How Good are Open Source Development Tools

Glenn Seiler Sr. Director, Market Development

© 2008 Wind River

The Development Life Cycle



WIND RIVER

Diversity of Tools



3

Typical Build Cycle



WIND RIVER

Today's tools are getting the job done

BUT.....

Multicore changes everything

WIND RIVER

© 2008 Wind River

Challenges of Debugging with Multicore

- How to effectively manage shared resources, such as memory and peripherals
- How to debug OS and application code over multiple cores, boards, and systems
- How to synchronize on-chip debugging across multiple cores served by one or more JTAG connections
- How to debug homogeneous and heterogeneous cores on a single die, then coordinate the debugging over an entire system
- How to use JTAG debugging effectively with agentbased debug and ensure a smooth handoff between different debug tasks
- How to optimize use of the JTAG interface for responsiveness and throughput

Even harder with Virtualization and Multicore



WIND RIVER

Kernel & User Space Management

Hardware Bring-Up	Firmware Driver & BSP	Application Software	Code QA & Test	System Test	Deployment & Field Mgmt.
Application Development - Eins - Wind River Work Ele Edit Refactor Navigate Search Project Analyze Proj 32 File Sym User Build Arguments: Configuration User Build Arguments: Configuration Buser Space Configuration Buser Space Configuration Content Configuration Co	Abench Target Run Window Help Image: Second	Image:	Image: Size (kB) 12 13 14 15 16 1.3.34 13 14 15 16 1.3.34 17 18 19 10 11 12 13 14 15 16 1.3.34 20 21 22 247534kB Add 16 17 18 19 10 10 10 11 12 13 14 15 13 14 15 16 13 14 15 16 17 16 17 16 17 18 18 18 18 18 18 18 18	 Linux Kerne Configuration View and kernel cordination Include ar component Include ar component Find and we dependent Linux User Son Configuration View, add remove Ritthe target Build pack Prepare pare 	I on Editor modify the figuration of exclude of exclude
Build succeeded for pro	Ject Eins. Start time: 2007-0End time: 2007	2007-01-23 20:14:09 (Elapsed Time: -01-23 20:14:09, Elapsed time: 04:36.	04:36)	patching	
8		© 20	008 Wind River	V	VIND RIVER

Patch Management

Deployment & Firmware Application Code QA System Hardware Field Mgmt. **Driver & BSP** Software Test Bring-Up & Test Apply Patcl Linux Patch Model Verify Patch The tree shows the contents of the patch. A checked item indicates that a patch can be applied successfully. To remove an item, clear its Never touch the original source Patch options Ignore leading path name segments: 1 🛩 Reverse patch **Provide changes as patches** Maximum offset (+/- lines): 0 Ignore white space 2 Maximum fuzz factor: Create inline rejection Patches allow you to track Offset Fuzz Factor changes and versions ✓ 1006,8 -> 1006,8 1327.8 -> 1327.8 0 0 ✓ ± 58.6 -> 58.7 0 0 Patch management: ▽ 🗹 📄 src/main/Makefile.tmp Text Compare Classic open-source tools (diff, Original Result 1005echo "## Use this shell script to re 1005echo "## Use this shell script t 🔺 1006echo "## restoring your configur 1006echo "## restoring your configuration patch, and quilt) 1007echo "##" >>\$configstatus 1007echo "##" >>\$configstatus 1008echo "" >>\$configstatus 1008echo "" >>\$configstatus 1009 for var in CC CPP OPTIM CFLAGS CFLAGS 1009 for var in CC NCC CPP OPTIM NCFLA Browse, apply, and preview 1010LDFLAGS_SHLIB LDFLAGS_SHLIB_EXPOR 1010 LDFLAGS_SHLIB_EXPORT LIBS 1011 eval "val=\"\\$\$var\ 1011 eval "val=\"\\$\$var\ 1012 if ["x\$val" != "x"]; then patches 1012 if ["x\$val" != "x"]; then 1013 echo "\$var=\$val" |\ 1013 echo "\$var=\$val" |\ 1014 sed -e 's:\(["\$\\]\):\\\1:g' 1014 sed -e 's:\(["\$\\]\):\\\1 Vendor specific enhancments 1015 1015 -e 's:\([A-Z]*=\):\1":' -e 's:\([A-Z]*=\):\1' 1016 -e 's:\$:" \\:' >>\$configs 1016 -e 's:\$:" \\:' >>\$cor fi 1017 fi 1017 such as highlighting patched 1018done 1018done 1019echo \$SEO "./configure \\" >>\$configs 1019echo \$SE0 "./configure \\" >>\$cor 1020for arg 1020for arg code and displaying the patch 1021do 1021do * 4 4 3 number Update ? < Back Next > Finish Cancel

Analyze & Visualize Behavior



A set of dynamic visualization and analysis tools to help uncover memory leaks, do performance analysis, track changes in system variables, and locate untested code

<u>File E</u> dit <u>P</u> roject	Window Hel	elp		
163 while(keep 164 while(keep 165 {* 166 /* conth 167 conth 168 f(numth 169 { 170 happen 171 numth 172 bags 173 else 174 { 175 keep 176 } 177 takkel 178 } 181 * 182 * coverage 183 * 184 * 185 void coverage 186 { 1872 while(ever 188 * 190 coverage	Going) ins a deliberat last loop if come er >= 0 8& (co soing = 1; ier-; iong = 0; ay(5); leartBeat - a ta everything is eHeartBeat() ythingsOK) : beat message eSleep();	bug, sets everythingsOK to 0 here verageError is not equal to 0 % overageError == 0 (everythingsOK = ask that beats once a second as long as s ok.	• numIter)))	
193) 194 195 /* 196 * coverage Highlight: IV Func IV Exit IV Block IV Deci IV Conc	ileep - a simple tion Coverage Coverage : Coverage sion Coverage dition Coverage	e sleep function that delays the task for 	FullyCovered PartiallyCovered NotCovered	Custom Custom

CoverageScope is now available for Linux

- Helps complete your test suite by showing what's been tested and what hasn't
- Important tool for improving your product quality—now available for Workbench 2.6

Summary - The Open Source Tools Cycle

Edit, Compile, Debug



11