Modern Software Development Methodology for RISC-V Devices

RISC-V Workshop – Chennai, India
July 18-19 2018

Kevin McDermott – Imperas Software Ltd
The size of the RISC-V market share will depend more on the software ecosystem than on specifics of RISC-V implementations.

“nobody designs an SoC without extensive simulation based testing, we believe that nobody should be developing embedded software without simulator based testing”
Agenda

- Challenges in embedded software development
- New methodology is needed
  - Continuous Integration
  - Continuous Test
- Test with hardware – necessary evil? – help or hindrance?
- Adoption of Continuous Test for embedded
  - And how simulation is used
- Worked example
- Summary
New Markets With New Software Requirements

- Schedule
- Quality
- Reliability
- Security
- Safety
- Engineering productivity / automation
- Predictability on software development schedules
- Unknown / unmeasurable software delivery risk
Processor Platform Configurations

Single core, simple

Multi-core shared memory

Many-cores

Heterogeneous

Booting OS, eg Linux
Embedded Software Methodology is Evolving

- Modern software development moving away from traditional V-shaped development flow to …

- Agile
- Continuous Integration
- DevOps
Modern Development Methodology: Agile, Not V-Shaped

CONTINUOUS INTEGRATION

- Repository
- Commit
- Trigger
- Compile
- Build
- Test
- Fail
- Pass
- Errors
- Packaging
- Stress Test
- Full Application
- QA Test
- Tester
- Release / Deploy

Developer

Code & Tests
The Challenge…

- How to apply this methodology and gain these benefits in the Embedded Software world…

- In this talk the focus is only on the Continuous Integration piece of the Agile methodology
Motivation for Change: Benefits of Continuous Integration

- Better code structure and quality
  - Frequent code check-in pushes developers to create modular, less complex code
  - Enforces discipline of frequent automated testing
  - Software metrics generated from automated testing and CI (such as metrics for code coverage, code complexity, and feature completeness) focus developers on developing functional, quality code, and help develop momentum in a team

- Easier debug
  - When unit tests fail or a bug emerges, if developers need to revert the codebase to a bug-free state only a small number of changes are lost

- Fewer major integration bugs
  - Immediate feedback on system-wide impact of local changes
  - Integration bugs are detected early and are easy to track down due to small change sets. This saves both time and money over the lifespan of a project.
  - Avoids last-minute chaos at release dates, when everyone tries to check in their slightly incompatible versions

- Constant availability of a "current" build for testing, demo, or release purposes
First, some of the problems

- Multiple code streams (release versions) to manage
  - Development, under test, in field
- Many hardware/OS targets: processor variants (ARM, MIPS, Renesas), OS versions (Linux xyz, 32/64, Windows 7/10, 32/64) and a large amount of common code between targets
- With many teams and tasks all in parallel

- Access/configuration of available hardware
  - (e.g. customer USAF 1 prototype, 2 weeks to get access, shift work)
  - (recall: old computers, card decks, or early timeshare 30 mins per day)
- Not just about testing something works
  - ensure what you think is being tested is being tested, i.e. need metrics
- And then, need to run 1,000s of tests on many targets to validate software changes

- And with many common libraries, any change proliferates to many projects
  - need to re-validate ALL projects
OK – so automation can address this

- Continuous Integration (CI)
  - Create a build server so that any change builds software
    - for multiple code streams
    - for multiple targets

- Then require Continuous Test (CT)
  - For each build for each target run N test cases
    - Quantify correctness
    - Coverage
    - Performance
Continuous Integration Continuous Test (CICT)
Now, how to test?

- Use x86 PC native?
  - e.g. x86 compile and run – works well for simple code
  - What about binary libraries, e.g. for ARM CMSIS
  - What about CPU architectures with restrictions e.g. reduced address space, available memory, …

- For embedded, real target code should be used
  - Cross compile
    - Use correct binary libraries
    - Use correct instruction streams
  - Need to run
    - on real cpu architecture
    - with real data
    - with real stimulus
  - Need to capture real outputs
But using real hardware is a problem

- Need Device Under Test and environment (world) both available in hardware
  - How to stimulate the environment, sensors, buttons
  - How to monitor responses and measure correctness in both value and time
- Hardware may require manual intervention, which is prone to errors
- Can a hardware testbench do everything you need?
  - For example, trigger interrupt after 15msecs of xyz event
- Can hardware be set into the correct state to start a test sequence?
- It can be hard to model the real world, and hard to make reproducible
And there are more issues

- How much access can you get for your testing
  - Including setup and versioning of the hardware
- And can you have several users using in parallel
- And prototypes are costly to acquire and maintain
- And they only run in real time
  - Can they run faster to get more testing done?
    - (e.g. customer Audi – 6 months of road data need to run tests overnight)
    - (e.g. GPS chip sends position every second)
- …
Adopting Continuous Test for Embedded Needs Simulation

- Imagine a software build system without access to ‘make’ or ‘ant’
  - they enable effective build automation

- Simulation enables the effective automation of testing embedded systems as part of a CICT environment
- Simulation enables full automation
  - with no manual intervention

- Use of hardware is just too hard

=> Virtual Platforms (simulation) enable CICT for embedded
Modern Software Methodology: Virtual Platforms Complement Hardware-Based Software Development

- Virtual platform based methodology delivers controllability, visibility, repeatability, automation, access
  - 75-90% of bugs are functional, and can be found using software simulation testing
- Testing of timing sensitive software, and final testing, still needs to be done on hardware

Application Layer: Customer Differentiation

Middleware: TCP/IP, DHCP, LCD, …

OS: Linux, FreeRTOS, μC/OS, ThreadX, …

Drivers: USB, SPI, ethernet, …

Virtual platforms – software simulation – provide a complementary technology to the current methodology

Hardware or Virtual Platform
So what are we talking about here in terms of simulation

- An Instruction Set Simulator (ISS) or
- A Virtual Platform (Virtual Prototype) simulation
  - CPUs, memories, peripherals
  - Test components, stimulus generation
  - Models of the world/environment
  - Verification/validation tools
Virtual prototype runs same software as first Si samples

- Common Software Development Environment
Virtual Platforms Provide a Simulation Environment Such That the Software Does Not Know That It Is Not Running On Hardware

- The virtual platform is a set of instruction accurate models that reflect the hardware on which the software will execute
  - Could be 1 SoC, multiple SoCs, board, system; no physical limitations
- Runs the executables compiled for the target hardware
- Models are typically written in C or SystemC
- Models for individual components – interrupt controller, UART, ethernet, … – are connected just like in the hardware
- Peripheral components can be connected to the real world by using the host workstation resources: keyboard, mouse, screen, ethernet, USB, …
- Typical performance is 200-500 million instructions per second

Andes certified N25 & NX25
Imperas Reference Simulator
Software Verification, Analysis & Profiling (VAP) tools
- Trace
- Profile
- Coverage
- Schedule
- Memory monitor
- Protocol checker
- Assertion checkers
- ...

JIT simulator engine

Multiprocessor / Multicore Debugger

Eclipse IDE
Simulation is a key component of embedded CICT environment

CONTINUOUS INTEGRATION & CONTINUOUS TEST

Compiler
Build
Test

Developer
Code & Tests

Tester
Packaging
Stress Test
Full Application
QA Test
Release / Deploy

Virtual Platform Simulation
Demonstration

- Imperas simulation used with Jenkins environment
  - Jenkins is a widely used, open source CICT environment for general software
- Edit, compile, local test, check in -> triggers build
- Successful build -> triggers testing
- Testing completion -> triggers results
Jenkins ‘Items’

- Simple example is jpeg encoder targeting 4 different target ISA (ARM, MIPS, Renesas, Altera) with 11 different algorithm arguments to test

Can use scripts, like bash e.g. Test

Jenkins Pipeline
Create ‘Stages’ from items

- Note use of parallel & serial
Stages running (1)

- Can trigger start of run from code check-in or directly

Can see results at end

- Each test run records test results from its group
- Final task stage in pipeline collates

Drill down to see failures

- See how tests perform over each run
- Management get a dashboard for visibility of project status
Demo Wrap up

- This showed simple example of developing and testing code for embedded targets using cross compilers to build and ISS to execute
- Used CICT system (Jenkins) to manage processes, data, and results
- Very simple to set up / manage
- Automates build/test – and can provide high level monitoring and results to developers
- Easily extends to full platforms using Virtual Platform simulations
  - e.g. testing applications under operating systems
Agenda

- Challenges in embedded software development
- New methodology is needed
  - Continuous Integration
  - Continuous Test
- Test with hardware – necessary evil? – help or hindrance?
- Adoption of Continuous Test for embedded
  - And how simulation is used
- Worked example
- A little more about Imperas solutions
- Summary
Over 200 Fast Processor Models in OVP Library
Models and platforms are open source (Apache 2.0 license)

ARM®: Models for ARMv4™, v5™, v6™, v7™ and v8™ architectures

MIPS®: Models for nanoMIPS, microMIPS, MIPS32 and MIPS64 architectures
  ▪ Verification, licensing, and distribution relationship

Renesas: Models for RH850, V850 architectures; 16 bit microcontroller cores
  ▪ RH850G3, V850 ES, E1, E1F, E2; RL78, M16C cores

Synopsys (ARC): ARC6xx, ARC7xx, EM families

RISC-V: RV32/64 GCN
  ▪ SiFive E31, E51, U54
  ▪ Andes N25, NX25

Altera Nios II

Xilinx Microblaze

“OVP is addressing key issues in software development for embedded systems. By supporting the creation of virtual platforms, OVP is enabling early software development and helping expand the ARM user community.”

Noel Hurley, VP Business Development, ARM
Key Technology: Multicore Development, Debug & Test Tools

- Verification, Analysis & Profiling (VAP) software tools
  - Non-intrusive: no modification of software or model source code
  - Users can easily define custom tools
- 3Debug™ capability for debug of software on complex, heterogeneous platforms
- Tools at the appropriate and valuable levels of abstraction, granularity
  - Instruction tracing shows everything at lowest level of abstraction, no granularity
  - Function tracing and OS tracing show higher levels of abstraction
  - Instruction subset tracing, e.g. SIMD or hardware virtualization, show finer granularity
RISC-V EPK based on SiFive U54-MC

The Virtual Platform Provides a Simulation Environment Such That the Software Does Not Know That It Is Not Running On Hardware

https://www.sifive.com

Under 10 seconds to get to booted Linux login prompt!
Imperas Solution Contents

Methodology
Collaboration with customers, vendor ecosystem

Models
200+ CPU models
100's peripheral models
50+ platforms

Tools
Leading simulation, debug, software verification tools

Resources
Imperas and partners
Model development
Tool development

Training
Imperas and partners
On-site, customized agenda
Agenda

▪ Challenges in embedded software development
▪ New methodology is needed
  ▪ Continuous Integration
  ▪ Continuous Test
▪ Test with hardware – necessary evil? – help or hindrance?
▪ Adoption of Continuous Test for embedded
  ▪ And how simulation is used
▪ Worked example
▪ A little more about Imperas work with RISC-V
▪ Summary
RISC-V Status

- Processor models
  - RV32/64 GCN, RV32EC
  - Andes N25, NX25 including custom instructions
  - SiFive Mi-V (RV32IMA), E31, E51, U54
  - Added capability to enable easy addition of custom instructions, registers, etc. to processor model via side library
    - Does not perturb known, validated model source
    - All Imperas tools supported for complete model

- Platforms
  - Various RV32 models booting FreeRTOS
  - Single core RV64GC booting Linux in under 5 sec
  - Quad core RV64GCN (SiFive U540 platform) booting SMP Linux in about 7 sec

- Imperas tool support for RISC-V
  - MPD debugger for heterogeneous, multiprocessor/multicore platforms and driver-peripheral co-debug
  - Verification, Analysis and Profiling (VAP) tools including tracing, profiling, code coverage, OS-aware tools, timing estimation (paper at Embedded World)
RISC-V Compliance Work

- Active member of RISCV.org Compliance Working Group
  - Driving architecture of test environment
  - Providing infrastructure for better quality compliance tests
  - Working with Formal Model Working Group to develop device configuration file format

- Migrating internally developed tests to be useable in device compliance testing

- Working with customers’ RTL developers to check the compliance of their devices to the RISCV.org standards
  - Using Imperas simulation/modeling technology
Imperas Users Benefit From Improved Software Quality, and Reduced Schedules & Costs

- Key technologies: 200+ processor model library, large peripheral model library, fastest simulator, advanced Verification, Analysis and Profiling (VAP) tools
- Solutions for embedded use cases: custom CPU, semiconductor vendors, embedded systems developers
- Experience with Continuous Integration and Continuous Test usage
Thank you

Kevin McDermott
VP Marketing, Imperas Software Ltd.
KevinM@Imperas.com

For more information:

www.Imperas.com
www.OVPworld.org

Contact us: info@Imperas.com