Embedded Software Verification
Using Virtual Platforms

Duncan Graham
DAC Virtual Platform Workshop
29 July 2009
Embedded Software Verification Using Virtual Platforms

- Embedded software/systems issues: complexity

- Key software development tasks
  - OS
  - Drivers
  - Applications

- Virtual platform infrastructure: necessary but not sufficient

- Requirements for software development
Embedded Software Verification Using Virtual Platforms

- Embedded software/systems issues: complexity

- Key software development tasks
  - OS
  - Drivers
  - Applications

- Virtual platform infrastructure: necessary but not sufficient

- Requirements for software development
The Future of SoCs is Multicore

Future development work is all MP

ARM roadmap slides from ARM DevCon 2008
But, To ARM’s Dismay, the Future (and Present) is Heterogeneous
Multicore SoC Programming
Challenge: Parallel Software on Parallel Hardware

1. Which programming model and methodology do I adopt for my customers…?

2. How do I parallelize the application?

3. How do I build a programming environment?

4. How do I develop and verify a range of reference applications for an MPSoc?

5. Can I automate the software verification?

6. How do I deliver the software and programming environment to customers?

Which CPU do I choose?

Is this the right bus architecture?

Which hardware IP do I need to develop?
Hardware / Software Complexity Growth

- From 1990 to 2000 chip gate capacity increased about 100x
- But design complexity increased by 100,000x
- This resulted in RTL functional verification taking about 70% of design resources
- New technologies, new methodologies
  - Constrained random generation
  - Coverage driven verification
  - Dynamic assertion checking
  - Better debugging tools

- Software code for embedded systems is doubling annually
- Amount of concurrency is doubling every 18 months
- Complexity of software driven by concurrency, shared resources
- New technologies, methodologies needed for functional and performance verification of embedded software
  - Simulation (virtual platforms)
  - ???
- Conclusion: borrow from the progress made in RTL functional verification
Embedded Software Verification
Using Virtual Platforms

- Embedded software/systems issues: complexity

- Key software development tasks
  - OS
  - Drivers
  - Applications

- Virtual platform infrastructure: necessary but not sufficient

- Requirements for software development
Embedded Software Tasks: Hardware-Related Software

- Boot OS
- Static Compiled Drivers
- Dynamic Compiled Drivers (LKMs)
- Bare Metal Applications

- Which OS?
- New version?
- New port?
- SMP?
- AMP?
- Performance bugs

- New OS?
- New driver?
- How to verify functionality?
- How to debug core + peripheral?

- Multicore?
- Shared memory?
- Functional bugs
- Performance bugs
OS Issues

- How to …
  - Port OS to new platform
  - Port new version of OS
  - Go from single core to multithread/multicore OS
  - Analyze performance issues
Static & Dynamic Driver Issues

- How to …
  - Port driver to new OS
  - Develop driver for new peripheral
  - Verify driver functionality
  - Debug core + peripheral simultaneously
Issues With Bare Metal Applications

- How to …
  - Analyze shared memory
  - Develop on multicore platforms
  - Find functional bugs
  - Find performance bugs

How to …
- Analyze shared memory
- Develop on multicore platforms
- Find functional bugs
- Find performance bugs
Complex Platforms (AMP) Bring It All Together

Data Processing

System Management

MP SoC users may only use 1 or 2 cores for Linux, the remainder for data processing applications

Boot OS

Compiled Drivers

LKM Drivers

Bare Metal Applications

Produce r1
Produce r2
Produce r3
Consumer

Shared Memory

r

Lock

FIFO

Thread 0
Local

Thread 1
Local

Thread 2
Local

Thread 3
Local

Thread 0
Shared Memory

Thread 2
Shared Memory

Thread 3
Shared Memory

VPE0

VPE1

MIPS32
4KEc

Smartprom
(Linux)

Memory
(RAM)

Local Bus

Local Bus

Local Bus

PCI BUS

PCI BACK

PCI Config

IDE

IntControl
(PiIX4)

USB
(PiIX4)

PM
(PiIX4)

RTC
(PiIX4)

Timer
(PiIX4)

VGA

SDL

KbControl
(SuperI0)

UART (TTY0)
(SuperI0)

UART (TTY1)
(SuperI0)

UART (TTY2)
(16450)

SysControl
(GT64120)

UART (TERA)

FPGA

Melat

Drivers

Drivers

LKM

Drivers

Bare Metal

Applications

Compiled

Drivers

Drivers

LKM

Drivers

Bare Metal

Applications

Page 13

© 2009 Imperas

29-Jul-09
Embedded software/systems issues: complexity

Key software development tasks
- OS
- Drivers
- Applications

Virtual platform infrastructure: necessary but not sufficient

Requirements for software development
Simulation is Necessary

- Unimaginable to build a SoC today without simulation of the hardware
- Similarly, embedded software needs simulation as part of the development flow
  - Earlier start to development
  - Easier to find/fix bugs

- Virtual platform needs
  - Open source models developed in non-proprietary language
  - Easy to develop new models
  - Fast simulation
Open Virtual Platforms (OVP)

- **Modeling APIs**
  - Publishing of C OVP APIs for Processor, Peripheral, and Platform modeling
  - Documentation & header files

- **Open source library of models**
  - C source of models written to C OVP APIs
  - Processor models of ARM, ARC, MIPS, OpenRisc OR1K, x86, …
  - Peripheral models of standard embedded devices
  - Example embedded platforms in C, C++, SystemC, TLM-2.0
    - Including full platforms that boot operating systems like Linux, Nucleus

- **OVP reference simulator, free for non-commercial use**
  - Runs processor models fast, 500 MIPS typical
  - Interfaces to GDB via RSP/socket
  - MP capable, scalable and very efficient
  - Can encapsulate existing processor models (ISS)
  - Callable with C/C++/SystemC/TLM-2.0 wrapper

- [www.OVPworld.org](http://www.OVPworld.org)
Application is Fibonacci series generated on one and read by second processor from shared memory

- Local memory
  - heap and stack
- Shared memory
  - program and data

- Easy to create platform and use
  - C:\> platform .exe application .elf
  - Loads application into shared memory and runs it
  - On 3GHz PC runs up to 500 MIPS
Local memory for heap and stack
- Shared memory for program and data
- Application is Fibonacci series generated on one processor, and read by second from shared memory
- Very simple to use and runs very fast
OVPsim booting uClinux on ARM Atmel AT91sam7

```
rmdisk: root's filesystem found at block 0
rmdisk: loading 721 blocks [1 disk] into root disk... done.
VFS: Mounted root (rmdisk's filesystem) read-only,
Formatting init memroy: 400
Shell allowed to run File/Menu
Command: bootm
Command: /dev/ram/rootfs.img /dev/real
Command: mount -t proc proc /proc
Command: mount -t ext2 /dev/real /usr
Command: mkdir /var/log
Command: mkdir /var/run
Command: mkdir /var/junk
Command: mkdir /var/empty
Welcome to

GNU/ARMulator support by david@estergar.com
For further information check:
http://www.uClinux.org
Execution finished, exiting.
```

Boot uClinux
Boot Linux on Windows PC…
Easy to run: platform.exe vmlinux
OVPsim Heterogeneous Platform
ARM Nucleus / MIPS SMP Linux

Run Platform

ARM920T
- SSRAM
- SDRAM
- PIC
- AHB Decoder
- UART
- LCD Controller
- Keyboard/Mouse
- config regs
- LED
- RTC
- MMC Interface
- UART
- GPIO
- Flash

MIPS32
- 34Kc
dualcore
- UART (TTY2) (16450)
- Memory (RAM)

LOCAL BUS
- SSA
- Bus

PCI BUS
- IDE
- IntControl (PIIX4)
- PIIX4 (Base)
- USB (PIIX4)
- PM (PIIX4)
- RTC (PIIX4)
- Timer (PIIX4)

Memory (RAM)
- Malta FPGA
- SysControl (GT64120)
- UART (TTY1) (SuperIO)
- UART (TTY0) (SuperIO)
- VGA
- KbControl (SuperIO)

Keyboard
Mouse

telnet localhost 9999


Page 21
Case Study 1: Automotive Electronics

- Key need is running more simulations
- Platform is simple microcontroller based on ARM7
- Simulation speed of the vendor simulator was too slow for complete regression runs in reasonable amount of time
- Used Open Virtual Platforms to achieve 50x simulation speed improvement

- Simulation speed is enough
Embedded Software Verification
Using Virtual Platforms

- Embedded software/systems issues: complexity

- Key software development tasks
  - OS
  - Drivers
  - Applications

- Virtual platform infrastructure: necessary but not sufficient

- Requirements for software development
Case Study 2: Home Entertainment System

- **Original SoC**
  - 1 general purpose core
  - Running Linux 2.6.n
  - 12 proprietary DSPs
  - 3 DDR1 memory banks
    - 1 dedicated to general purpose core

- **Cost Reduction SoC**
  - 1 general purpose core
  - Running Linux 2.6.n+m
  - 2nd general purpose core for housekeeping
  - 12 proprietary DSPs
  - 2 DDR2 memory banks
    - Not dedicated to general purpose core

- Cost reduction SoC has hardware fully verified, uses existing software
- Chip brought up in lab, fully functional, but ...
- Runs at half the speed in about 30% of the operating scenarios
- Insufficient diagnostics on chip for debug
- Simulation platform built, but by itself not able to debug problem
Simulation Is Necessary, But Not Sufficient

- Simulation is only the infrastructure
- Just like on the hardware side, with more complexity in the software, new tools and methodology are needed
- Need to find functional bugs and performance bugs
- Need new metrics
New Tools/Methodologies

- Non-intrusive: cannot have “heisenbugs”
- Work on heterogeneous, AMP platforms
- Address functional and performance bugs
  - Deadlock, race, stall, …
  - Cache thrashing, performance bottlenecks, …
- Provide introspection of the complete platform
  - Processors, peripherals, memory
- Metrics
  - Code coverage
  - Functional coverage
  - ???
Communication implemented as shared memory protected by semaphores ...

=> Risk of deadlocks !!!

- Control Simulation
  - Processors and peripherals
- Force application into potential error state
- Create worst case conditions quickly
User Defined Dynamic Assertions

- Create a specific user assertion
  - *Buffer is not accessed without a lock*
- Complex sequence of operations
- Over multiple processor cores

Data Buffer written without lock on one channel!

1. Check for lock
2. Acquire lock
3. Read & write data
4. Release lock
Platform Introspection for Debugging Driver Software

- Debug the software running on processor(s) with all platform executing
- Introspection enables view of other platform components, including all registers
- Ability to switch from code on different processors onto code inside behavioral models
- Standard gdb features
Conclusions

- Embedded software development is hard, and getting harder very quickly
- Need to recognize the real issues – the development tasks that need to be successful – and address those needs
- Virtual platforms (models and simulation) are necessary, but …
- New tools and methodologies, including metrics, are required