Finding the needle in the haystack

System software debugging at the right level of abstraction

1st International Workshop on

Multicore Application Debugging

(MAD 2013)

Achim Nohl, Synopsys Inc.

11/14/2013

System Integration

Example: Mobile hand-set, modem and sensor

Network of things
TCP over Network

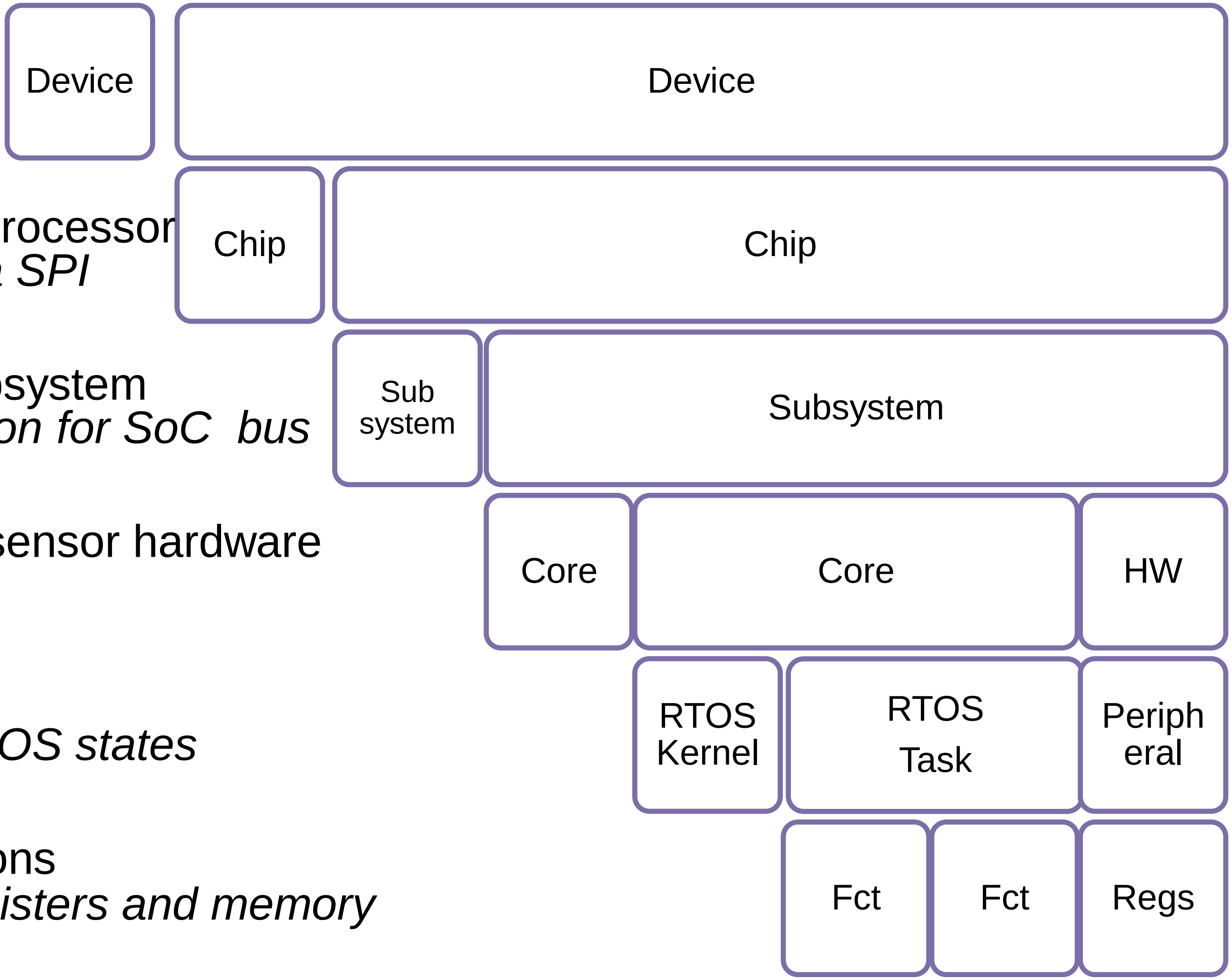
Modem-Application Processor
Hayes commands via SPI

CPU and Sensor Subsystem
Mailbox communication for SoC bus

Micro-controller and sensor hardware
Via sub-system bus

RTOS tasks
Via task APIs and RTOS states

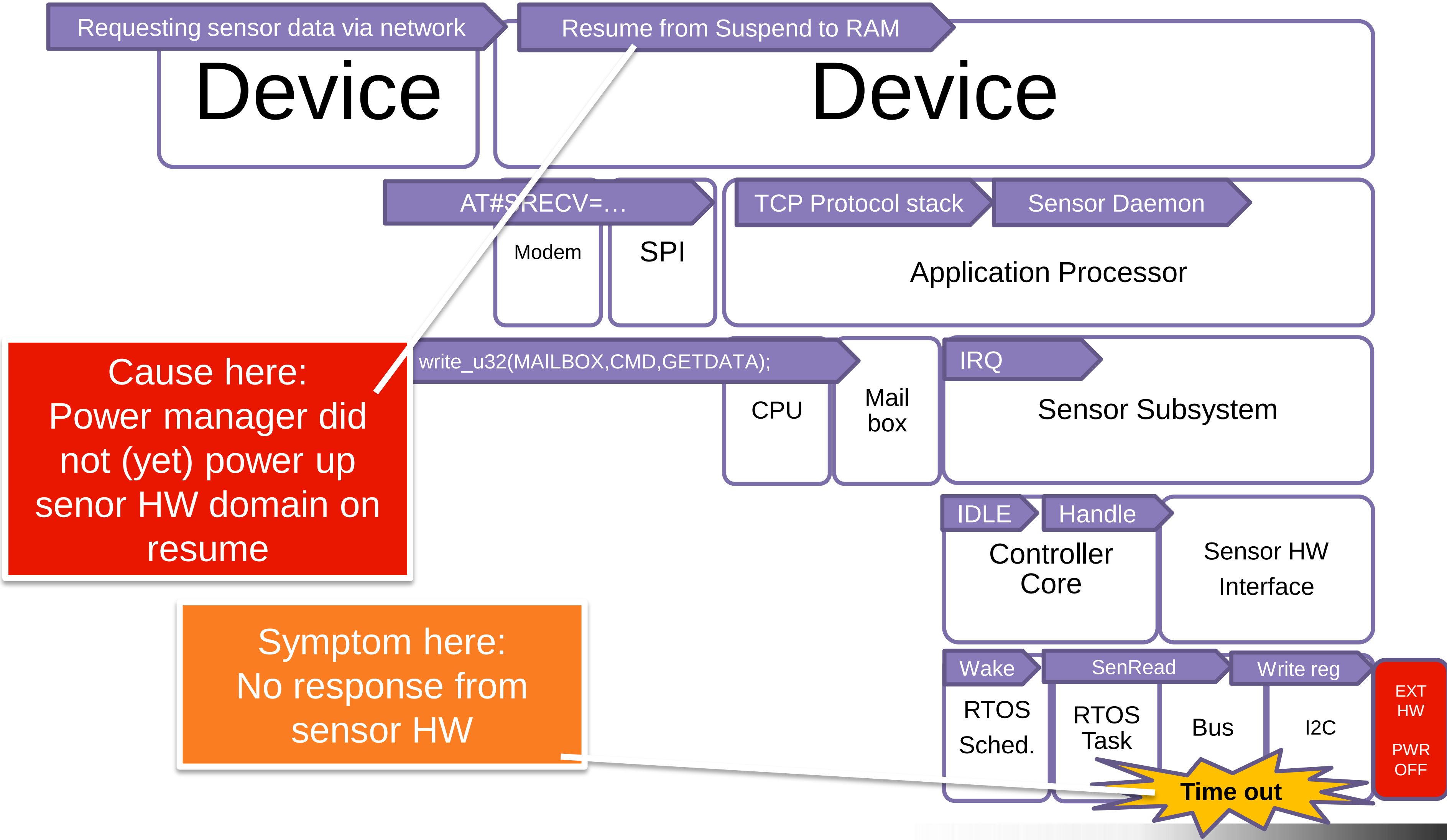
Sensor control functions
Via function calls, registers and memory



Device
Device
Chip
Chip
Sub system
Subsystem
Core
Core
HW
RTOS Kernel
RTOS Task
Peripheral
Fct
Fct
Regs

Top-down debugging

In an ideal world

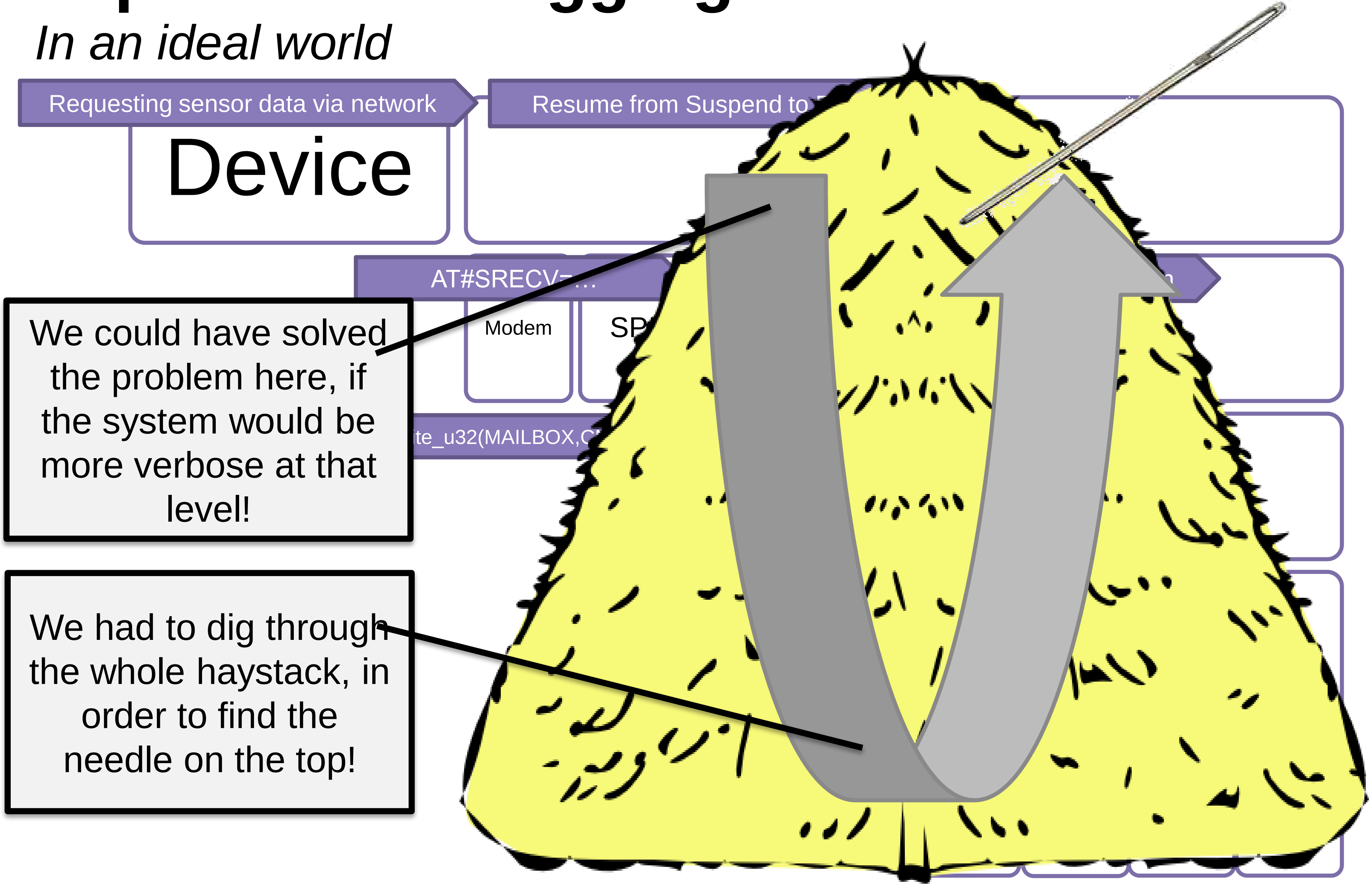


Cause here:
Power manager did not (yet) power up senor HW domain on resume

Symptom here:
No response from sensor HW

Top-down debugging

In an ideal world



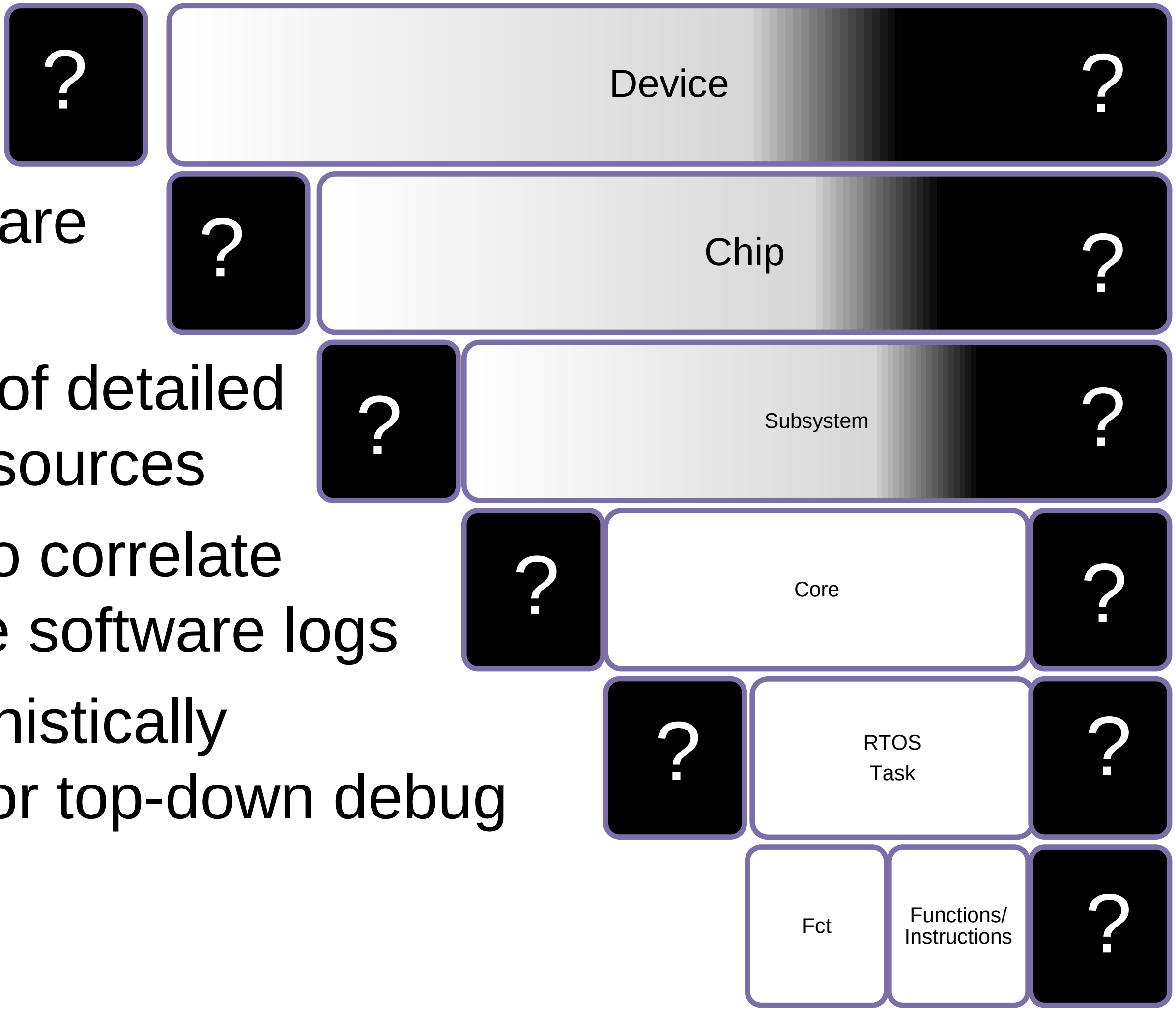
We could have solved the problem here, if the system would be more verbose at that level!

We had to dig through the whole haystack, in order to find the needle on the top!

Top-down debugging

Impractical in the real world

- No info about high level states
- Most components are black boxes
- If traces, then lots of detailed traces from many sources
- Often impossible to correlate traces and multiple software logs
- No way to deterministically repeat scenarios for top-down debug refinement



Top-down debugging

Wish list

Expose high level component state

- E.g. Domain powered off, sensor subsystem idle

Increase debugging scope to the maximum

- Under consideration of constraints such as IP protection etc.
- Whitebox debugging, where possible

Interface tracing with protocol-awareness

- TCP packets vs. Ethernet frames
- SMBus vs. I2C packets

Synchronized, concurrent debugging of SW on multi-cores

- Stop-mode debugging

Correlation of component traces and logs

Repeatable, deterministic debug scenarios

- For top-down iterations

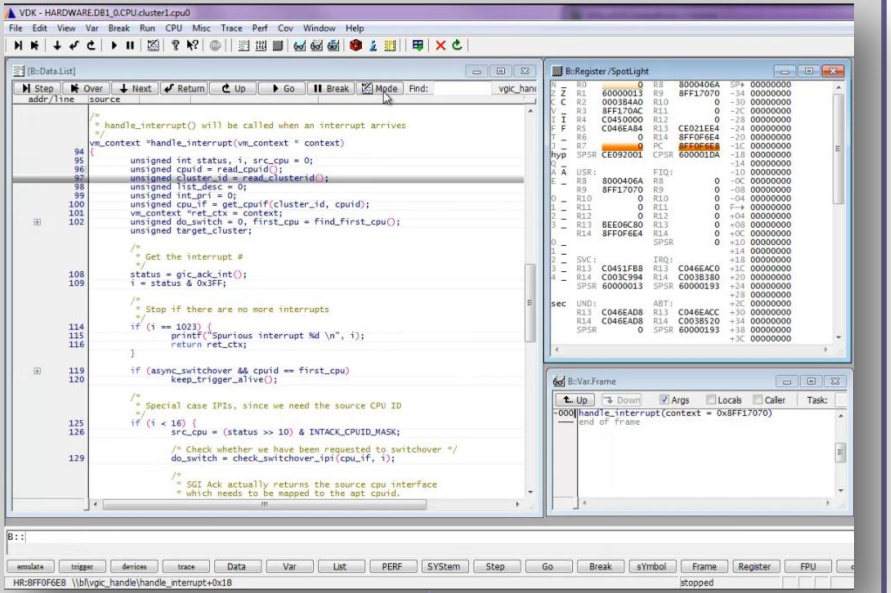
Addressing the debug challenger

Using Virtual Prototypes

Software Developer’s View

No change of habits required – It can be the same as with hardware

Standard 3rd party Software Debugger



VDK - HARDWARE.DB1_0.CPU.cluster1.cpu0
File Edit View Var Break Run CPU Misc Trace Perf Cov Window Help
[B::Data.List]
Step Over Next Return Up Go Break Mode Find: vgic_hand
addr/line source
B::Register /SpotLight
B::Var.Frame
Up Down ☑ Args □ Locals □ Caller Task:
emulate trigger devices trace Data Var List PERF SYStem Step Go Break sYmbol Frame Register FPU
stopped

```
/*
 * handle_interrupt() will be called when an interrupt arrives
 */
vm_context *handle_interrupt(vm_context * context)
{
        unsigned int status, i, src_cpu = 0;
        unsigned cpuid = read_cpuid();
        unsigned cluster_id = read_clusterid();
        unsigned list_desc = 0;
        unsigned int_pri = 0;
        unsigned cpu_if = get_cpuif(cluster_id, cpuid);
        vm_context *ret_ctx = context;
        unsigned do_switch = 0, first_cpu = find_first_cpu();
        unsigned target_cluster;

        /*
         * Get the interrupt #
         */
        status = gic_ack_int();
        i = status & 0x3FF;

        /*
         * Stop if there are no more interrupts
         */
        if (i == 1023) {
                printf("Spurious interrupt %d \n", i);
                return ret_ctx;
        }

        if (async_switchover && cpuid == first_cpu)
                keep_trigger_alive();

        /*
         * Special case IPIs, since we need the source CPU ID
         */
        if (i < 16) {
                src_cpu = (status >> 10) & INTACK_CPUID_MASK;

                /* Check whether we have been requested to switchover */
                do_switch = check_switchover_ipi(cpu_if, i);

                /*
                 * SGI Ack actually returns the source cpu interface
                 * which needs to be mapped to the apt cpuid.
```

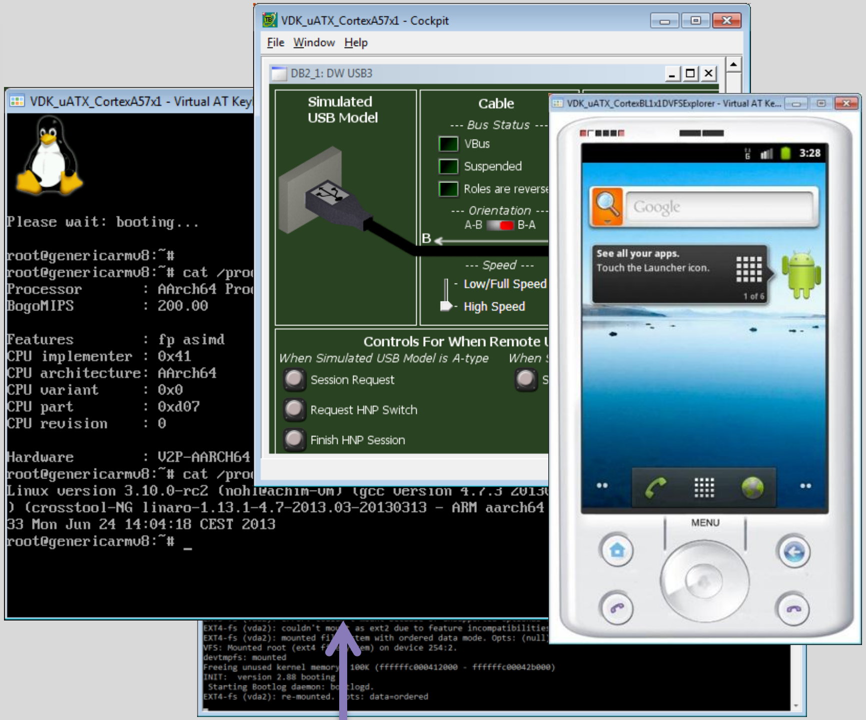
```
N  _  R0          0  R8   8000406A  SP>  00000000
Z  Z  R1   60000013  R9   8FF17070  -34  00000000
C  C  R2   0003B4A0  R10         0  -30  00000000
V  _  R3   8FF170AC  R11         0  -2C  00000000
I  I  R4   C0450000  R12         0  -28  00000000
F  F  R5   C046EA84  R13  CE021EE4  -24  00000000
T  _  R6          0  R14  8FF0F6E4  -20  00000000
J  _  R7          0  PC   8FF0F6E8  -1C  00000000
hyp   SPSR CE092001  CPSR 600001DA  -18  00000000
Q  _                                -14  00000000
A  A  USR:           FIQ:           -10  00000000
E  _  R8   8000406A  R8          0  -0C  00000000
      R9   8FF17070  R9          0  -08  00000000
0  _  R10         0  R10         0  -04  00000000
1  _  R11         0  R11         0  F->  00000000
2  _  R12         0  R12         0  +04  00000000
3  _  R13  BEE06C80  R13         0  +08  00000000
      R14  8FF0F6E4  R14         0  +0C  00000000
0  _                 SPSR        0  +10  00000000
1  _                                +14  00000000
2  _  SVC:           IRQ:           +18  00000000
3  _  R13  C0451FB8  R13  C046EAC0  +1C  00000000
4  _  R14  C003C994  R14  C003B380  +20  00000000
      SPSR 60000013  SPSR 60000193  +24  00000000
                                    +28  00000000
sec   UND:           ABT:           +2C  00000000
      R13  C046EAD8  R13  C046EACC  +30  00000000
      R14  C046EAD8  R14  C003B520  +34  00000000
      SPSR        0  SPSR 60000193  +38  00000000
                                    +3C  00000000
```

```
-000 handle_interrupt(context = 0x8FF17070)
---- end of frame
```

```
B::
```

```
HR:8FF0F6E8  \\bl\vgic_handle\handle_interrupt+0x18
```

Virtual and physical IO



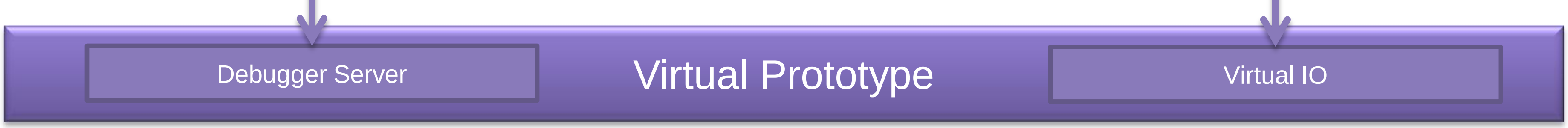
```
Please wait: booting...

root@genericarmv8:~#
root@genericarmv8:~# cat /proc
Processor       : AArch64 Proc
BogoMIPS        : 200.00

Features        : fp asimd
CPU implementer : 0x41
CPU architecture: AArch64
CPU variant     : 0x0
CPU part        : 0xd07
CPU revision    : 0

Hardware        : V2P-AARCH64
root@genericarmv8:~# cat /proc
Linux version 3.10.0-rc2 (nohl@achim-vm) (gcc version 4.7.3 2013
) (crosstool-NG linaro-1.13.1-4.7-2013.03-20130313 - ARM aarch64
33 Mon Jun 24 14:04:18 CEST 2013
root@genericarmv8:~# _
```

```
EXT4-fs (vda2): couldn't mount as ext2 due to feature incompatibilities
EXT4-fs (vda2): mounted filesystem with ordered data mode. Opts: (null)
VFS: Mounted root (ext4 filesystem) on device 254:2.
devtmpfs: mounted
Freeing unused kernel memory: 100K (ffffffc000412000 - ffffffc00042b000)
INIT:  version 2.88 booting
 Starting Bootlog daemon: bootlogd.
EXT4-fs (vda2): re-mounted. Opts: data=ordered
```



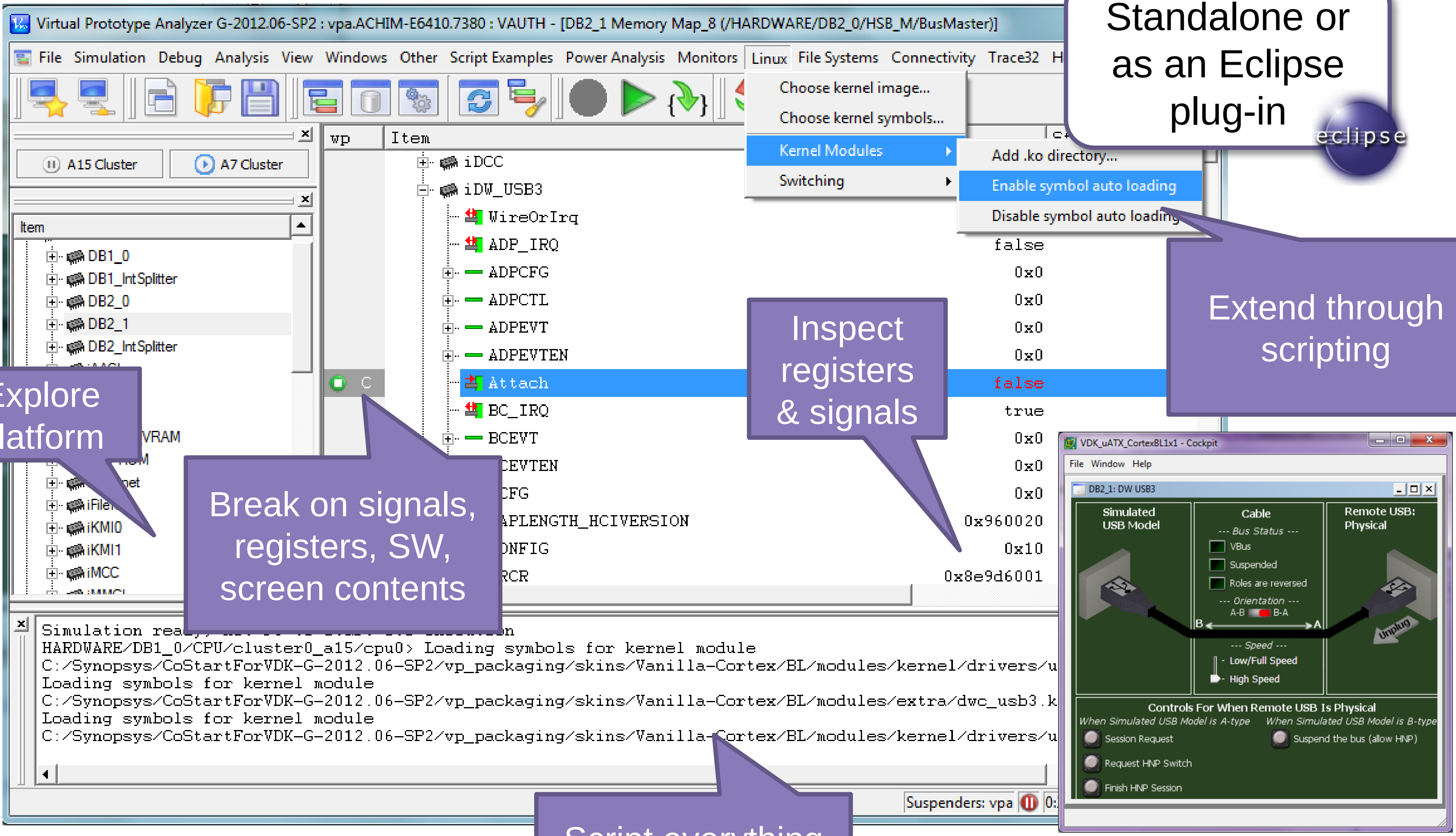
Debugger Server

Virtual Prototype

Virtual IO

System level software debugging

Visibility beyond traditional software debugging



Virtual Prototype Analyzer G-2012.06-SP2 : vpa.ACHIM-E6410.7380 : VAUTH - [DB2_1 Memory Map_8 (/HARDWARE/DB2_0/HSB_M/BusMaster)]
File Simulation Debug Analysis View Windows Other Script Examples Power Analysis Monitors Linux File Systems Connectivity Trace32
Choose kernel image...
Choose kernel symbols...
Kernel Modules
Switching
Add .ko directory...
Enable symbol auto loading
Disable symbol auto loading
A15 Cluster
A7 Cluster
Item
DB1_0
DB1_IntSplitter
DB2_0
DB2_1
DB2_IntSplitter
VRAM
iFile
iKMI0
iKMI1
iMCC
wp
VDK_uATX_CortexBL1x1 - Cockpit
File Window Help
DB2_1: DW USB3
Simulated USB Model
Cable
--- Bus Status ---
VBus
Suspended
Roles are reversed
--- Orientation ---
A-B B-A
--- Speed ---
Low/Full Speed
High Speed
Remote USB: Physical
Unplug
Controls For When Remote USB Is Physical
When Simulated USB Model is A-type
When Simulated USB Model is B-type
Session Request
Request HNP Switch
Finish HNP Session
Suspend the bus (allow HNP)
Suspenders: vpa

```
Item
iDCC
iDW_USB3
  WireOrIrq
  ADP_IRQ                         false
  ADPCFG                          0x0
  ADPCTL                          0x0
  ADPEVT                          0x0
  ADPEVTEN                        0x0
  Attach                          false
  BC_IRQ                          true
  BCEVT                           0x0
  CEVTEN                          0x0
  CFG                             0x0
  APLENGTH_HCIVERSION             0x960020
  ONFIG                           0x10
  RCR                             0x8e9d6001
```

```
Simulation rea                                    n
HARDWARE/DB1_0/CPU/cluster0_a15/cpu0> Loading symbols for kernel module
C:/Synopsys/CoStartForVDK-G-2012.06-SP2/vp_packaging/skins/Vanilla-Cortex/BL/modules/kernel/drivers/u
Loading symbols for kernel module
C:/Synopsys/CoStartForVDK-G-2012.06-SP2/vp_packaging/skins/Vanilla-Cortex/BL/modules/extra/dwc_usb3.k
Loading symbols for kernel module
C:/Synopsys/CoStartForVDK-G-2012.06-SP2/vp_packaging/skins/Vanilla-Cortex/BL/modules/kernel/drivers/u
```

Standalone or as an Eclipse plug-in

Extend through scripting

Inspect registers & signals

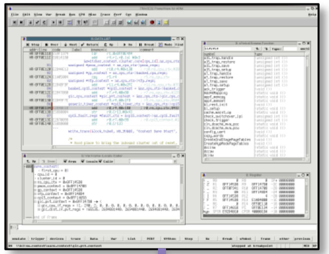
Explore platform

Break on signals, registers, SW, screen contents

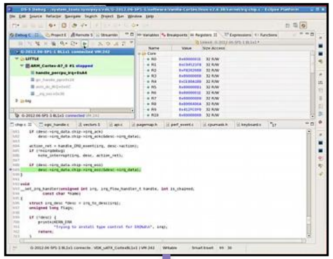
Script everything

Using a VDK

Lauterbach

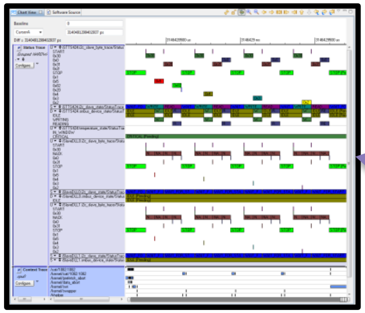


ARM DS-5/RVDS

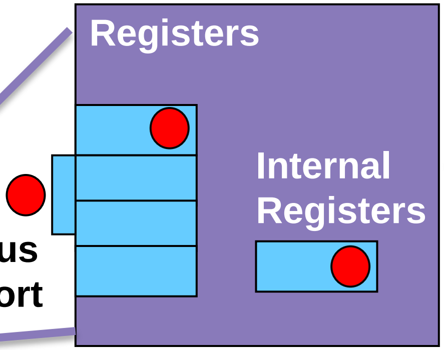


System Debug

Tracing



Full HW visibility



Registers

Bus Port

Internal Registers

Non-intrusive, stop mode debugging

Virtualizer Multi-core Debugger Server

Virtualizer Developer Kit (VDK)

*Simulation of a hardware device/SoC**

ARM Cortex A

ARM Cortex R

ARM Cortex M

USB

MMC

I2C/SPI

GMAC

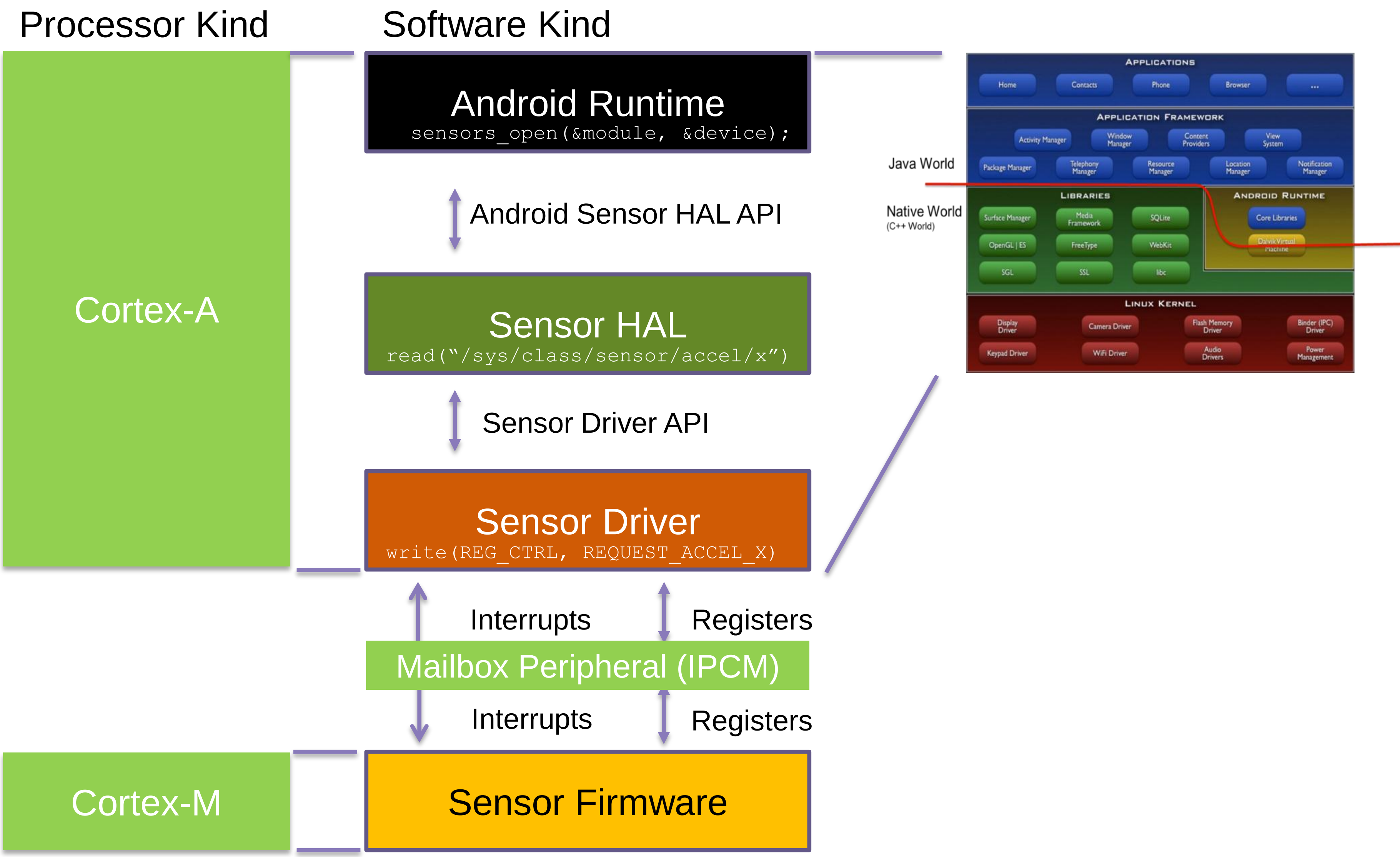
LCD

Unmodified binary SW stack images
E.g. bootROM + Linux kernel + Linux filesystem

Script everything

*Simulation using industry standard based Transaction Level Models (TLM)

Sensor HW/SW Integration



Processor Kind

Software Kind

Cortex-A

Android Runtime

`sensors_open(&module, &device);`

Android Sensor HAL API

Sensor HAL

`read("/sys/class/sensor/accel/x")`

Sensor Driver API

Sensor Driver

`write(REG_CTRL, REQUEST_ACCEL_X)`

Interrupts

Registers

Mailbox Peripheral (IPCM)

Interrupts

Registers

Cortex-M

Sensor Firmware

SYNOPSYS® Accelerating Innovation

Multi-Core Software Development

Cortex-A Linux Device Driver – Code Excerpt

```
printk("-- Set IPCM PL320 Mailbox source register.\n");
IPCM_WRITE(IPCM0SOURCE,(1<<0));
printk("-- Set IPCM PL320 Mailbox interrupt mask register.\n");
IPCM_WRITE(IPCM0MSET, 0x3);
printk("-- Set IPCM PL320 Mailbox destination register.\n");
IPCM_WRITE(IPCM0DSET  ,(1<<2));
printk("-- Set IPCM PL320 Mailbox data 0 register to %.8x.\n",0);
IPCM_WRITE(IPCM0DR0,0);
printk("-- Set IPCM PL320 Mailbox data 1 register to %.8x.\n",sensor);
IPCM_WRITE(IPCM0DR1,sensor);

printk("-- Send IPCM PL320 Mailbox message.\n");
IPCM_WRITE(IPCM0SEND  ,(1<<0));
printk("-- Waiting for acknowledge\n");
m4sensor_device_control.ack=false;
interruptible_sleep_on_timeout(&m4sensor_queue, 1000 /* jiffies */);
if(m4sensor_device_control.ack==0) {
  printk("-- Error: Timeout while waiting for acknowledge\n");
  return 0;
}

printk("-- Acknowledge received. Command completed. Data ready.
value= IPCM_READ(IPCM0DR2);
printk("-- Reading sensor data from mailbox: %d\n",value);
```

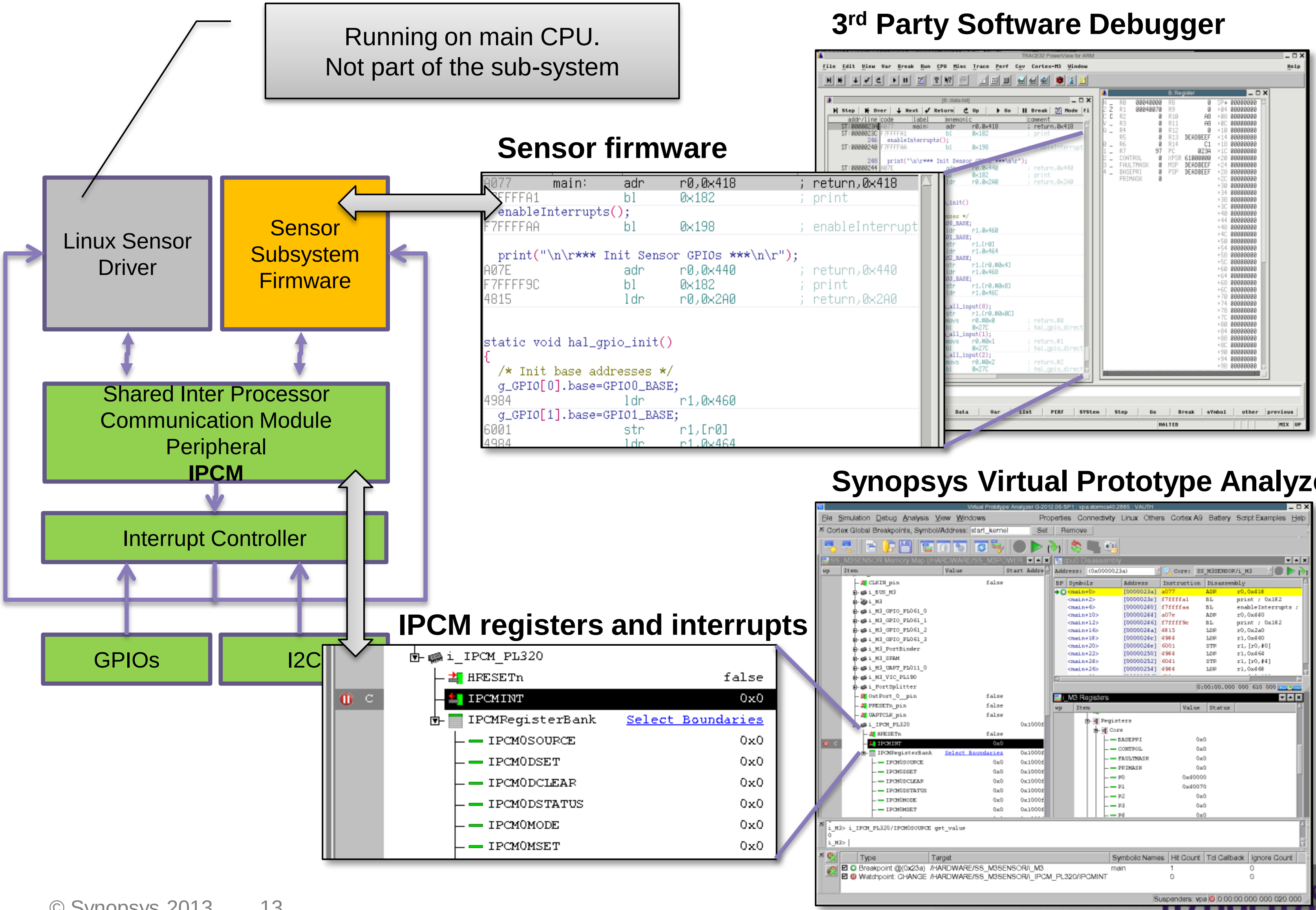
Cortex-A Linux Kernel debug messages

Cortex-M Firmware Debug messages

Cortex-M Firmware Code Excerpt

```
  print("  Handle IPCM interrupt\n\r");
  print("  Reading IPCM mailbox message\n\r");
  task=IPCM_READ(IPCM0DR0);
  switch(task){
  case 0:
    print("  Reading sensor from GPIO\n\r");
    sensor=IPCM_READ(IPCM0DR1);
    value=hal_gpio_get_value_word8(sensor);
    print("  Writing to mailbox\n\r");
    IPCM_WRITE(IPCM0DR2,value);
    break;
   ….
}
print("  Clearing ipcm interrupt, acknowledge\n\r");
IPCM_WRITE(IPCM0SEND  ,(1<<1));
```

Subsystem Firmware Debugging


Running on main CPU.
Not part of the sub-system
Linux Sensor Driver
Sensor Subsystem Firmware
Shared Inter Processor Communication Module Peripheral
IPCM
Interrupt Controller
GPIOs
I2C
3rd Party Software Debugger
Sensor firmware
IPCM registers and interrupts
Synopsys Virtual Prototype Analyzer
File Edit View Var Break Run CPU Misc Trace Perf Cov Cortex-M3 Window Help
File Simulation Debug Analysis View Windows Properties Connectivity Linux Others Cortex A9 Battery Script Examples Help
Cortex Global Breakpoints, Symbol/Address: start_kernel Set Remove

```
A077        main:    adr    r0,0x418        ; return,0x418
FFFFA1               bl     0x182           ; print
 enableInterrupts();
F7FFFFAA             bl     0x198           ; enableInterrupt

 print("\n\r*** Init Sensor GPIOs ***\n\r");
A07E                 adr    r0,0x440        ; return,0x440
F7FFFF9C             bl     0x182           ; print
4815                 ldr    r0,0x2A0        ; return,0x2A0

static void hal_gpio_init()
{
 /* Init base addresses */
 g_GPIO[0].base=GPIO0_BASE;
4984                 ldr    r1,0x460
 g_GPIO[1].base=GPIO1_BASE;
6001                 str    r1,[r0]
4984                 ldr    r1,0x464
```

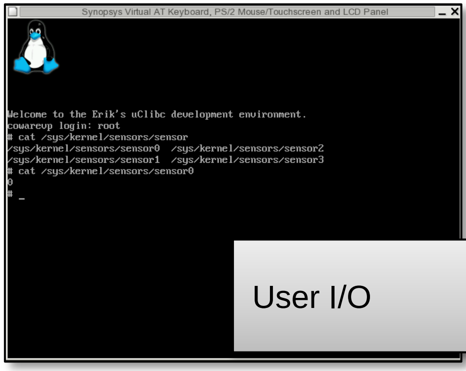
```
i_IPCM_PL320
  HRESETn                                false
  IPCMINT                                0x0
  IPCMRegisterBank     Select Boundaries
    IPCM0SOURCE                          0x0
    IPCM0DSET                            0x0
    IPCM0DCLEAR                          0x0
    IPCM0DSTATUS                         0x0
    IPCM0MODE                            0x0
    IPCM0MSET                            0x0
```

```
i_M3> i_IPCM_PL320/IPCM0SOURCE get_value
0
i_M3>
```

Type	Target	Symbolic Names	Hit Count	Tcl Callback	Ignore Count
Breakpoint @(0x23a)	/HARDWARE/SS_M3SENSOR/i_M3	main	1		0
Watchpoint: CHANGE	/HARDWARE/SS_M3SENSOR/i_IPCM_PL320/IPCMINT		0		0

MultiCore Logging & Tracing

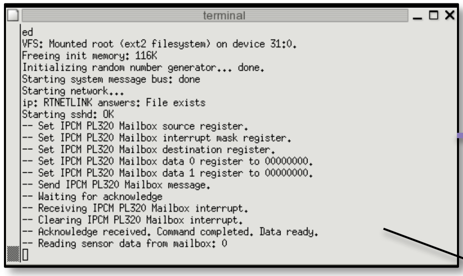
Linux VGA Console


Synopsys Virtual AT Keyboard, PS/2 Mouse/Touchscreen and LCD Panel

```
Welcome to the Erik's uClibc development environment.
cowarevp login: root
# cat /sys/kernel/sensors/sensor
/sys/kernel/sensors/sensor0  /sys/kernel/sensors/sensor2
/sys/kernel/sensors/sensor1  /sys/kernel/sensors/sensor3
# cat /sys/kernel/sensors/sensor0
0
# _
```

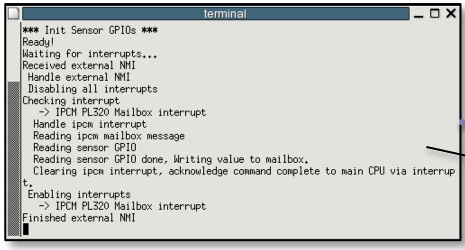
User I/O

Linux UART Console



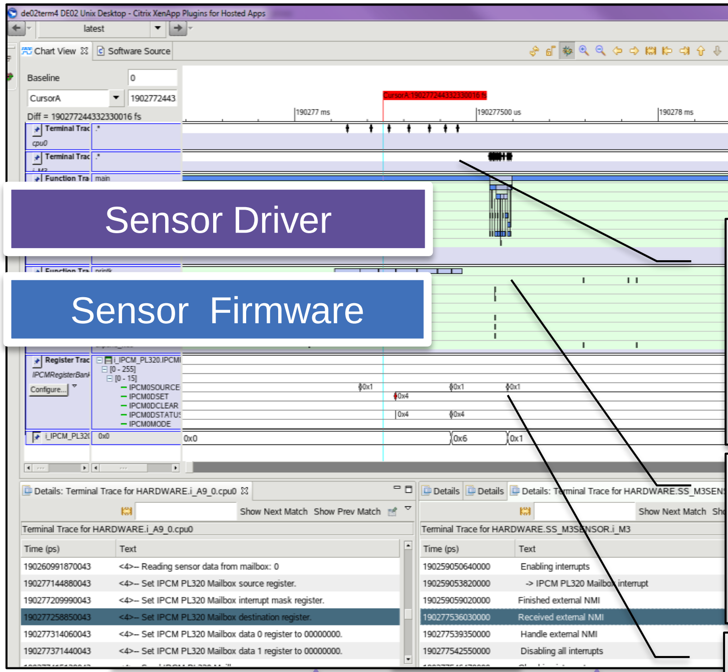
```
ed
VFS: Mounted root (ext2 filesystem) on device 31:0.
Freeing init memory: 116K
Initializing random number generator... done.
Starting system message bus: done
Starting network...
ip: RTNETLINK answers: File exists
Starting sshd: OK
-- Set IPCM PL320 Mailbox source register.
-- Set IPCM PL320 Mailbox interrupt mask register.
-- Set IPCM PL320 Mailbox destination register.
-- Set IPCM PL320 Mailbox data 0 register to 00000000.
-- Set IPCM PL320 Mailbox data 1 register to 00000000.
-- Send IPCM PL320 Mailbox message.
-- Waiting for acknowledge
-- Receiving IPCM PL320 Mailbox interrupt.
-- Clearing IPCM PL320 Mailbox interrupt.
-- Acknowledge received. Command completed. Data ready.
-- Reading sensor data from mailbox: 0
```

Sensor UART



```
*** Init Sensor GPIOs ***
Ready!
Waiting for interrupts...
Received external NMI
 Handle external NMI
 Disabling all interrupts
Checking interrupt
   -> IPCM PL320 Mailbox interrupt
  Handle ipcm interrupt
  Reading ipcm mailbox message
  Reading sensor GPIO
  Reading sensor GPIO done, Writing value to mailbox.
  Clearing ipcm interrupt, acknowledge command complete to main CPU via interrup
t.
 Enabling interrupts
   -> IPCM PL320 Mailbox interrupt
Finished external NMI
```

Virtual Prototype Tracing



Sensor Driver

Sensor Firmware

Details: Terminal Trace for HARDWARE.i_A9_0.cpu0

Terminal Trace for HARDWARE.i_A9_0.cpu0

Time (ps)	Text
190260991870043	<4>-- Reading sensor data from mailbox: 0
190277144880043	<4>-- Set IPCM PL320 Mailbox source register.
190277209990043	<4>-- Set IPCM PL320 Mailbox interrupt mask register.
190277258850043	<4>-- Set IPCM PL320 Mailbox destination register.
190277314060043	<4>-- Set IPCM PL320 Mailbox data 0 register to 00000000.
190277371440043	<4>-- Set IPCM PL320 Mailbox data 1 register to 00000000.

Details: Terminal Trace for HARDWARE.SS_M3SEN

Terminal Trace for HARDWARE.SS_M3SENSOR.i_M3

Time (ps)	Text
190259050640000	Enabling interrupts
190259053820000	-> IPCM PL320 Mailbox interrupt
190259059020000	Finished external NMI
190277536030000	Received external NMI
190277539350000	Handle external NMI
190277542550000	Disabling all interrupts

Unified and synchronized logging of debug messages from multiple sources (UARTs, Android Logcat etc.)

Synchronized software traces for Cortex-A and Cortex-M

Peripheral access and signal traces. Here: shared IPCM

Cortex-A Linux Kernel debug messages

Cortex-M Firmware Debug messages

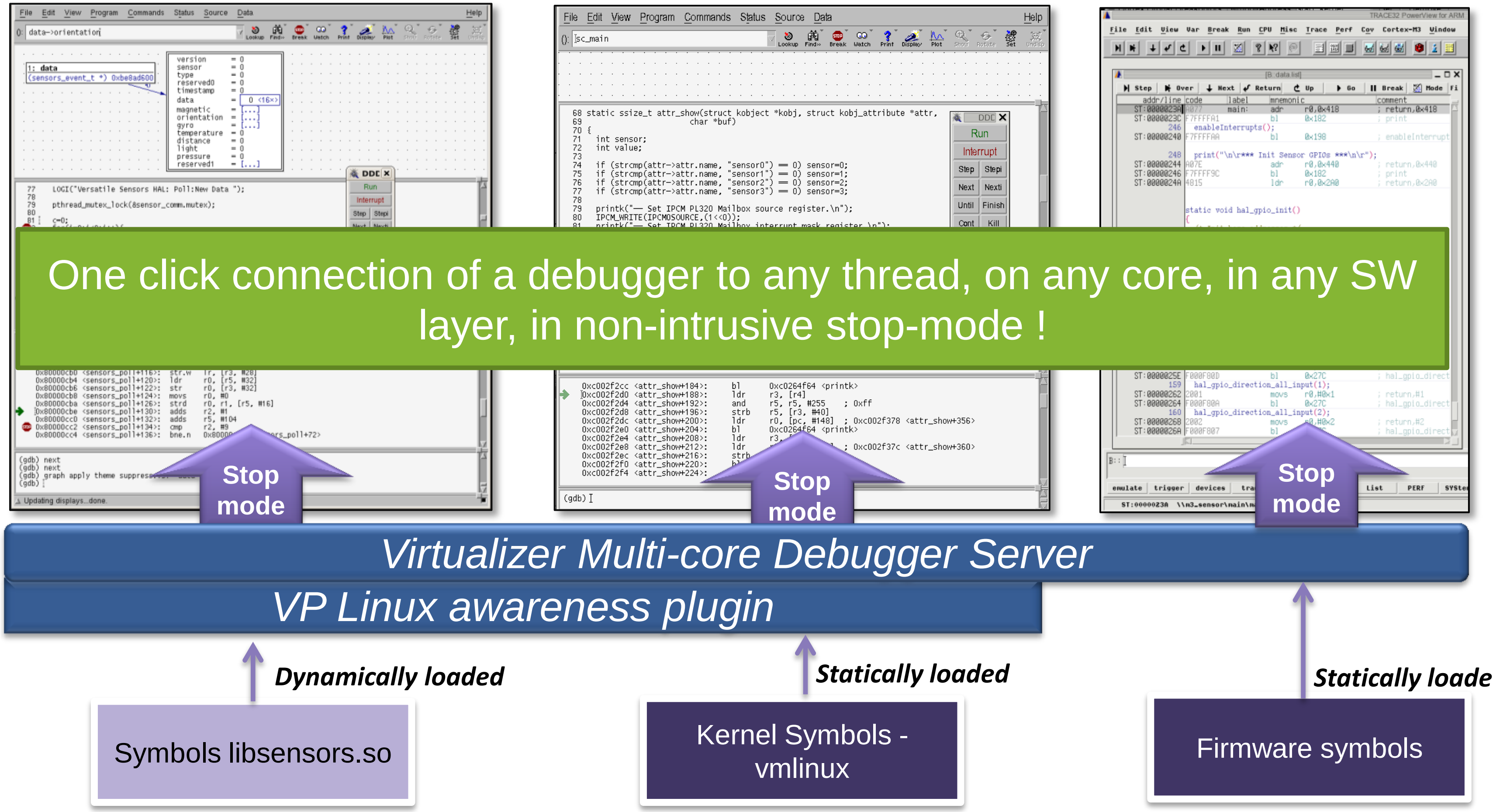
Vertical HW/SW integration: Middleware & Drivers

Multi-layer, multi-core, multi-tool SW debugging

GDB:
Sensor middleware – Cortex A

GDB:
Sensor driver – Cortex A

Lauterbach:
Sensor firmware– Cortex M



```
(): data->orientation

1: data
(sensors_event_t *) 0xbe8ad600

version      = 0
sensor       = 0
type         = 0
reserved0    = 0
timestamp    = 0
data         = 0 <16×>
magnetic     = [...]
orientation  = [...]
gyro         = [...]
temperature  = 0
distance     = 0
light        = 0
pressure     = 0
reserved1    = [...]

    LOGI("Versatile Sensors HAL: Poll:New Data ");

    pthread_mutex_lock(&sensor_comm.mutex);

    c=0;

0x80000cb0 <sensors_poll+116>:  str.w   lr, [r3, #28]
0x80000cb4 <sensors_poll+120>:  ldr     r0, [r5, #32]
0x80000cb6 <sensors_poll+122>:  str     r0, [r3, #32]
0x80000cb8 <sensors_poll+124>:  movs    r0, #0
0x80000cba <sensors_poll+126>:  strd    r0, r1, [r5, #16]
0x80000cbe <sensors_poll+130>:  adds    r2, #1
0x80000cc0 <sensors_poll+132>:  adds    r5, #104
0x80000cc2 <sensors_poll+134>:  cmp     r2, #9
0x80000cc4 <sensors_poll+136>:  bne.n   0x80000    rs_poll+72>

(gdb) next
(gdb) next
(gdb) graph apply theme suppress
(gdb)
Updating displays...done.
```

```
(): sc_main

static ssize_t attr_show(struct kobject *kobj, struct kobj_attribute *attr,
                         char *buf)
{
  int sensor;
  int value;

  if (strcmp(attr->attr.name, "sensor0") == 0) sensor=0;
  if (strcmp(attr->attr.name, "sensor1") == 0) sensor=1;
  if (strcmp(attr->attr.name, "sensor2") == 0) sensor=2;
  if (strcmp(attr->attr.name, "sensor3") == 0) sensor=3;

  printk("-- Set IPCM PL320 Mailbox source register.\n");
  IPCM_WRITE(IPCMOSOURCE,(1<<0));
  printk("-- Set IPCM PL320 Mailbox interrupt mask register.\n");

0xc002f2cc <attr_show+184>:    bl      0xc0264f64 <printk>
0xc002f2d0 <attr_show+188>:    ldr     r3, [r4]
0xc002f2d4 <attr_show+192>:    and     r5, r5, #255    ; 0xff
0xc002f2d8 <attr_show+196>:    strb    r5, [r3, #40]
0xc002f2dc <attr_show+200>:    ldr     r0, [pc, #148]  ; 0xc002f378 <attr_show+356>
0xc002f2e0 <attr_show+204>:    bl      0xc0264f64 <printk>
0xc002f2e4 <attr_show+208>:    ldr     r3,
0xc002f2e8 <attr_show+212>:    ldr                     ; 0xc002f37c <attr_show+360>
0xc002f2ec <attr_show+216>:    strb
0xc002f2f0 <attr_show+220>:    bl
0xc002f2f4 <attr_show+224>:

(gdb)
```

```
TRACE32 PowerView for ARM
[B::data.list]

     addr/line code      label   mnemonic              comment
  ST:0000023A A077       main:   adr      r0,0x418     ; return,0x418
  ST:0000023C F7FFFFA1           bl       0x182        ; print
          246   enableInterrupts();
  ST:00000240 F7FFFFAA           bl       0x198        ; enableInterrupt

          248   print("\n\r*** Init Sensor GPIOs ***\n\r");
  ST:00000244 A07E               adr      r0,0x440     ; return,0x440
  ST:00000246 F7FFFF9C           bl       0x182        ; print
  ST:0000024A 4815               ldr      r0,0x2A0     ; return,0x2A0

static void hal_gpio_init()
{

  ST:0000025E F000F80D           bl       0x27C        ; hal_gpio_direct
          159   hal_gpio_direction_all_input(1);
  ST:00000262 2001               movs     r0,#0x1      ; return,#1
  ST:00000264 F000F80A           bl       0x27C        ; hal_gpio_direct
          160   hal_gpio_direction_all_input(2);
  ST:00000268 2002               movs     r0,#0x2      ; return,#2
  ST:0000026A F000F807           bl                    ; hal_gpio_direct

B::
ST:0000023A  \\m3_sensor\main\m
```

One click connection of a debugger to any thread, on any core, in any SW layer, in non-intrusive stop-mode !

Virtualizer Multi-core Debugger Server

VP Linux awareness plugin

Dynamically loaded

Statically loaded

Statically loaded

Symbols libsensors.so

Kernel Symbols - vmlinux

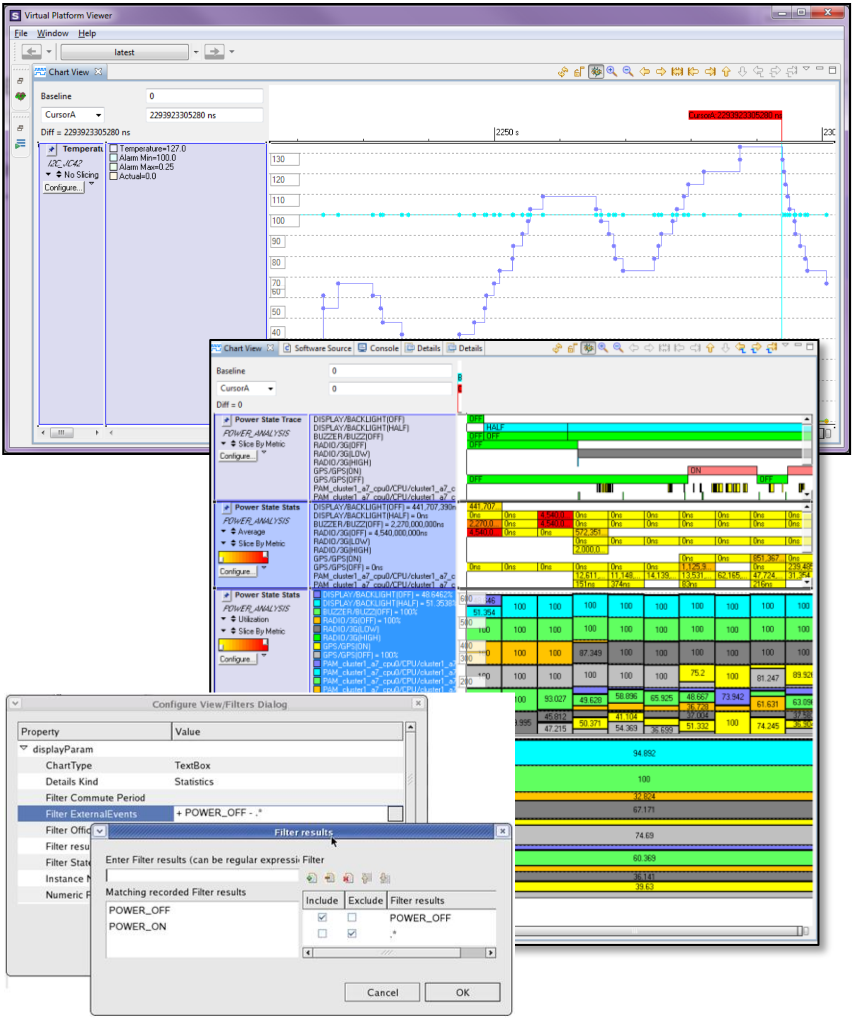
Firmware symbols

See article: **Android hardware/software design using virtual prototypes**
http://www.embedded.com/design/prototyping-and-development/4399520

Enabling analysis

Analysis infrastructure for Virtual Prototype TLM (models)

- Simple APIs for Models
  - Message logging
  - Data tracing
  - State tracing
- SW awareness plugins
  - (RT)OS threads and processes
  - Exceptions
- Out-of-the-box analysis
  - Traces
  - Configurable statistics
  - Filters/Regular Expressions



Baseline 0
CursorA 2293923305280 ns
Diff = 2293923305280 ns

Temperatu
I2C_JC42
No Slicing
- Temperature=127.0
- Alarm Min=100.0
- Alarm Max=0.25
- Actual=0.0

Baseline 0
CursorA 0
Diff = 0

Power State Trace
POWER_ANALYSIS
Slice By Metric
- DISPLAY/BACKLIGHT(OFF)
- DISPLAY/BACKLIGHT(HALF)
- BUZZER/BUZZ(OFF)
- RADIO/3G(OFF)
- RADIO/3G(LOW)
- RADIO/3G(HIGH)
- GPS/GPS(ON)
- GPS/GPS(OFF)
- PAM_cluster1_a7_cpu0/CPU/cluster1_a7_c
- PAM_cluster1_a7_cpu0/CPU/cluster1_a7_c

Power State Stats
POWER_ANALYSIS
Average
Slice By Metric
- DISPLAY/BACKLIGHT(OFF) = 441,707,390n
- DISPLAY/BACKLIGHT(HALF) = 0ns
- BUZZER/BUZZ(OFF) = 2,270,000,000ns
- RADIO/3G(OFF) = 4,540,000,000ns
- RADIO/3G(LOW)
- RADIO/3G(HIGH)
- GPS/GPS(ON)
- GPS/GPS(OFF) = 0ns
- PAM_cluster1_a7_cpu0/CPU/cluster1_a7_c
- PAM_cluster1_a7_cpu0/CPU/cluster1_a7_c

Power State Stats
POWER_ANALYSIS
Utilization
Slice By Metric
- DISPLAY/BACKLIGHT(OFF) = 48.6462%
- DISPLAY/BACKLIGHT(HALF) = 51.3538%
- BUZZER/BUZZ(OFF) = 100%
- RADIO/3G(OFF) = 100%
- RADIO/3G(LOW)
- RADIO/3G(HIGH)
- GPS/GPS(ON)
- GPS/GPS(OFF) = 100%
- PAM_cluster1_a7_cpu0/CPU/cluster1_a7
- PAM_cluster1_a7_cpu0/CPU/cluster1_a7
- PAM_cluster1_a7_cpu0/CPU/cluster1_a7
- PAM_cluster1_a7_cpu0/CPU/cluster1_a7

Configure View/Filters Dialog

Property	Value
displayParam	
ChartType	TextBox
Details Kind	Statistics
Filter Commute Period	
Filter ExternalEvents	+ POWER_OFF - .*
Filter Offic	
Filter resu	
Filter State	
Instance N	
Numeric F	

Filter results

Enter Filter results (can be regular expressi Filter

Matching recorded Filter results
- POWER_OFF
- POWER_ON

Include	Exclude	Filter results
☑	□	POWER_OFF
□	☑	.*

Summary

Virtual Prototypes

- Can enable top-down debug process

Needs:

- Models needs to be instrumented, delivering traces at the right level of abstraction

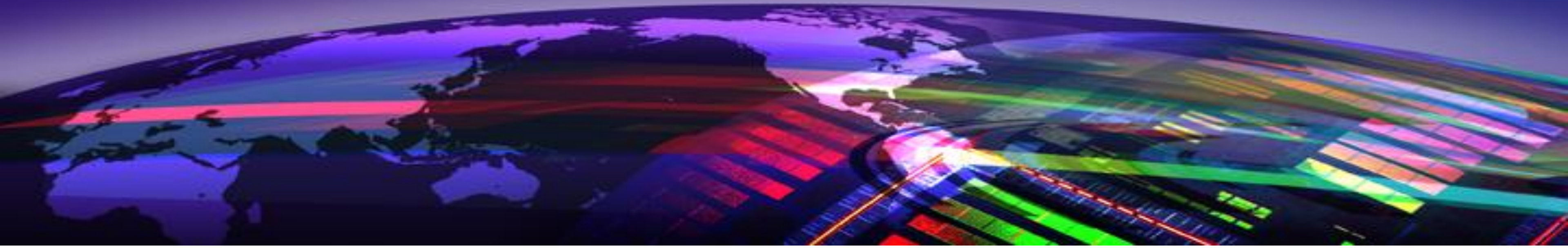
Challenges:

- Tracing and logging APIs not part of the IEEE 1666 SystemC TLM-2.0 standard

Thank You

SYNOPSYS®

Accelerating Innovation



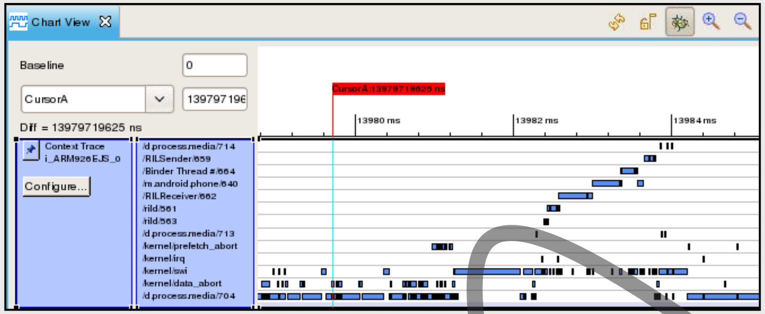
Backup

System Level Software Debug & Analysis

Consistent top-down analysis

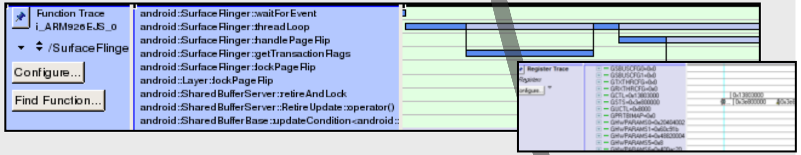
Multi-core OS Level Process Trace:

Global system SW root-cause analysis



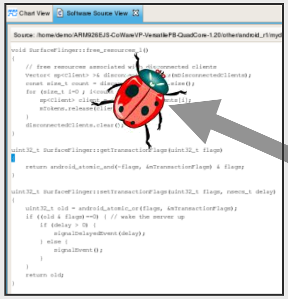
Function Trace/ Register Trace

Local function level root-cause analysis


Function Trace
i_ARM926EJS_0
/SurfaceFlinge
Configure...
Find Function...
android::SurfaceFlinger::waitForEvent
android::SurfaceFlinger::threadLoop
android::SurfaceFlinger::handlePageFlip
android::SurfaceFlinger::getTransactionFlags
android::SurfaceFlinger::lockPageFlip
android::Layer::lockPageFlip
android::SharedBufferServer::retireAndLock
android::SharedBufferServer::RetireUpdate::operator()
android::SharedBufferBase::updateCondition<android::
Register Trace
GSBUSCFG0=0x0
GSBUSCFG1=0x0
GTXTHRCFG=0x0
GRXTHRCFG=0x0
GCTL=0x13803000
GSTS=0x3e800000
GUCTL=0x8000
GPRTBIMAP=0x0
GHWPARAMS0=0x20404002
GHWPARAMS1=0x60c91b
GHWPARAMS4=0x48820004
GHWPARAMS5=0x8
0x13803000
0x3e800000

Source Code

Detailed analysis



Source: /home/demo/ARM926EJS-CoWareVP-VersatilePB-QuadCore-1.20/other/android_r1/myd

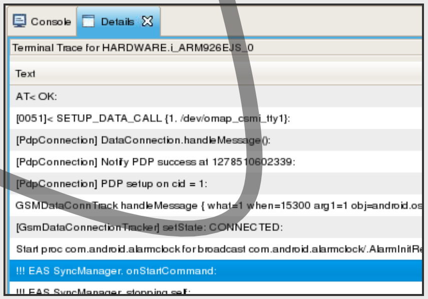
```
void SurfaceFlinger::free_resources_l()
{
    // free resources associated with disconnected clients
    Vector< sp<Client> >& discon                (mDisconnectedClients);
    const size_t count = disc              .size();
    for (size_t i=0 ; i<count
        sp<Client> client               nts[i];
        mTokens.release(clie
    }
    disconnectedClients.clear();
}

uint32_t SurfaceFlinger::getTransactionFlags(uint32_t flags)
{
    return android_atomic_and(~flags, &mTransactionFlags) & flags;
}

uint32_t SurfaceFlinger::setTransactionFlags(uint32_t flags, nsecs_t delay)
{
    uint32_t old = android_atomic_or(flags, &mTransactionFlags);
    if ((old & flags)==0) { // wake the server up
        if (delay > 0) {
            signalDelayedEvent(delay);
        } else {
            signalEvent();
        }
    }
    return old;
}
```

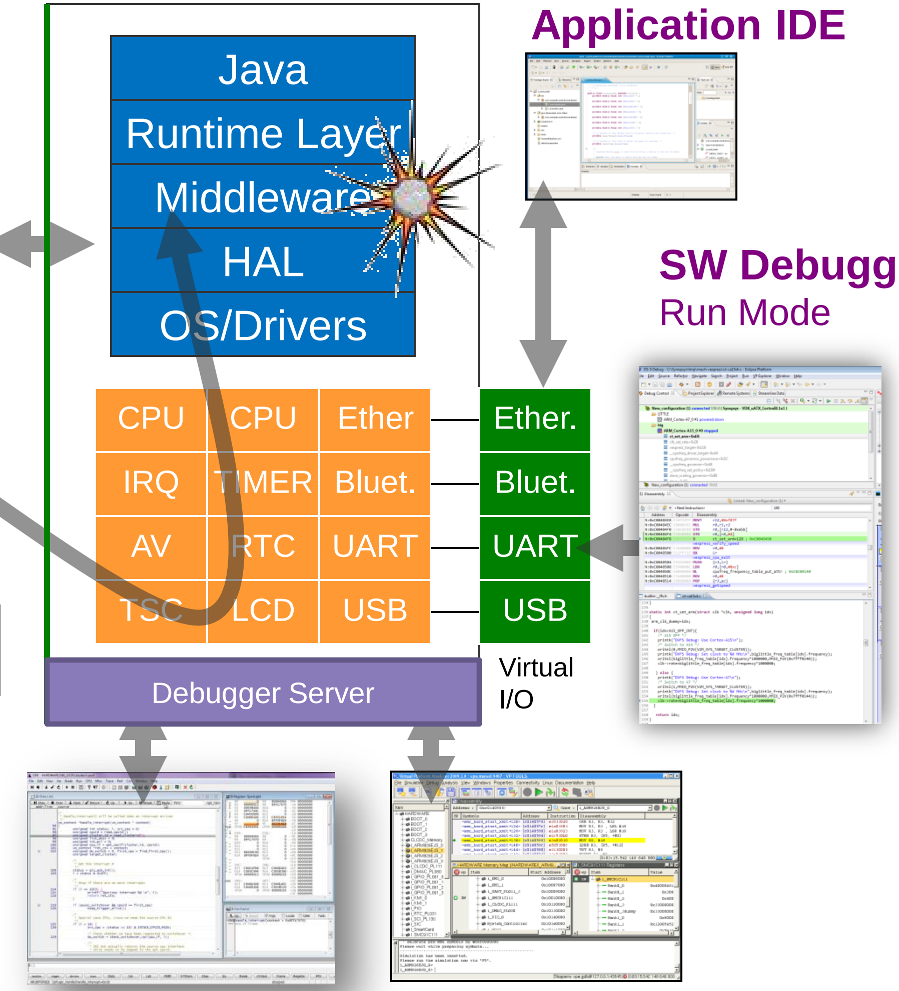
Software Logging

Kernel log, printfs etc.



Terminal Trace for HARDWARE.i_ARM926EJS_0

```
Text
AT< OK:
[0051]< SETUP_DATA_CALL {1. /dev/omap_csmi_tty1}:
[PdpConnection] DataConnection.handleMessage():
[PdpConnection] Notify PDP success at 1278510602339:
[PdpConnection] PDP setup on cid = 1:
GSMDataConnTrack handleMessage { what=1 when=15300 arg1=1 obj=android.os
[GsmDataConnectionTracker] setState: CONNECTED:
Start proc com.android.alarmclock for broadcast com.android.alarmclock/.AlarmInitRe
!!! EAS SyncManager. onStartCommand:
!!! EAS SyncManager. stopping self:
```



SW Debugger

Stop Mode

System HW/SW Debug

Registers, Signals, Memory, Code

SYNOPSYS® Accelerating Innovation