Boosting Debug Productivity -
Practical Applications of Verdi Debug Innovations
Embedded Software Debug Tutorial

Alex Wakefield

March 2017

CONFIDENTIAL INFORMATION

The following material is confidential information of Synopsys and is being disclosed to you pursuant to a non-disclosure agreement between you or your employer and Synopsys. The material being disclosed may only be used as permitted under such non-disclosure agreement.

IMPORTANT NOTICE

In the event information in this presentation reflects Synopsys' future plans, such plans are as of the date of this presentation and are subject to change. Synopsys is not obligated to develop the software with the features and functionality discussed in these materials. In any event, Synopsys' products may be offered and purchased only pursuant to an authorized quote and purchase order or a mutually agreed upon written contract.

SYNOPSYS

Agenda

Synchronized HW SW Debug

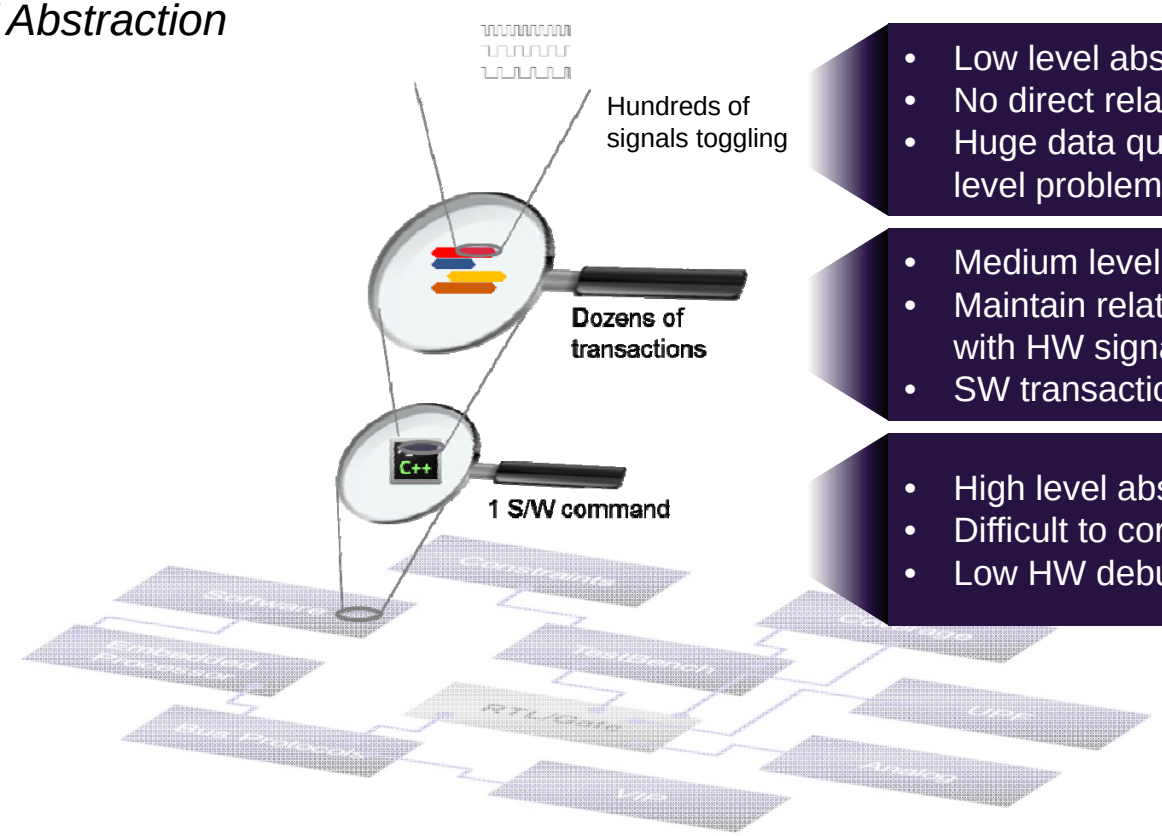
Practical Debug Application: Linux Boot

Performance Profiling of Embedded Software

Coverage Metrics for Embedded Software

SoC Debug Challenges

Levels of Abstraction



- Low level abstraction
- No direct relation to SW transaction
- Huge data quantity to debug system level problems

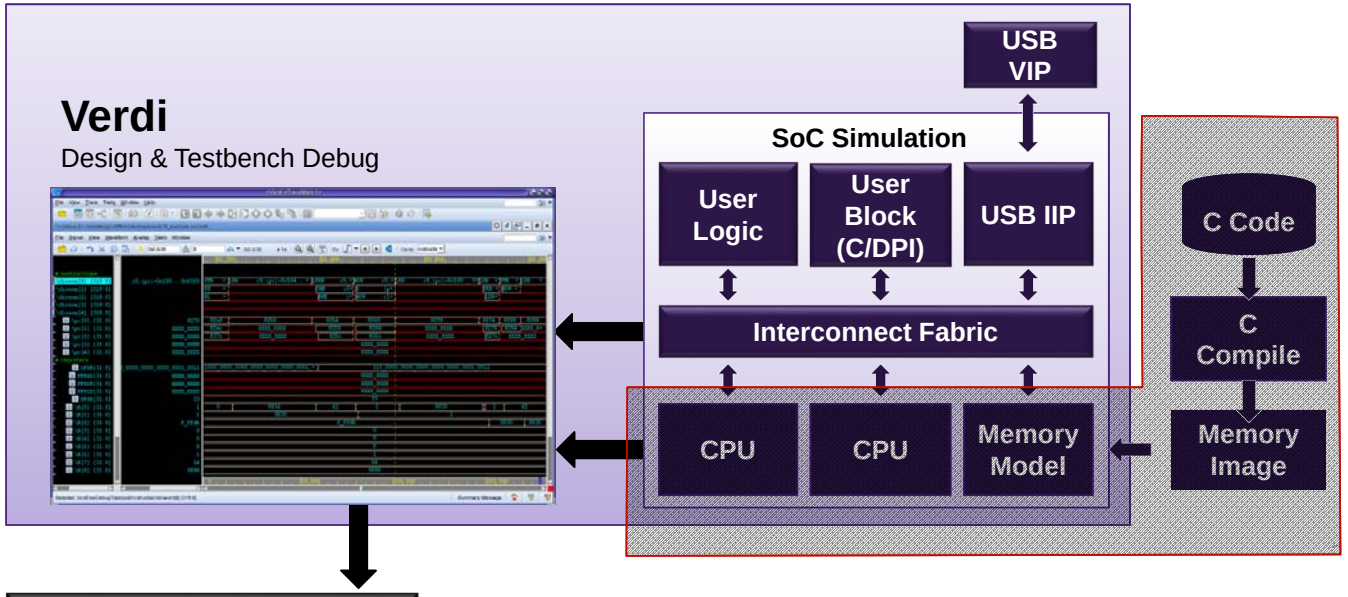
- Medium level abstraction
- Maintain relation to correlate with HW signals
- SW transaction concept available

- High level abstraction
- Difficult to correlate to HW
- Low HW debug visibility

Correlated debug across abstraction levels is required

SoC Debug Challenges

Lack of Visibility on the Software Side



CPU Software Debug

- Encrypted Model ?
- Registers + Nets ?
- No or Low-Visibility
- Looks like a Black Box

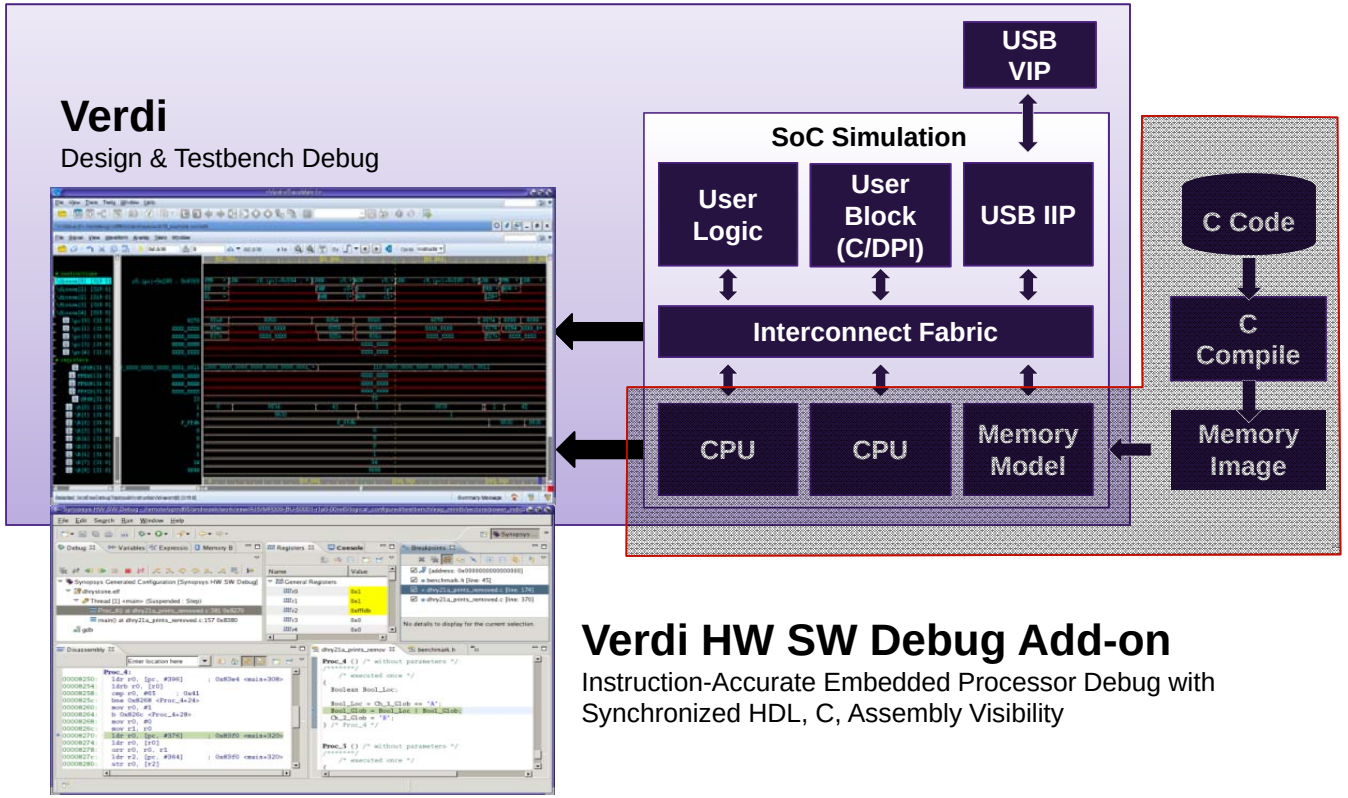
Time Consuming Manual Debug:

- Log-files, Waveforms
- ELF Dump, Disassembly
- Decoding Stack, Variables

At every point in time…

SoC Debug Challenges

Lack of Visibility on the Software Side

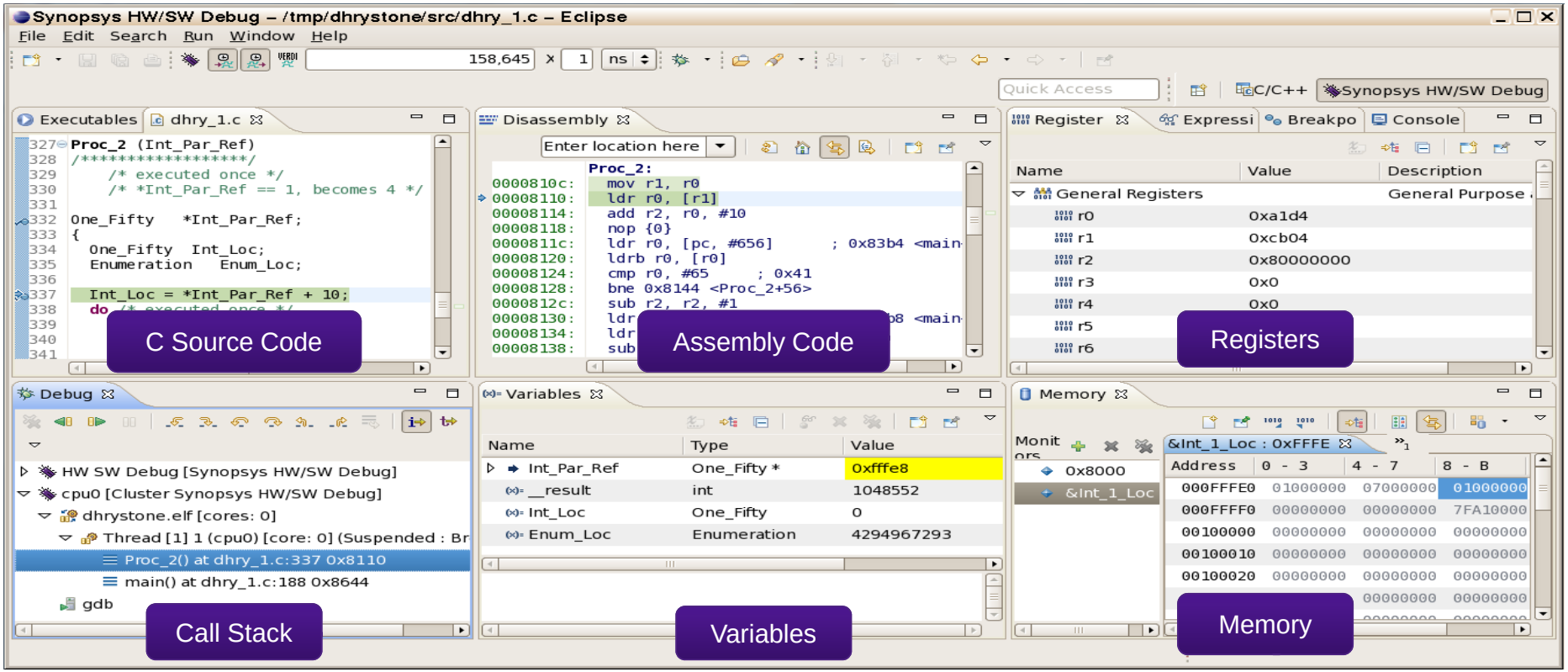


Verdi HW SW Debug Add-on

Instruction-Accurate Embedded Processor Debug with Synchronized HDL, C, Assembly Visibility

Embedded Software Debug in Eclipse

Synopsys Users Group
Silicon Valley

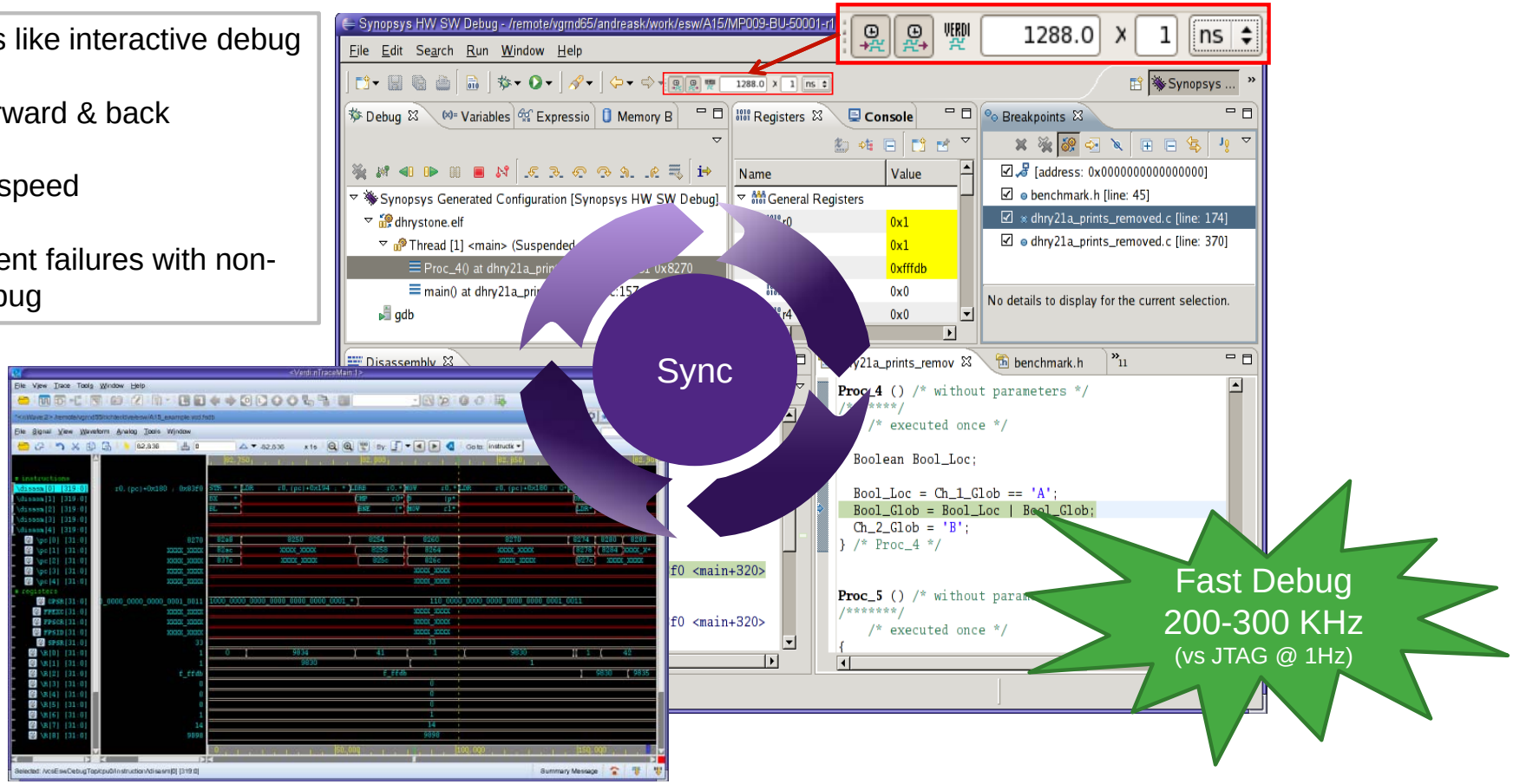


Provides programmer's view of code running on simulated processor

SYNOPSYS

Post-Process Debug Approach

- Eclipse feels like interactive debug
- Run/step forward & back
- Fast debug speed
- Solve transient failures with non-intrusive debug

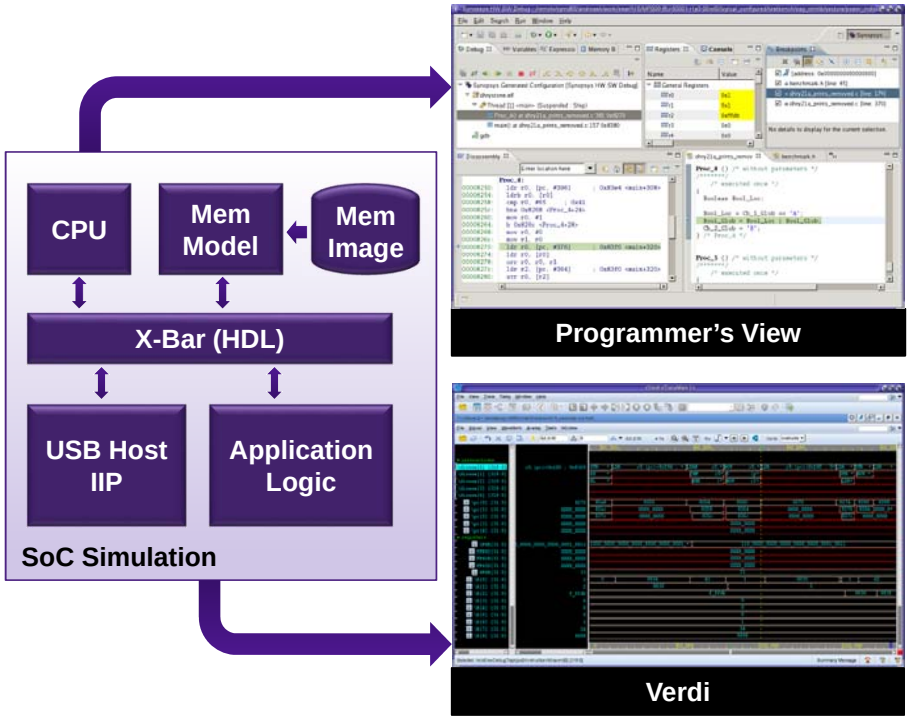


Verdi HW SW Debug

Embedded Processor Debug with Synchronized Hardware & Software

Synopsys Users Group
Silicon Valley

- Enables co-debug between HW & SW
- HW and SW time synchronization
- View C/Assembly source, C variables, stack, memory
- Debug multiple cores
- Simulation supports all ARM® Cortex® cores
- Easy to support additional/custom cores



SYNOPSYS®

Agenda

Synchronized HW SW Debug

Practical Debug Application: Linux Boot

Performance Profiling of Embedded Software

Coverage Metrics for Embedded Software

Debug Challenges of Software Tests

Synopsys Users Group
Silicon Valley

Dealing with different OS – Linux , Android, RTOS etc.

How can I debug the OS code itself?

How can I debug apps running on the OS?

Data sizes are huge! 600M-1B cycles for Linux boot!

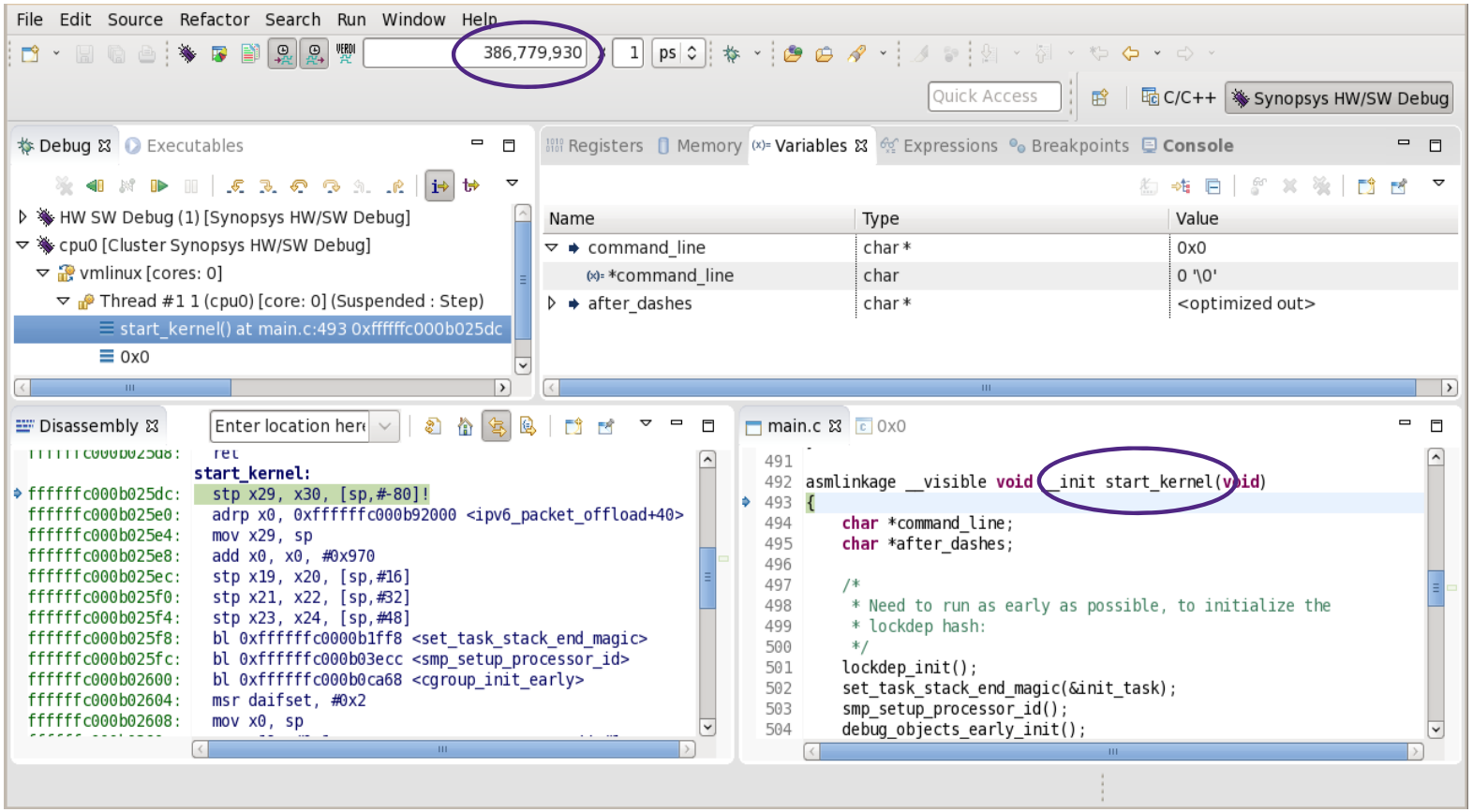
Debugging effort-intensive due to big source code base

Time-consuming test debug - multiple features tested

High speed embedded code debug with Verdi HW SW Debug

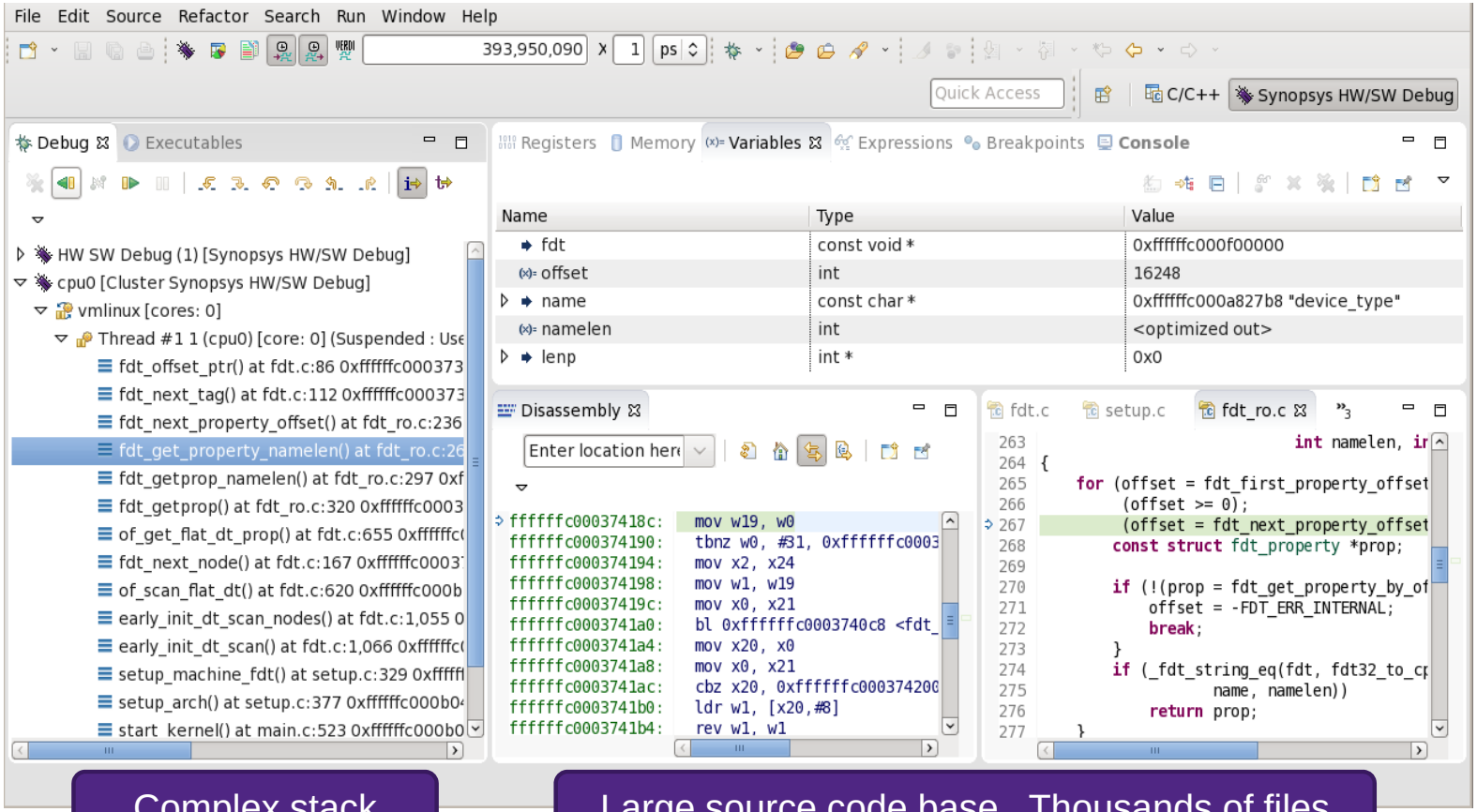
SYNOPSYS

Linux Boot Debug



- Long test split into time slices
- Time slice starts at 38M cycles
- Start debug from symbol name (e.g. app_start, start_kernel)
- Slice generated from “start_kernel” symbol
- Run forwards and backwards through this time slice

Debugging a Complex Software Stack



Complex stack

Large source code base , Thousands of files

- Large source code base
- Highly complex stack with many levels
- Variables include integer, string and complex types like linked lists
- Unix console visible in window with timestamps

Verdi HW SW Debug Demonstration

Agenda

Synchronized Debug with Verdi HW SW

Practical Debug Application: Linux Boot

Performance Profiling of Embedded Software

Coverage Metrics for Embedded Software

Measuring Embedded SW Performance

- **How fast does my embedded software run?**
  - How do I measure this before silicon arrives?
  - How do I measure without modifying the embedded executable?
- **What can I do if it's not fast enough?**
  - How can I get reports of time per function, callers, etc.?
  - Software team can do this with silicon…But how can I do this now?

Really need a **standard software profiler** for embedded software

Performance Profiling

Disassembly | gprof

gmon file: /tmp/multi_core/cluster1.gmon.out
program file: /tmp/multi_core/test_sw/build64/hwsw_debug_test64.elf
timestamp: 1/23/17 6:08 AM
4 bytes per bucket, each sample counts as 1ns

type filter text

Name (location)	Samples	Calls	Time/Call	% Time
Summary	1310760			100.0%
hwsw_debug_test.c	560160			42.74%
mallocr.c	163560			12.48%
utilities.c	105720			8.07%
memset.S	95280			7.27%
memchr.c	46920			3.58%
findfp.c	46320			3.53%
fvwrite.c	44640			3.41%
hwsw_debug_test.h	41760			3.19%
makebuf.c	26040			1.99%
puts.c	23280			1.78%
fflush.c	23040			1.76%

- Eclipse Open Platform Profiler Plugin
- Performance Profile of embedded software
- Task call totals
- Assembly level detail
- Non-intrusive: No insertion of code or opcodes
- Measure performance of production firmware

Agenda

Synchronized Debug with Verdi HW SW

Practical Debug Application: Linux Boot

Performance Profiling of Embedded Software

Coverage Metrics for Embedded Software

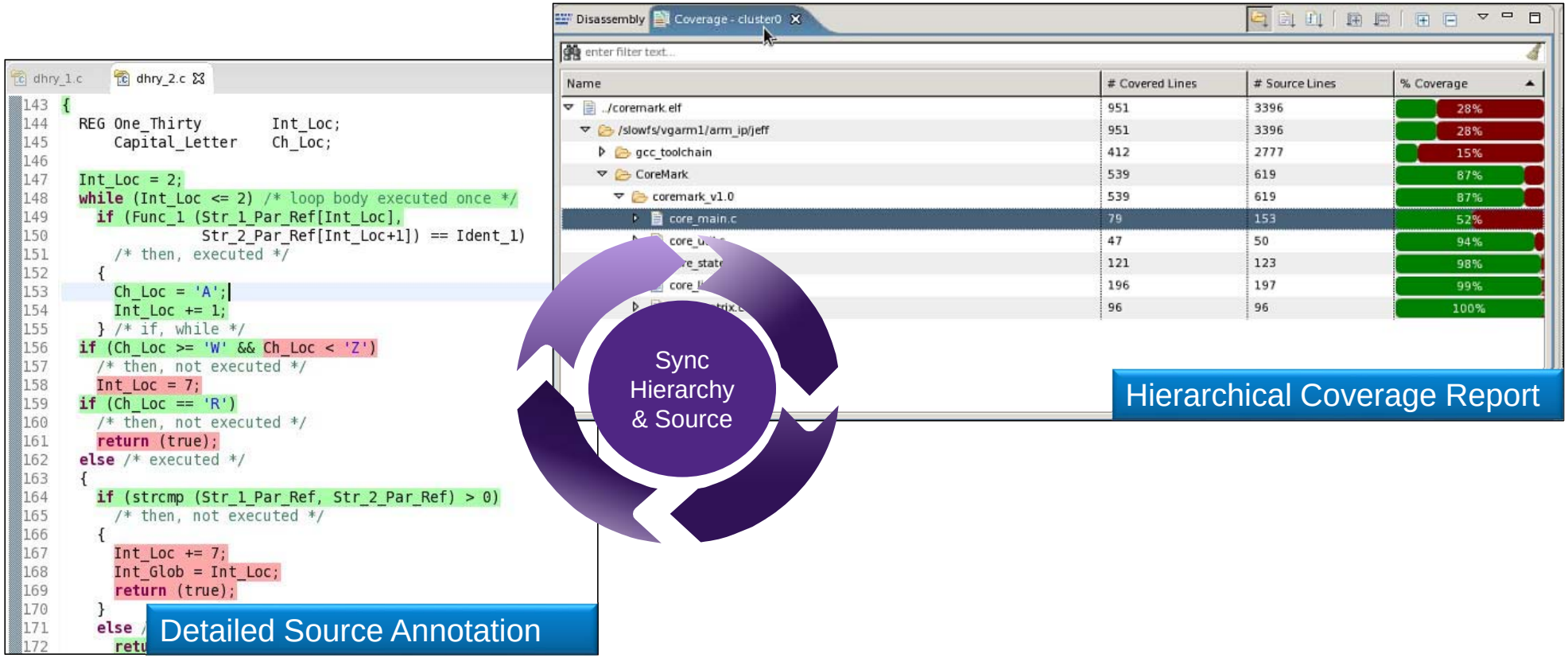
Embedded Software Code Coverage

- **How do I meet ISO 26262 Embedded Software requirements?**
  - Are any pieces of code not tested/executed?
  - Do I have dead code that I do not expect?
  - Sounds familiar for verification engineers…Think coverage!

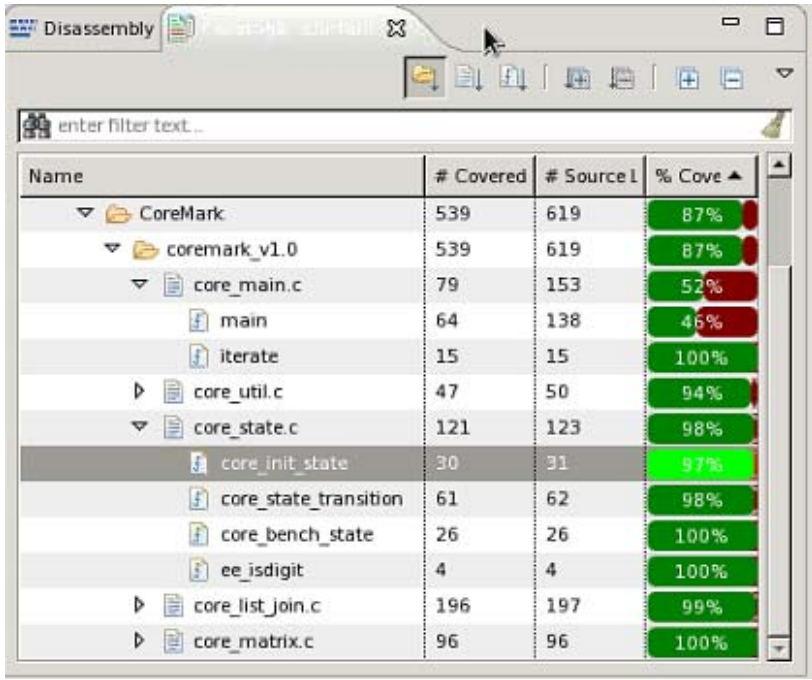
- **Line Coverage – Current method**
  - Software teams use coverage (gcov, purecov, etc)
  - Problem: gcov heavily modifies the compiled binary (Firmware, OS, App)
  - Teams want to run production image without changes

Need a **non-intrusive** way to measure line coverage

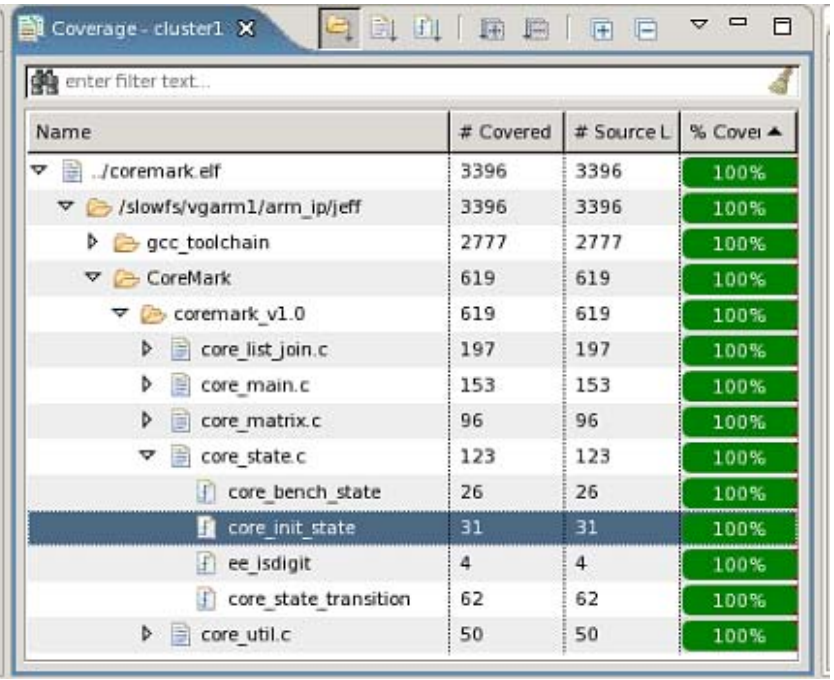
Embedded Software - Line Coverage



Embedded Code Coverage on Multiple Cores



Cluster #0



Cluster #1

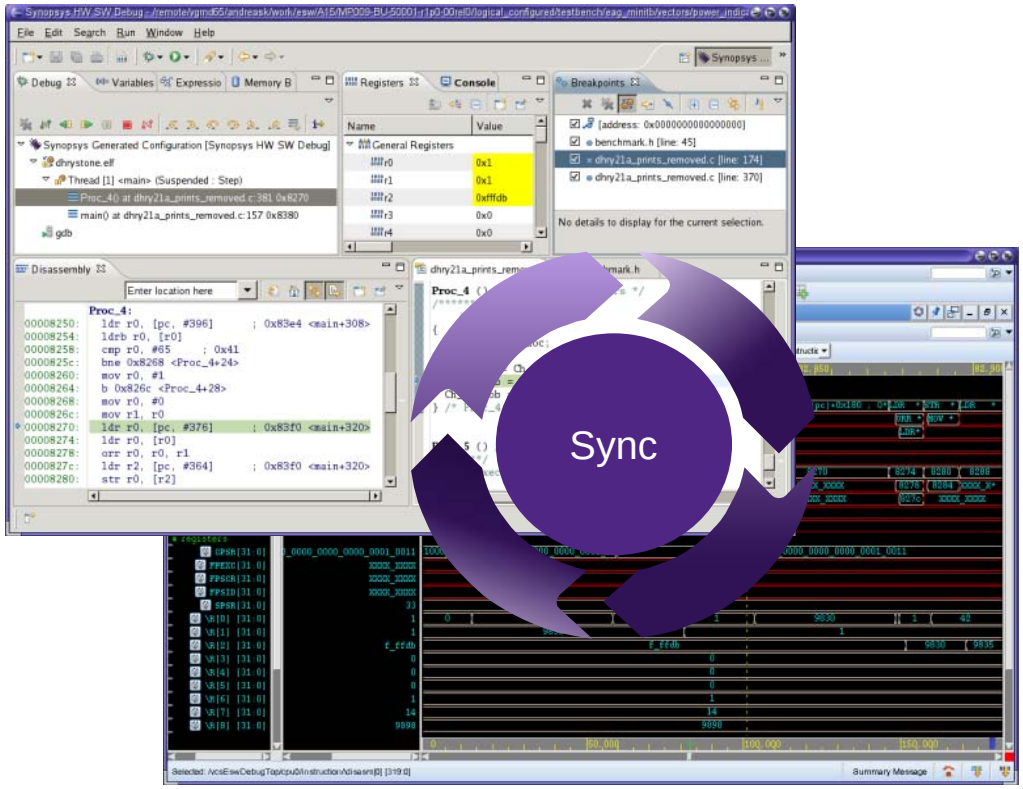
- Support for multiple clusters/cores
- Collect coverage per cluster/core
- Collect coverage per executable/elf

Verdi HW SW Debug Demonstration

Performance Profiling, Line Coverage

Verdi HW SW Debug

Boosting Debug Productivity



- **Verification Engineer Productivity**
  - Debug C-source code on embedded core
  - Debug software/hardware issues
  - Measure code coverage for ISO 26262
- **Software Engineer Productivity**
  - Full software development environment
  - Pre-silicon debug of boot and applications
  - Profile performance before silicon arrives

SYNOPSYS®
Silicon to Software™

Thank You

SYNOPSYS