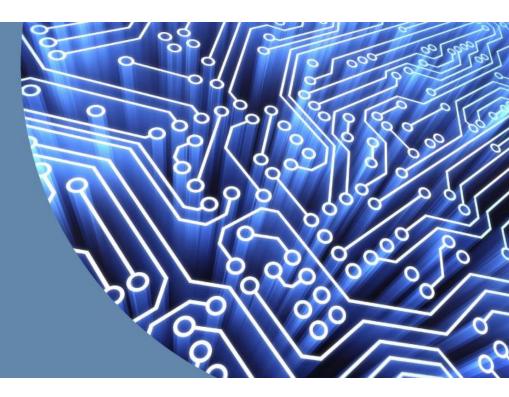


Combining Arm & RISC-V in Heterogeneous Designs Gajinder Panesar, CTO, UltraSoC gajinder.panesar@ultrasoc.com RISC-V Summit

3 – 5 December 2018 – Santa Clara, USA



Problem statement Deterministic multi-core Example scenarios In-field analysis/ML Summary



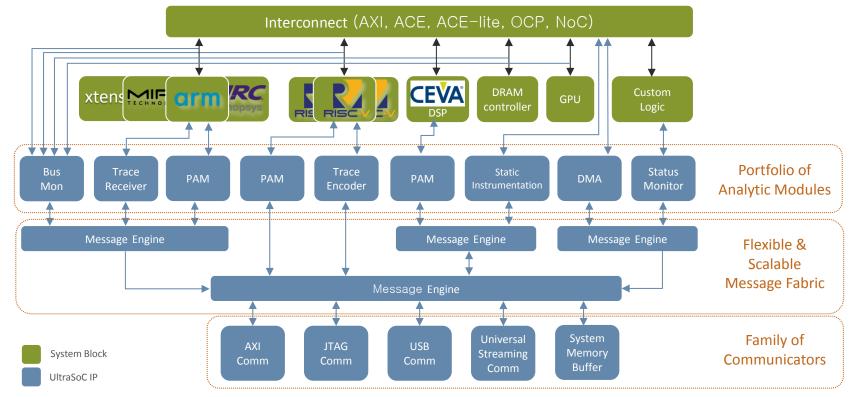




- It is not about the ISA(s)
- It is not about the core(s)
 - Compute is largely 'solved'
- The challenge today is systemic complexity, for example:
 - Ad-hoc programming paradigms
 - Processor-processor interactions
 - HW/SW interactions
 - Interconnect, NoC & deadlock
 - System not architected

Advanced Debug/Monitoring for the Whole SoC





5 December 2018

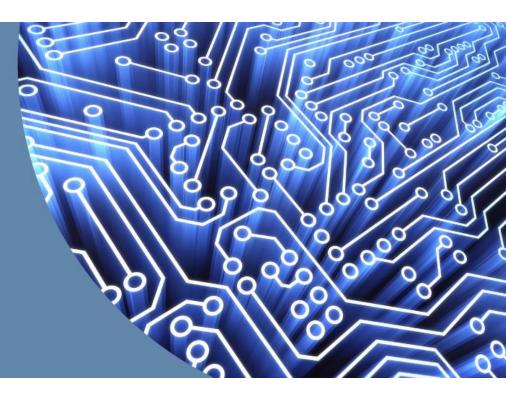




- A coherent architecture to debug, monitor and provide rich data for run-time analytics
 - Silicon IP is highly parameterizable allows customers to trade hardware resources and thus silicon area
 - Hardware resources are configurable at runtime
 - Allows reuse of hardware resources for different scenarios and different algorithms
 - Help with security and safety of systems
 - Hardware provides data so CPU load is small

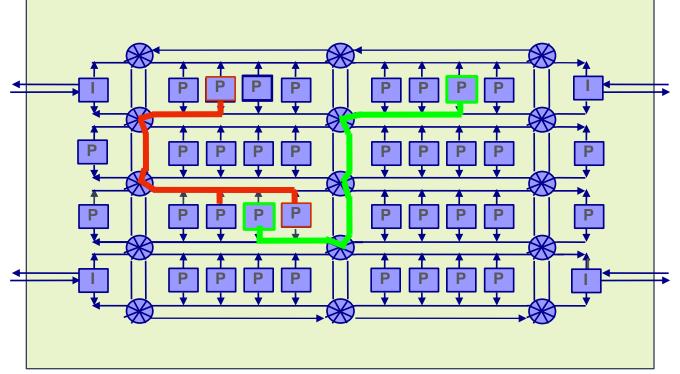


Problem statement
 Deterministic multi-core
 Example scenarios
 In-field analysis/ML
 Summary





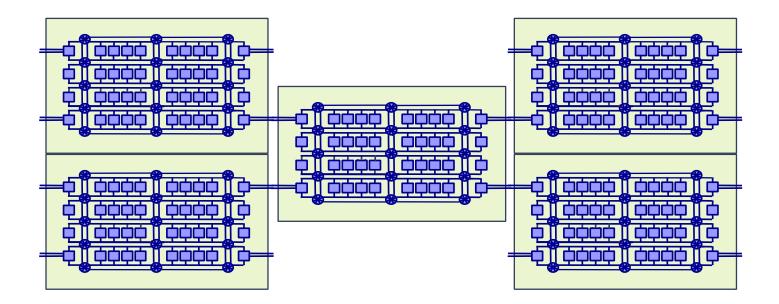




picoArray concept, circa 2000



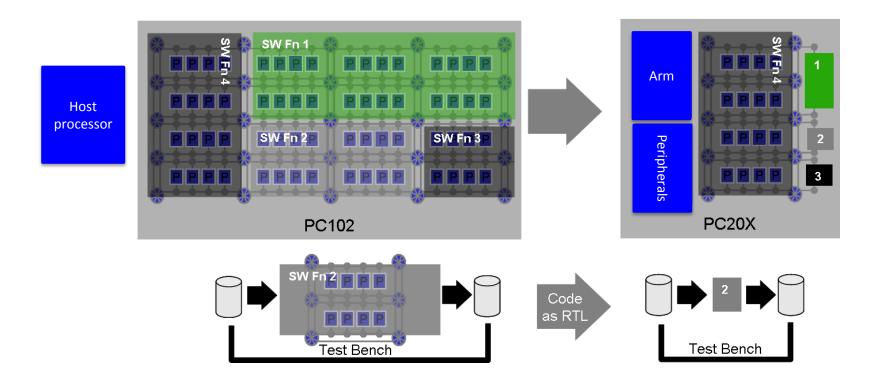




picoArray concept, circa 2000

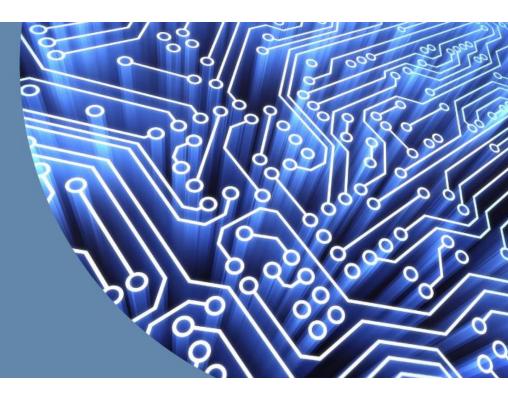








Problem statement
Deterministic multi-core
Example scenarios
In-field analysis/ML
Summary







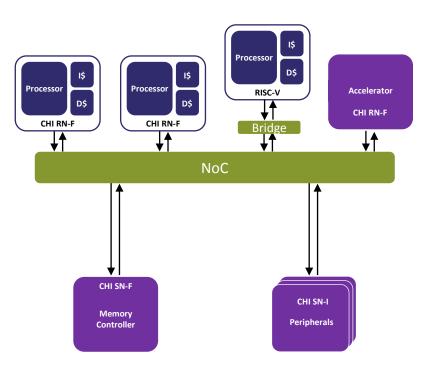
- There is a need for heterogeneous architectural and modelling exploration systems
 - Be able to feed in run-time system data to close the loop
- There is a need for true heterogeneous core tool chain
- This is especially true for debugging tools
 - Open source tools such as GDB and OpenOCD need to handle this in an efficient manner
 - Strangely, not everyone likes or wants open source tools it is True!
 - Need one cockpit for the different cores in a system
- Need to have tools that help with run-time visibility
 - These need to have open APIs
- As complexity continues to increase, need means of autonomous analysis

Commercial in Confidence UL-002358-PT



Typical SoC with network on chip

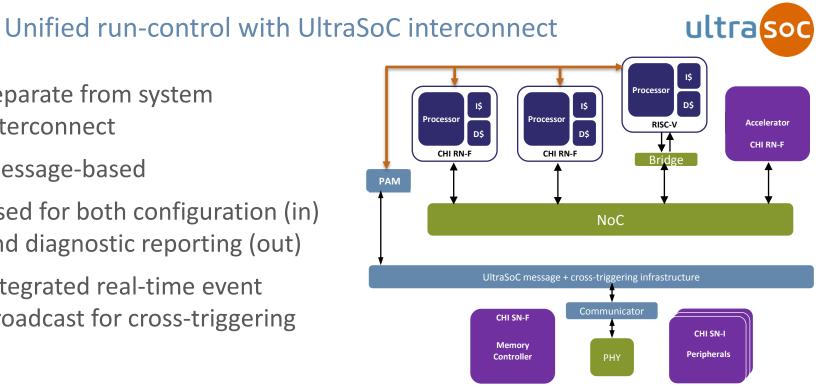
- Multiple processors with fully coherent caches
- I/O coherent accelerator
- Shared memory controller
- NoC could be ring, mesh or crossbar





13

- Separate from system interconnect
- Message-based
- Used for both configuration (in) and diagnostic reporting (out)
- Integrated real-time event broadcast for cross-triggering



Heterogeneous run-control

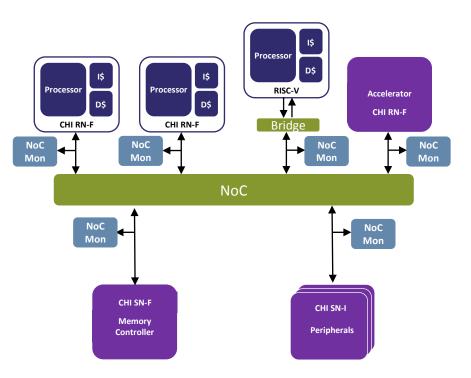


) • 🛛 🕼 🔌 🕪 🗉 🖷 🗱 🕉 🕾 📭 • 🖬 🗟 💉 🖉 • 🗛 • 🗛 • 🗛	↓ • ⊘ <i>∧</i> • ⊘ • ⊘ • ⊘ • ⊘ • ⊘ • ⊘ •				Quick Access
Debug ☆ 🎉 🕪 🖩 🛤 🔍 🕫 🕫 📭 マ 🗟 👘 マ 🖓 🖉 🖓	(x)= Variables 💁 Breakpoints 🔀 📑 Modules		- 8	해 Registers 😒 🔬 🛪	• • • • • • • • • • • • • • • • • • •
🚥 Platform Launch [Imperas - Connect to running simulator] 🗸 🔐 arm0 [Cortex-A9MPx1 arm]	x % & 3 x % D D % ~			Name	Value
ID #1 [arm0] Cortex-A9MPx1 arm (Suspended : Container)				1010 zero	0x0 (Hex)
Iv_vletter() at lv_draw_vbasic.c:359 0x12dae0				10101 ra	0x700001e4 (Hex)
■ 0x0				1010 sp	0x7000581c (Hex)
v 🔐 riscv0 [N25 riscv]				1010 gp	0x0 (Hex)
ID #2 [riscv0] N25 riscv (Suspended : Signal : :)				lili tp	0x0 (Hex)
main() at sender.c:84 0x70000200				1010 t0	0x70000020 (Hex)
ji mpd				1919 t1	0x7000580c (Hex)
					0x0 (Hex)
				1010 L2	0x7000587c (Hex)
				1010 SU	0x0 (Hex)
	<		>	1010 ST	0x50001000 (Hex)
crt0.S sender.c 🕱 🕼 lvgl_drv.c 🕼 display.c 🕼 lv_hal_disp.c 🕼 lv_vdb.c	c 💽 lv_task.c 🥅 qui.c 🔭 🗖 🗖		mers Vi 😒 😐 🗖	1010 al	0x70004564 (Hex)
crtu.s Sender.c 23 (E ivgi_drv.c) [E display.c] [E iv_nal_disp.c] [E iv_vdb.c				1010 a2	0x50001028 (Hex)
// Send Data	^	^ ÷		1010 a2	0x0 (Hex)
<pre>while(1) {</pre>		Group Value	Rbrk Wbrk	1010 a4	0x1 (Hex)
char number[64];		in Nam riscv0		1010 a4	0x1 (Hex)
		Type riscv		1010 a5	0x7f7f7f7f (Hex)
<pre>strcpy(data, ""); strcat(data, itoa(findex++, number, 10));</pre>		imp Varia N25		1010 a7	Oxffffffff (Hex)
<pre>strcat(data, itoa(findex++, number, 10)); strcat(data, ": ");</pre>		ing Instr 0		1010 s2	0x0 (Hex)
<pre>strcat(data, cookies[rand() % COOKIECOUNT]);</pre>		imp Mod		1010 s2	0x0 (Hex)
		imp Last		1010 s4	0x0 (Hex)
<pre>// step to the next frame</pre>		> MA gene		10101 S4	0x0 (Hex)
*LOCK = 1; while(*LOCK == 1) {};		> Millink I		1010 SG	0x0 (Hex)
while('LOCK == 1) {};		> Ma stack		1919 s7	0x0 (Hex)
}		> Min prog		1010 S7	0x0 (Hex)
		> 0101 prog		1010 SD	0x0 (Hex)
return 0;				1010 S9	0x0 (Hex)
}	~			1010 s11	0x0 (Hex)
4	>			1010 t3	0x66656463 (Hex)
			× • •	1010 t4	0x62613938 (Hex)
Debugger Console 🔀 📕 📮 🖛 🗖	🔄 Console 🔀 🧟 Tasks 🖹 Problems 🚺 E	recutables 🐉 Debug Source		1010 L4	0x37363534 (Hex)
latform Launch [Imperas - Connect to running simulator] mpd.exe (7.5) debug (criscv0) > (Ctcrl-C) No consoles to display at this time.			1010 to	0x33323130 (Hex)	
			- 1010 pc	0x70000200 (Hex)	
rning (MPD_NSR) RSP++ Command 'fn' (is simulation finished) is not support					0
ilation interrupted			1010 IU	0	
				1010 f2	0
70000200 in main () at source/application/ecookie-send/sender.c:84				1010 f3	0
<pre>while(*LOCK == 1) {};</pre>				1010 F4	0
ebug (riscv0) >				1010 F5	0





- UltraSoC NoC monitors on connections to the NoC
- Monitors are fully transaction aware



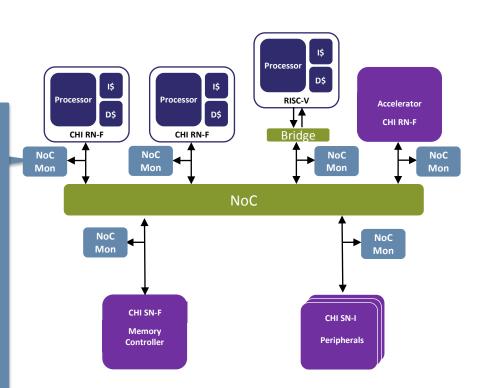


Automatically detect deadlock on the NoC

Trace all traffic into circular buffer within NoC monitor

Trigger trace if transaction duration exceeds threshold (e.g. 5k cycles)

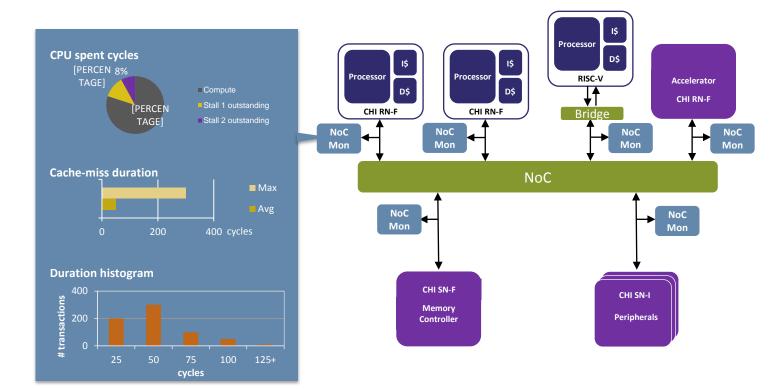
Stop tracing and output full details of deadlocked transaction and those immediately preceding it Nothing sent off-chip until deadlock occurs



ultrasoc

Example: "where have my MIPs gone?"



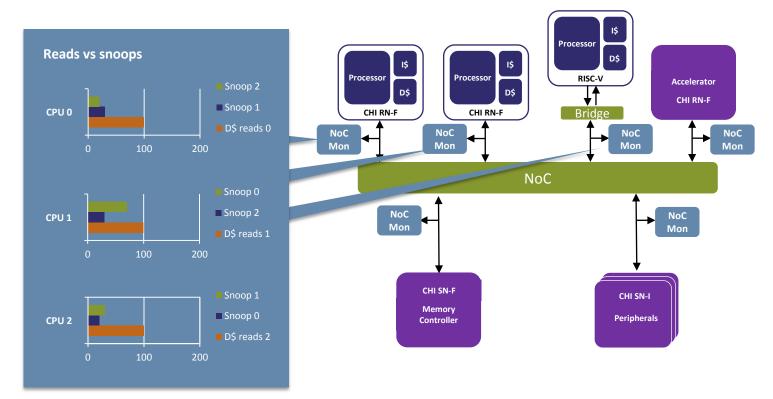


Commercial in Confidence

UL-002358-PT

Example: "how effectively is data shared?"





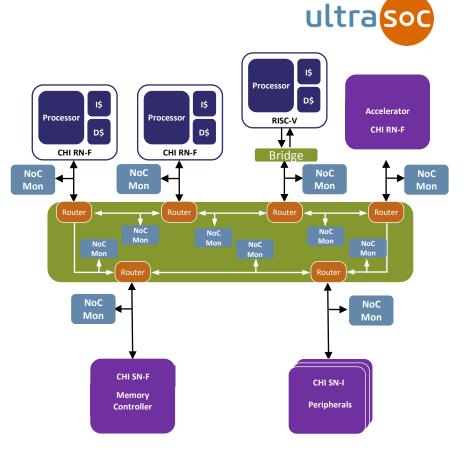
5 December 2018

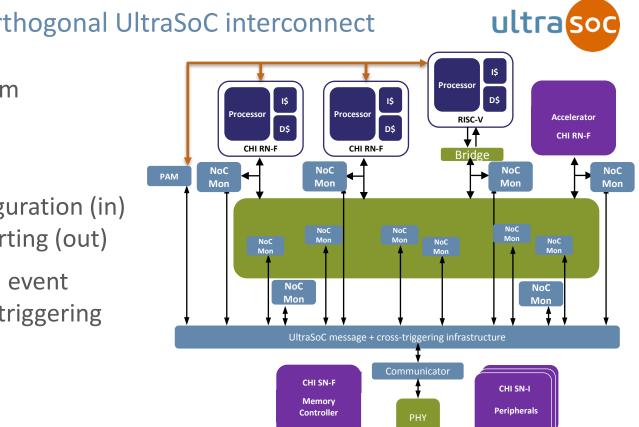
Commercial in Confidence UL-0

UL-002358-PT



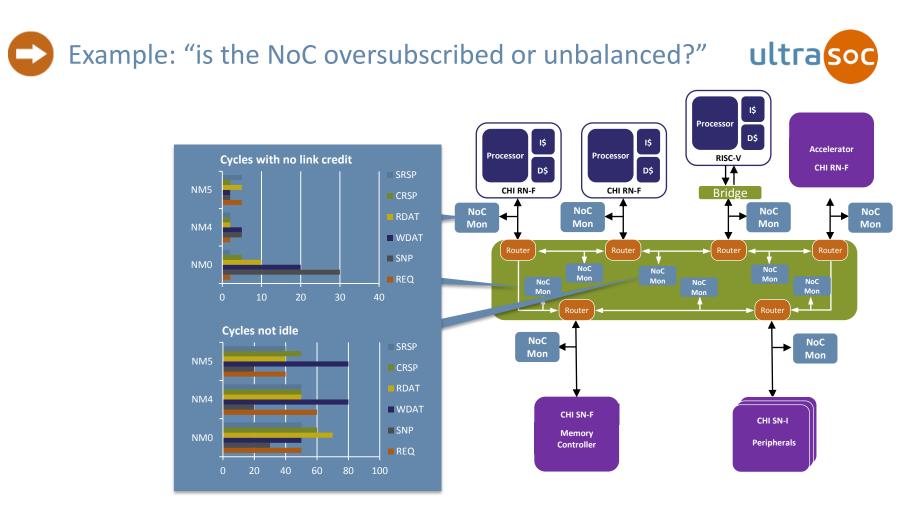
- Example shows ring topology
- Intra-router monitors are typically not transaction aware
 - Monitor individual channels only
 - Simpler

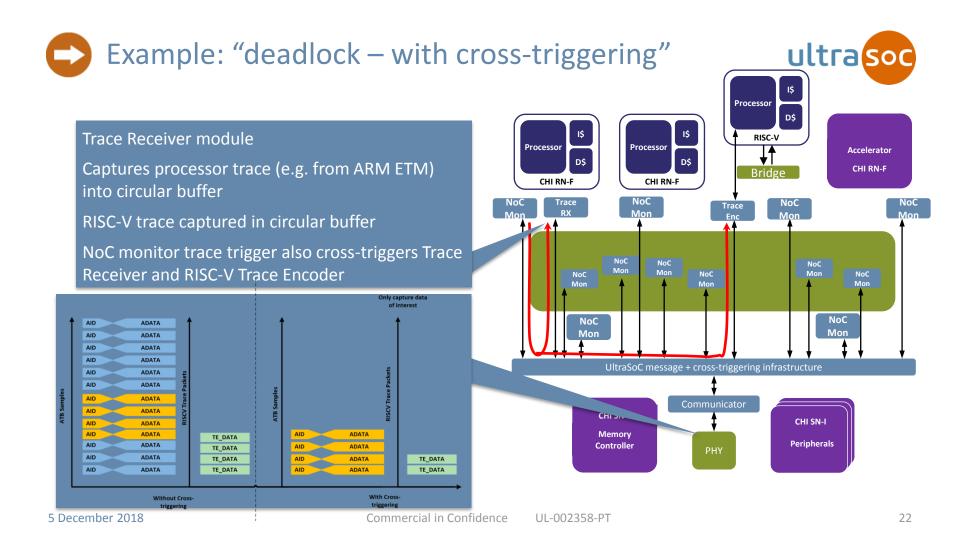




Independent, orthogonal UltraSoC interconnect

- Separate from system interconnect
- Message-based
- Used for both configuration (in) and diagnostic reporting (out)
- Integrated real-time event broadcast for cross-triggering



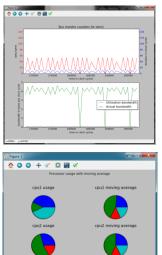


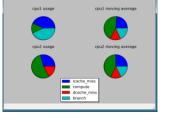




Eclipse based UltraDevelop IDE Single step & File Edit Navigate Search Project Run Window UltraDevelop Help 🔁 = 🗟 🍓 🕲 🛎 🖬 えつぶ 弓文 警台 回春 = 🔾 = 🔗 = 🚱 🔳 👘 = 🏷 = - - -Quick Access 🗄 😰 😂 😂 breakpoint E 😵 🕆 🗎 🗆 🕒 crt0.5 🔤 latch_address_... 🔤 *qu Project Explorer itor C... 🖾 🔉 Monitor M... 🔉 Downstre. Monitor Time View gdbinit auartz ui core 0.elf CPU code & xbm1:0 Module 0x70000010 jal \$ra, 0x12 ultilaunch.launch 119 x 11111 KSSEELEELAES444 4560409.. rt-agent.launch **RISC-V** 4570198 ../src/blocks.c xbm1: 20:int main(int argc. decoded trace 4610409 xbm2 /_rw_all.udt 4610409 vhm2 0x70000138 \$sp. CPU 4620198 vhm1 0x7000013C sw \$ra, 0x2c(\$sp) xhm21 30335 4660409 0x70000140 sw \$s0, 0x28(\$sp) elfielin Kernessunge 280000 30336 4660409... xbm2 0x70000144 addi \$s0, \$sp, 0x30 🍇 🐚 🔹 🖬 🖘 🖘 🗆 * Debug 30418 4670198 0x70000148 sw \$a0, 0xffffffdc(\$s0) xbm2: 0x7000014C sw \$a1, 0xffffffd8(\$s0 30687 4710409. 40040 v 😅 riscv-ptrace.start-agent (UltraDebug Agent) 4710409 2 Debug Memory Target 22: seed = 1; 30781 4720198 vhm1 🚚 UltraDebug Agent 31058 4760409 vhm2 I Target Communications 0x70000150 lui \$a5, 0x70000 0x70000154 addi \$a4, \$zero, 1 31059 t 4760409 vbm2 avi monitor por siscy-ptrace.riscy [UltraDebug Remote Target] 31147 4770198... xbm1 0x70000158 sw \$a4, 0x2b4(\$a5) avi monitor por 2,370,198,586 :45678901234567890123456 P Thread #1 (Running : User Request) 24: draw block(0, 0, 800, 480, 0x00000000); // cle 4,770,198,588 Denocd.exe Go Scroll Go to: C:/UltraSoC/demos/2018_03_15/riscv-tools/gdb.exe (7.12.50.2 0x7000015C addi \$a4, \$zero, 0 🗢 bus traffic.arm0 (UltraDebug Remote Target) itor Data 🐵 Traffic Generators emory 😳 Configuration 😳 PTrace N 🗉 🗸 SVirtual Console System Packe VC.Channel: vc1.0 Src ID Cha Sort By: Hierarchy 0.0 V fast N full address 20, 0.0006014573 0x70000004 21, 0.0004009715 **Multiple** 22 0.0002673144 full delta VVVNVVVNN addr 0y70000174 23, 0.0001782096 full delta 24, 0.0001188064 full delta VVVNVVVNN add 0x700001A8 25, 0.0000792043 full_delta YYYNYYYNN : 0x700001C0 other 26. 0.0000528028 full delta VVVNVVVNN 0x700001DC 27, 0.0000352019 0x0 full_delta YYYNYYYNN 0x700001F8 28. 0.0000234679 **CPUs** 0x0 full delta YYYNYYYNN -0x70000214 29, 0.0000156453 0x0 full_delta YYYNYYYNN : 0x70000230 **Real-time** 30, 0.0000104302 te=1 flows=0.1.2) 0x7000023C 0x0 full delta 31, 0.0000069535 Yar 0x0 full_addr_only addr 0x70000258 0x0 full_addr_only addr 0x7000026C **HW Data** 0x0 YYYNYYYNN addr full_delta **RISC-V** 0x0 full_delta Y addr SW & HW in full addr only Insert 1:1 instruction one tool packets

Script based

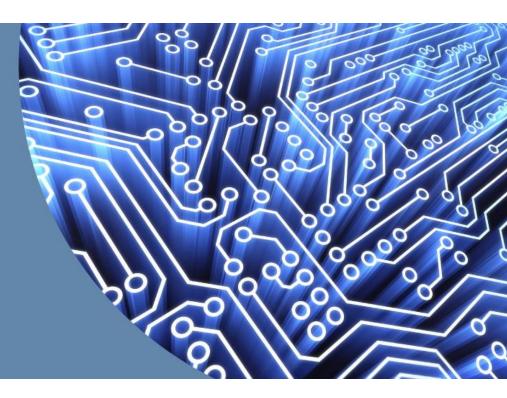




5 December 2018



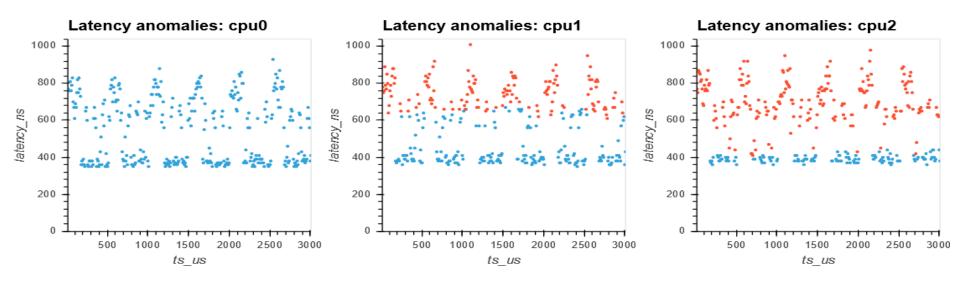
Problem Statement
Deterministic multi-core
Example scenarios
In-field analysis/ML
Summary







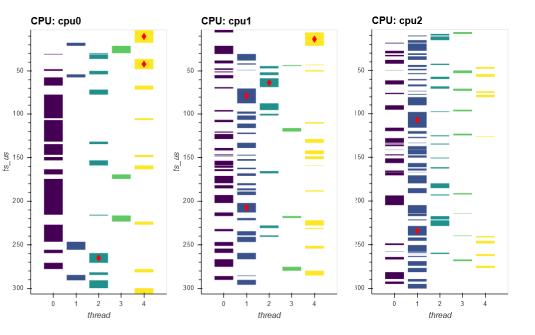
- Three CPU plots below show CPU cache-like traffic for 3 CPUs configured with different miss rates
- Excessive (anomalous) latencies are shown in red



26

Non-intrusive profiling with anomaly detection

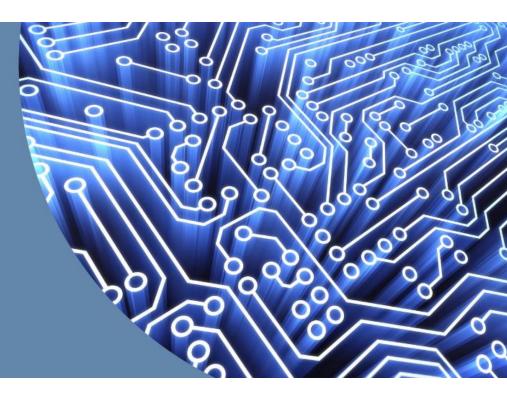
- Traditional profilers are inadequate:
 - Sampling = miss subtle or fast events (Nyquist)
 - Performance impact/intrusive
 - "Heisenbugs"
- UltraSoC is non-intrusive
- UltraSoC is wirespeed (100% coverage)
- Analytics and automated anomaly detection to make engineer more efficient







Problem Statement Deterministic multi-core Example scenarios In-field analysis/ML Summary







- The challenge today is systemic complexity
 - Architectural and modelling is needed but not enough
- Need tools that support true heterogenous systems
 - Both open source and commercial
- In addition to run-control, need non-intrusive monitoring
- More complex systems will require autonomous analytics and causality detection
- UltraSoC provides all or is working on all these