

Virtual Platform Software Simulation for Enhanced Multi-core Software Verification

Simon Davidmann

Company:	Imperas Software Ltd, 17 March 2014
Event:	TVS – Software Testing
Location:	UWE Conference Centre, Bristol



Agenda



- Background the changing needs of embedded product development
- "Traditional" Embedded Software development solutions
- Working with Multi-core hardware and software
 - The next generation of Embedded Software development systems
- Example using assertions on software
 - shared memory monitor
- Summary

The changing hardware





- Accelerators
- Multi-core [is going to be in] [is in] everything
 - AMP, SMP, homogeneous, heterogeneous

Page 3

The changing software





Hardware Dependent Software (HDS)

Layer and Layers of software

The changing SW Verification Requirements



Applications Home Automation Flash Player MP3 Player Fitness Planner Application Framework Openmoko Android Hildon Libraries Hardware Dependent Pango Linux Kernel NAND Flas ATA/ID **Touch Controlle** RTC Driver **Display Drive** I2C Driver **lideo** Drive Keynad Drive



- Hardware Dependent Software (HDS)
 - Most complex foundation layer
 - Drivers, hypervisors, assembly libraries, Operating System
 - Buried problems often appear elsewhere in a system, leading to misdirected analysis
 - Ripe for corner case type issues
 - Post development bugs hardest to fix
 - Testing needs to be platform centric not application centric
- Modern SoC verification is complex
 - SMP/AMP Multicore Interaction
 - Shared memory & devices
 - Extensive accelerators, peripherals
 - Complex SW/HW interaction (e.g security)
 - Externally authored, complex libraries

Software (HDS)

Current Solutions For Early Embedded SW Development

Traditional Breadboard

- Limited system availability
- Limited external test access
- Limited internal visibility
- Late in arrival





MULTICORE DESIGN SIMPLIFIED

Imperas

Emulation & Cycle Accurate Models

- Provides reasonable verification capability but 1000x too slow for effective HDS verification
- Hard to get started



Page 6

Agenda



- Background the changing needs of embedded product development
- "Traditional" Embedded Software development solutions
- Working with Multi-core hardware and software
 - The next generation of Embedded Software development systems
- Example using assertions on software
 - shared memory monitor
- Summary

Virtual Platforms: Transforming Engineering Efficiency

SW vs HW development platforms

- Often available earlier for engineering
- Provide more powerful tooling
- Easier to replicate for extended testing
- Real time or faster performance

"Best-in-class manufacturers that make extensive <u>use of simulation</u> early in the design process hit revenue, cost, launch date, and quality targets for 86% or more of their products. Best-in-class manufacturers of the most complex products get to market <u>158</u> <u>days earlier with \$1.9 million lower costs</u> than all other manufacturers." <u>Simulation-Driven Design, Aberdeen</u> Group Study



MULTICORE DESIGN SIMPLIFIED

as

And Virtual Platform simulators can be very fast



•	Altera Nios II		ARM32			Imagination MIPS32			
Benchmark	Simulated Instructions	Run time	Simulated MIPS	Simulated Instructions	Run time	Simulated MIPS	Simulated Instructions	Run time	Simulated MIPS
linpack	3,075,857,171	2.52s	1225	6,105,766,856	4.79s	1277	9,814,621,392	5.31s	1852
Dhrystone	1,810,082,387	1.18s	1547	2,250,079,359	2.32s	974	1,795,088,667	1.27s	1414
Whetstone	5,850,887,389	3.28s	1789	1,185,959,501	1.04s	1140	1,890,420,892	0.93s	2033
peakSpeed2	22,000,013,458	3.11s	7097	22,400,008,766	4.67s	4807	22,800,009,853	4s	5714
	Xilinx	Micro	Blaze	ARM	ARM AArch64		Imagination MIPS64		IIPS64
Benchmark	Simulated Instructions	Run time	Simulated MIPS	Simulated Instructions	Run time	Simulated MIPS	Simulated Instructions	Run time	Simulated MIPS
linpack	6,386,275,159	3.77s	1699	594,945,589	1.01s	594*	1,558,856,686	0.83s	1901
Dhrystone	3,770,115,740	2.61s	1450	3,030,061,475	2.79s	1086	1,590,094,345	1.23s	1293
Whetstone	27,108,532,655	13.23s	2054	488,724,620	0.64s	759*	2,133,926,552	0.99s	2156
peakSpeed2	22,000,023,433	5.76s	3826	11,200,003,894	3.73s	3011	17,100,018,075	4.23s	4052
	Po	PowerPC		Rene	Renesas v850		Syno	psys /	ARC
Benchmark	Simulated Instructions	Run time	Simulated MIPS	Simulated Instructions	Run time	Simulated MIPS	Simulated Instructions	Run time	Simulated MIPS
linpack	3,163,966,113	2.95s	1076	4,991,344,159	4.76s	1051	4,184,162,664	3.67s	1143
Dhrystone	2,205,068,239	1.75s	1260	6,410,133,101	4.01s	1603	3,155,082,476	2.75s	1148
Whetstone	6,424,865,755	3.97s	1622	10,296,940,591	7.41s	1393	7,883,567,047	4.4s	1796
peakSpeed2	22,400,002,937	5.6s	4007	22,400,007,569	3.53s	6364	22,000,002,100	4.05s	5446
All measurements on 3.40GHz Intel i7-3770, Linux, OVPsim 20140127.0 * Hardware Floating Point Instruction									

Example speed of Imperas simulation models

And booting OS can be fast too

MULTICORE DESIGN SIMPLIFIED

C:\Windows\system32\cmd.exe

Info Final program counter : 0x8001d668 Simulated instructions: 1,131,123,567 Info Info Simulated MIPS : 20.0 Info Info Info Info CPU 'ArmVersatileExpress-CA15/cpu_CPU2' STATISTICS Info Туре : arm (Cortex-A15MPx4) Nominal MIPS Info : 1000 Final program counter : 0x8001d668 Info Simulated instructions: 17,224,484,756 Info Info Simulated MIPS : 304.2 Info Info Info Info CPU 'ArmVersatileExpress-CA15/cpu_CPU3' STATISTICS : arm (Cortex-A15MPx4) Info Туре Nominal MIPS : 1000 Info Info Final program counter : 0x8001d668 Simulated instructions: 1,110,697,906 Info Simulated MIPS Info : 19.6 Info Info Info Info TOTAL Info Simulated instructions: 22,568,501,091 : 398.5 Simulated MIPS Info Info Info Info Info SIMULATION TIME STATISTICS Simulated time : 3264.16 seconds Info Info User time : 55.61 seconds Info System time : 1.01 seconds Elapsed time : 56.89 seconds Info : 57.37x faster Info Real time ratio Info CpuManagerMulti finished: Mon Mar 03 20:50:39 2014

_ _ × ArmVersatileExpress-CA15/uart0 o0/input/input0 usbcore: registered new interface driver usbhid usbhid: USB HID core driver oprofile: no performance counters oprofile: using timer interrupt. TCP: cubic registered NET: Registered protocol family 17 VFP support v0.3: implementor 41 architecture 3 rtc-p1031 1c170000.rtc: setting system clock to 2 MAM) ALSA device list: No soundcards found. Freeing init memory: 188K input: ImExPS/2 Generic Explorer Mouse as /device 00.kmi/serio1/input/input1 This root FS contains most basic linux utilities and the Lynx web browser. Kernel config is available through /proc/config.c Welcome to OVP simulation from Imperas Log in as root with no password. Imperas login:

CpuManagerMulti (64-Bit) v20140224.0 Open Virtual Platform simulator from www.IMPERAS.com. Visit www.IMPERAS.com for multicore debug, verification and analysis solutions.

Press any key to continue \ldots

- Boot Linux on ARM Cortex-A15x4 = 6 seconds on Win7 laptop
- Runs simulated Linux applications at 100s of MIPS

ARMv8 simulation using parallel host-cpu resources



Simulation



- Advanced parallel synchronization algorithm for SMP, AMP and hardware accelerators
- Transparent operation to user: No model, tool, software changes
- Total performance on benchmarks recorded up to 16 Billion ins/sec
- Performance advantage 15x over nearest commercial alternative





Virtual Platforms: The Right Performance to Capability Trade-off

- Virtual Platforms (simulators) with Instruction Accurate (IA) models provide:
- Pre-prototype verification
- Effective verification access
- Reasonable execution performance

However, Virtual Platforms require a simulation foundation to be effective

- Standardized modeling technology
- Services for verification tools
- Tool firewall for execution integrity
- Make use of host parallel resources for maximum performance



DESIGN SIMPLIFIED

OVP Standardized Modeling Infrastructure



Open Virtual Platforms[™] (OVP[™]) standardized set of Modeling APIs for platforms, cpu models and behavioral peripheral models



Model Library Extensive (200+), comprehensive open source model collection

OVP Modeling Easy-to-code modeling APIs

Environment Interfaces to SystemC, GDB, etc

Reference Simulator: OVPsim Easy access simulator for running models

Simulator Architecture to provide services for Verification



- Leveraging JIT Code Morphing simulation algorithms for highest possible performance
- Modeling APIs allow processor/platform functionality to be described efficiently while maintaining easy modeling environment
- Imperas technology allows verification and debug tool code to be combined with model and software execution efficiently and unobtrusively



Traditional Debug (1D)



- Debugging application code running on simulated embedded processor
- Trace
 - e.g.: instructions, source lines, register changes
- GDB-like debugger
- Examining registers, variables, source...
- Single step, breakpoints, …
- GUI

Spatial & Temporal Debug (2D)



Spatial

- For AMP/SMP examine applications on multiple cores across the chip
- Debugging peripheral/behavioral models in context of software running on the embedded cores
 - Programmers view, or model source

Temporal

- Considering the sequences of events over time
 - Sequential assertions, breakpoints
 - Using conditioning events to prime breakpoints
 - e.g.: break on next ISR after character input to UART

Layer-Aware Debug (3D)





Layered verification matches layered software architecture Simulator must allow focus and stratification

TRC (SCHD) 242131778: 'mipslel_TCO': scheduler switched ('khelper') TRC (TASK) 242137813: 'mipslel_TCO': do_execve called f TRC (EXEC) 242137813: 'mipslel_TCO': do_execve called f filename=/sbin/hotplug with: TRC (EXEC) 242137813: 'mipslel_TCO': argv virt=0x80 TRC (EXEC) 242137813: 'mipslel_TCO': argv virt=0x80 TRC (EXEC) 242137813: 'mipslel_TCO': envp virt=0x80 TRC (EXEC) 242137813: 'mipslel_TCO': envp virt=0x80 TRC (EXEC) 242137813: 'mipslel_TCO': envp virt=0x80 "PATH=/sbin:/bin:/usr/sbin:/usr/bin"

OS/CPU-Aware Focused Debug Commands analyze layer operation while excluding irrelevant detail (easier to view 1000 tasks operations than 1 Billion instructions trace)



Layer-aware Stratified Analysis Connecting commands through different layers for activity analysis

Page 18

Layered Tool Suite Capabilities (3D)



Rich set of commands to operate at all layers of abstraction

Operating System				
Bare Metal Apps & Midd	leware			
Platform (e.g. Drivers)				
Processor Trace coprocessor registers Trace TLB trace exceptions Trace modes Trace service calls Trace hypervisor calls Trace hypervisor calls Trace secure monitor calls Trace secure monitor calls Trace MT/MP extensions Trace system calls Trace timer Trace cache instructions Trace SIMD extensions Trace instruction Trace register change	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 Instruction coverage Instruction profiling Instruction fault Injection Cache analysis	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 Function profiling Heap checks Stack checks Malloc checks Semaphore checks	Trace console Trace execve Trace scheduler Trace tasks Trace module loads Trace printk
Simulator Break	on messages TCL c	allbacks Full GD	B command set	

Agenda



- Background the changing needs of embedded product development
- "Traditional" Embedded Software development solutions
- Working with Multi-core hardware and software
 - The next generation of Embedded Software development systems
- Example using assertions on software
 - shared memory monitor

Summary

Example 1 - Assertions

Altera Cyclone V Cortex A9MPx2 (AMP Linux/Micrium, SMP Linux) and Nios II (Linux)

MULTICORE DESIGN SIMPLIFIED



© 2014 Imperas Software Ltd.

Verification challenges...



OS Porting, Bring Up and Verification on Altera Cyclone V SoC FPGA

- 1) Linux boot on single core ARM Cortex-A9
- 2) SMP Linux boot on dual core ARM Cortex-A9
- 3) RTOS boot on single core ARM Cortex-A9
- 4) AMP boot on dual core ARM Cortex-A9
- 5) Linux boot on single core Nios II
- 6) SMP Linux boot on dual core ARM Cortex-A9 plus Linux boot on Nios II

Assertions: Memory Access Monitor Accelerates AMP Platform Debug



- Memory access monitor is just C code, loaded into simulation environment
- When simulation is run, monitor produces warning if memory access rules are violated

//								
// Define watch areas for memory and peripherals defined in the platform								
//								
<pre>memWatchT amcWatch[] = {</pre>								
// name	watchLow	watchHigh	allowedCPUs					
{ "Linux memory",	Ο,	0x2fffffff,	LINUX_CPU },					
{ "uCOS memory",	0x30000000,	0x31ffffff,	UCOSII_CPU },					
{ "gmac0",	0xff700000,	0xff700fff,	LINUX_CPU },					
{ "emac0_dma",	0xff701000,	0xff701fff,	LINUX_CPU },					
{ "gmac1",	0xff702000,	0xff702fff,	LINUX_CPU },					
{ "emac1_dma",	0xff703000,	0xff703fff,	LINUX_CPU },					
{ "uart0",	0xffc02000,	0xffc02fff,	LINUX_CPU },					
{ "uart1",	0xffc03000,	0xffc03fff,	UCOSII_CPU },					
{ "CLKMGR",	0xffd04000,	0xffd04fff,	LINUX_CPU },					
{ "RSTMGR",	0xffd05000,	0xffd05fff,	LINUX_CPU },					
{ "SYSMGR",	0xffd08000,	0xffd08fff,	LINUX_CPU },					
{ "GIC",	0xfffec000,	<pre>0xfffedfff,</pre>	LINUX_CPU },					
{ "L2",	0xfffef000,	Oxfffeffff,	LINUX_CPU },					
{ 0 } /* Marks end of list */								
};								

Warning (AMPCHK_MWV) cpu_CPU0: AMP write access violation in uart1 area. PA: 0xffc03008 VA: 0xffc03008 Warning (AMPCHK_MWV) cpu_CPU0: AMP write access violation in uart1 area. PA: 0xffc0300c VA: 0xffc0300c Warning (AMPCHK_MWV) cpu_CPU0: AMP write access violation in uart1 area. PA: 0xffc03010 VA: 0xffc03010 Warning (AMPCHK_MRV) cpu_CPU1: AMP read access violation in Linux memory area. PA: 0x00000020 VA: 0x0000020

Summary of verification example

- 1) Linux boot on single core ARM Cortex-A9
 - Bug found in Linux kernel preemptive scheduling
 - Linux boots and runs, but does not switch tasks properly
 - Not observed in previous virtual platform (different virtual platform vendor) using much slower model of ARM Cortex-A9MPx2
 - Could not run multiple applications for long enough simulation to observe the bug
- 2) SMP Linux boot on dual core ARM Cortex-A9
 - OK no problems found
- 3) RTOS boot on single core ARM Cortex-A9
 - Bugs found and fixed in GIC register accesses using OS-aware tools
- 4) AMP boot on dual core ARM Cortex-A9
 - Bug found in Linux accesses of GIC registers
 - Bugs found in RTOS access of Linux's reserved memory
- 5) Linux boot on single core Nios II
 - No problems found
- 6) SMP Linux boot on dual core ARM Cortex-A9 plus Linux boot on Nios II
 - No problems found



MULTICORE DESIGN SIMPLIFIED

eras

Summary



- Simulation is necessary but not sufficient
 - Fast simulation finds more bugs
 - Making use of multi-core host is even better
- Trace, temporal, and multi-core debug are essential for AMP/SMP systems
- 'Layer-aware' analysis makes debug manageable
 - Allows focus at different levels of abstraction
- Ability to extend functionality and write own tools are the key to providing efficient development environments

Conclusions



- It is inevitable that simulation will form the basis of the next generation of embedded software development methodology
- Ensure your chosen simulator is fast, has a standardized modeling capability, and has the ability to include integrated advanced tools
- 2D and 3D verification, analysis and debug tools are essential for multi-core designs
- To find the most complex bugs and ensure product quality an advanced verification approach is needed using layered, customizable tools

Thank you



- For more modeling/model information
 - www.OVPworld.org
- For technology/product information
 - www.imperas.com

Questions?