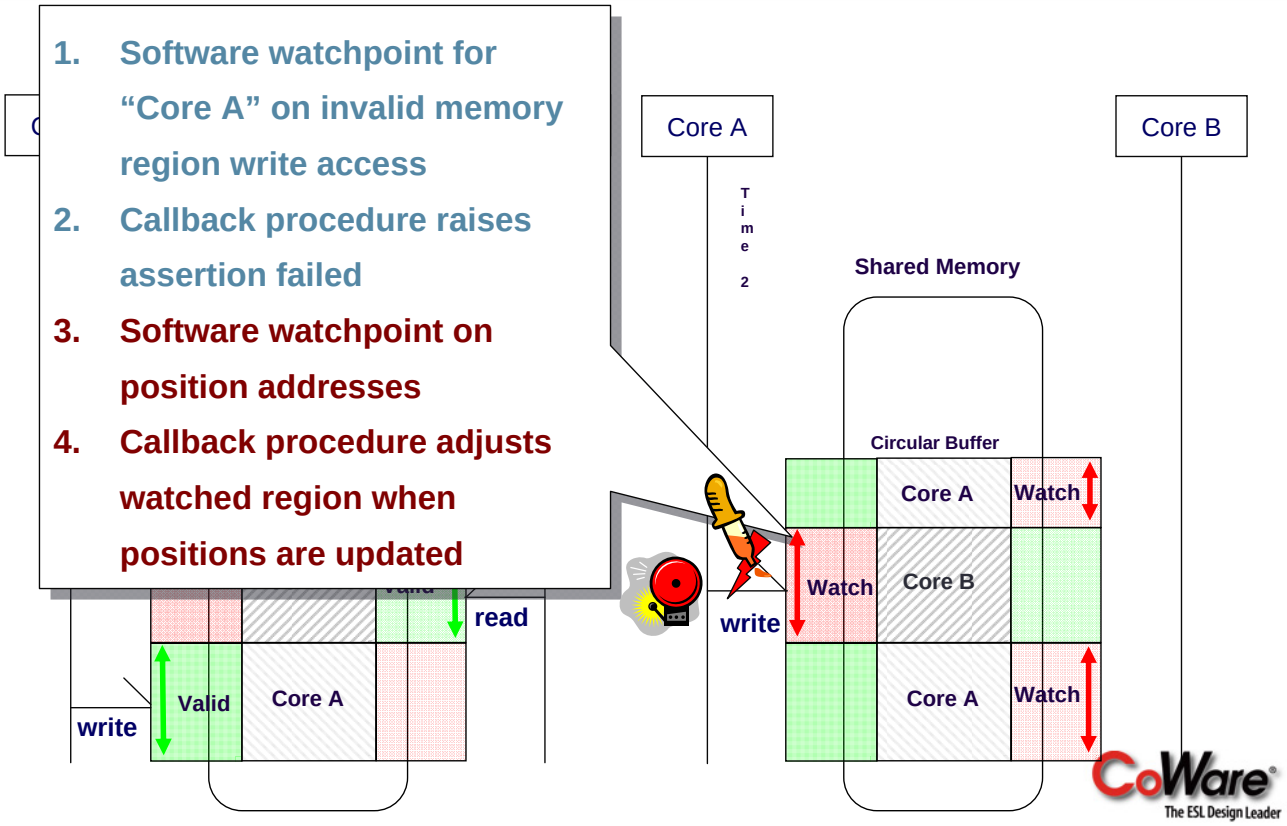
Shared Memory Analysis Example

- **Core A streams video data to core B:**
  - Core A runs and OS with a stream device driver.
  - Device driver puts data into a circular buffer
  - Core B decoder firmware reads data from buffer.
- **Debugging/Analysis goal:**
  - Identify periods while buffer is empty (starvation)
  - Assert buffer overrun (corruption)



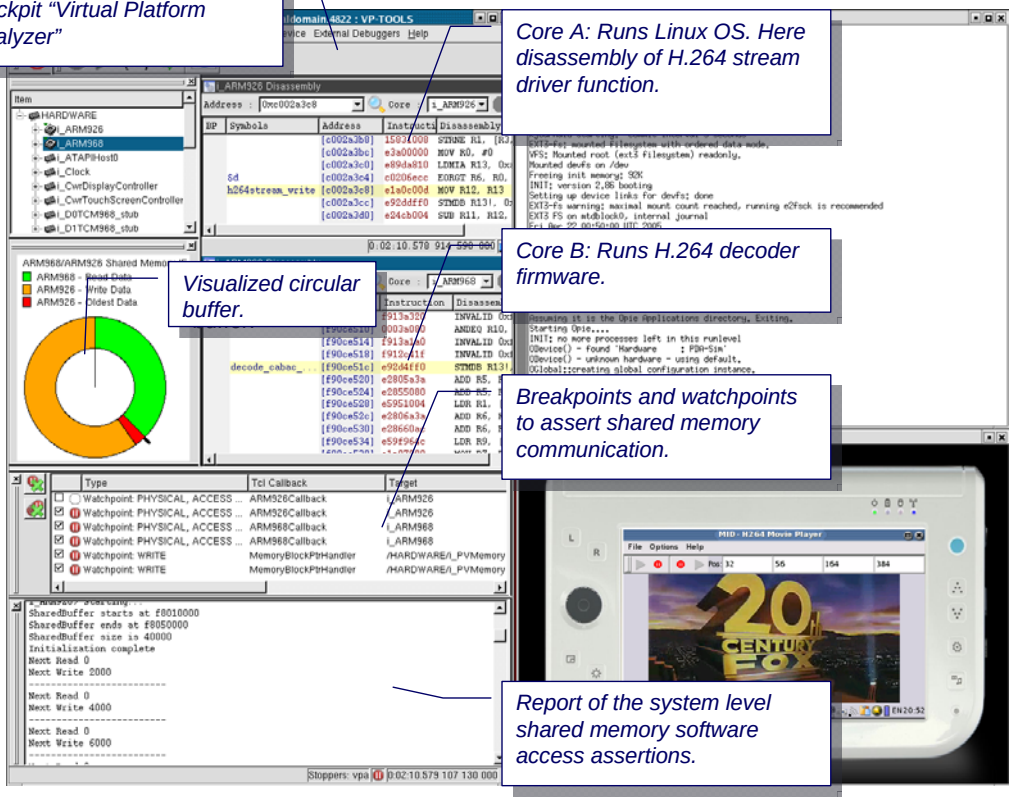
Shared Memory Analysis Example

1. Software watchpoint for "Core A" on invalid memory region write access
2. Callback procedure raises assertion failed
3. Software watchpoint on position addresses
4. Callback procedure adjusts watched region when positions are updated



Shared Memory Analysis Example

Virtual Platform Control Cockpit "Virtual Platform Analyzer"



Core A: Runs Linux OS. Here disassembly of H.264 stream driver function.

Core B: Runs H.264 decoder firmware.

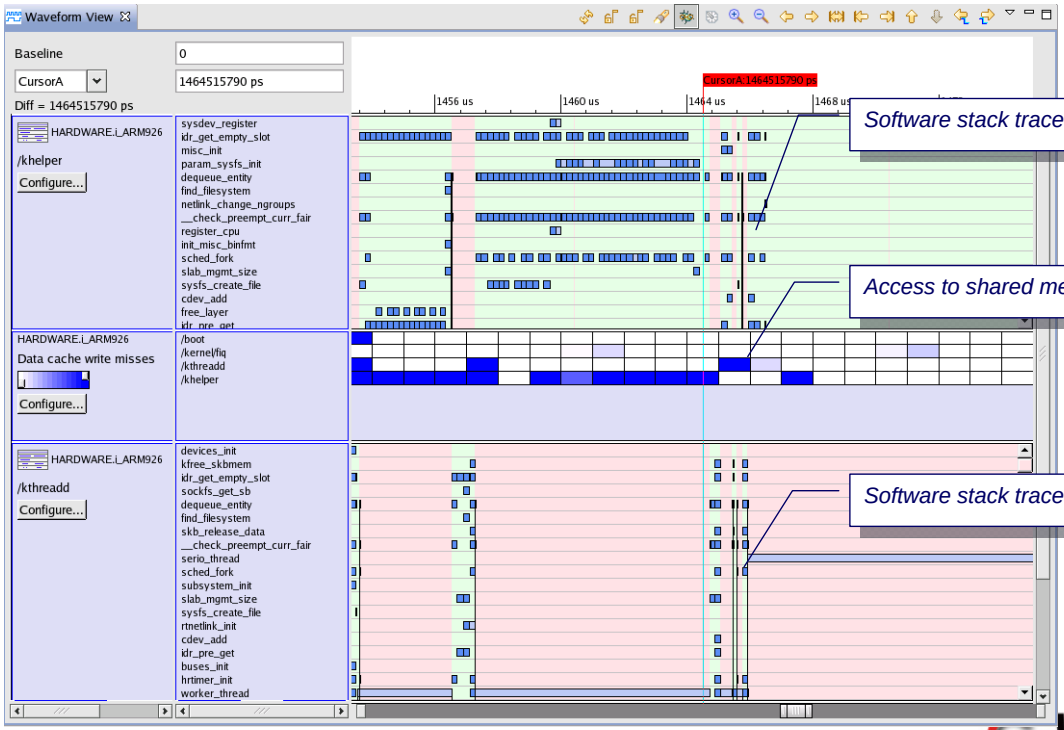
Visualized circular buffer.

Breakpoints and watchpoints to assert shared memory communication.

Report of the system level shared memory software access assertions.

CoWare
The ESL Design Leader

Virtual Platform Software Analysis



Software stack trace of core A

Access to shared memory

Software stack trace of core B

CoWare
The ESL Design Leader

Virtual Platform Debugging

Virtual Platform based debugging solutions enable a methodical process to debug parallel software defects

Debugging Process:

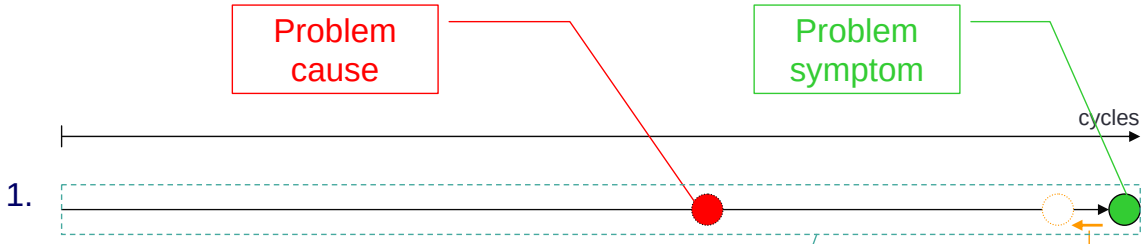
- ✓ Trigger a defect
- ✓ Catch the defect
- **3. Reproduce the defect**
- **4. Trace back symptom to defect cause**

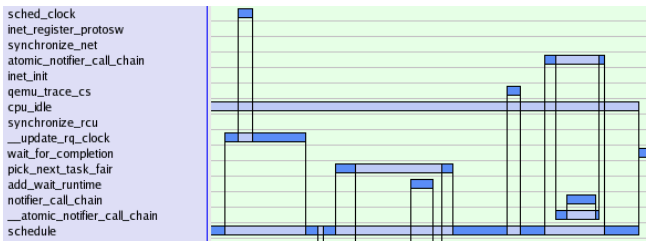
Parallel software defects:

- **Synchronization**
  - Deadlocks
- **Shared memory communication**
  - ✓ Race conditions
  - ✓ Data corruption
- **Processor utilization!**
  - ✓ Data corruption

CoWare The ESL Design Leader

Tracing the Defect



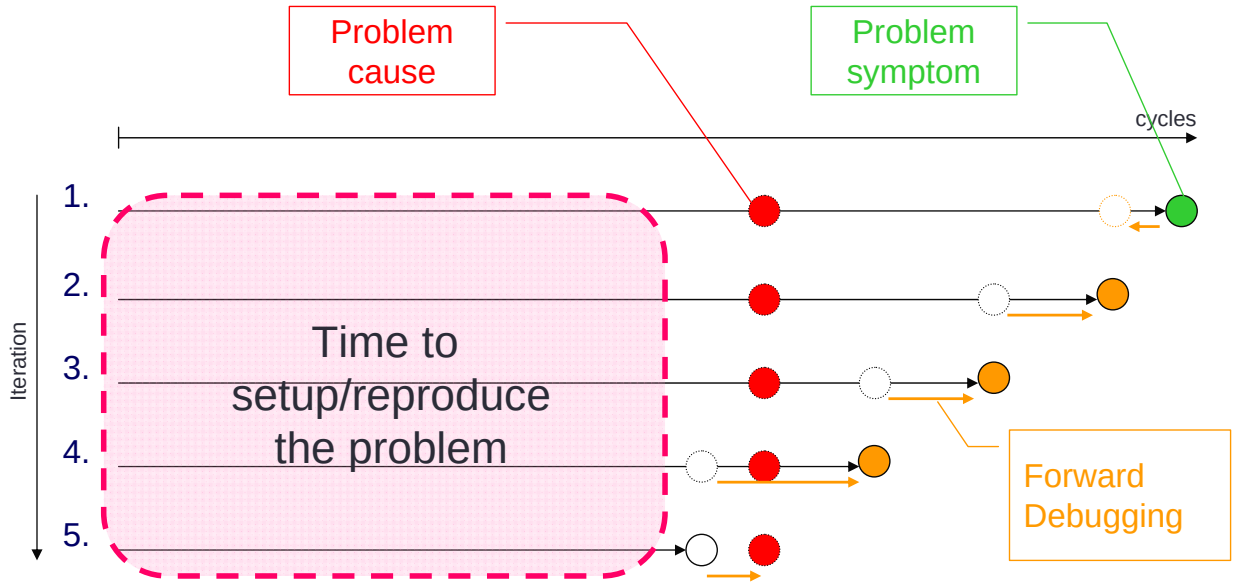


Stack trace analysis over time gives clarity about entire history of function and instruction executions. Helps to exclude many potential error sources to efficiently narrow down the cause of the problem.

Program state can be traced back non-ambiguously and manually only for a limited amount of cycles.

CoWare The ESL Design Leader

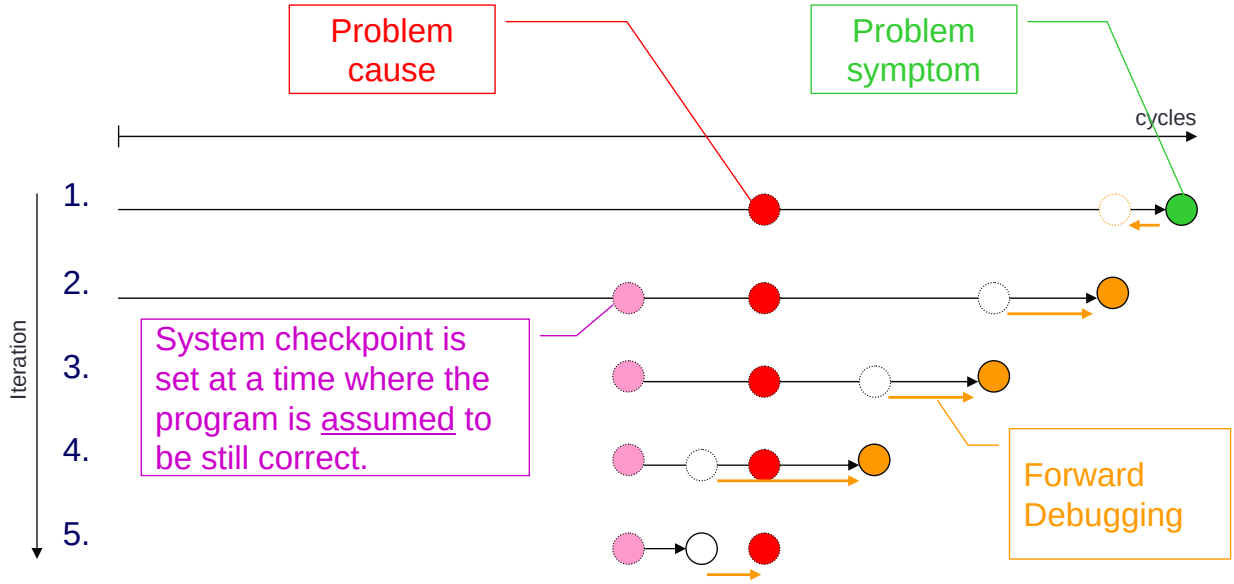
Tracing the Defect



- Debugging is an iterative and manual task
- Huge time is spent re-producing the problem

CoWare The ESL Design Leader

Tracing the Defect



- Debugging is an iterative and manual task
- Huge time is spent re-producing the problem
- Checkpoint/restore as a big productivity factor

Summary

- **Using Virtual Platforms to**
  - Trigger, assert, trace software defects defect such as
  - Deadlocks, race conditions, data corruption, starvation
- **Virtual Platforms**
  - Will become the main means to debug defects during embedded software development for MP-SoCs.
  - Provide unique non-intrusive and deterministic
    - Observability &
    - Controllability
  - Allow for new debugging techniques, that
  - increase productivity and reduce risk
  - for software development