spEEDO: Energy Efficiency through Debug suppOrt (& On Chip Analytics)

PEHAM Project: Power estimation from high-level models

David Greaves
Ali Zaidi
Klaus McDonald Maier

University of Cambridge
Computer Laboratory
And Ultrasoc Ltd
NMI Multicore Meeting,
Cambridge March 2014.

Computer Laboratory Research 1

- Energy Management Techniques in Modern Mobile Handsets

(N Vallina-Rodriguez, J Crowcroft, IEEE COMMUNICATIONS SURVEYS 2012).

- Dynamic Microarchitectural Adaptation Using Machine Learning

(C Dubach, TM Jones EV Bonilla , ACM Transactions on Architecture and Code Optimization TACO 2013)

- The Smart Cache: An Energy-Efficient Cache Architecture Through Dynamic Adaptation

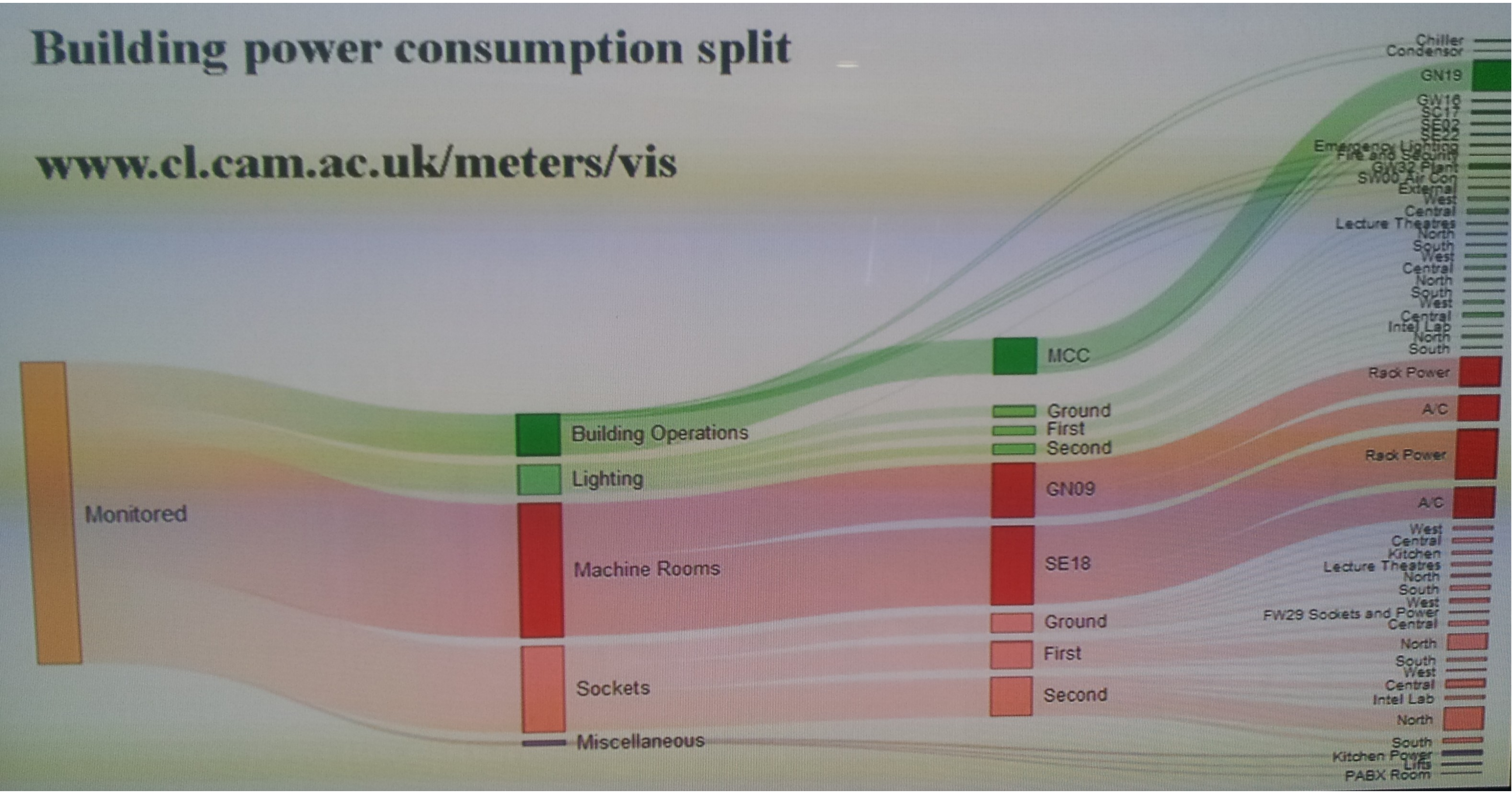
(KT Sundararajan, TM Jones and NP Topham International Journal of Parallel Programming 2013)

- Computer Laboratory: C-AWARE

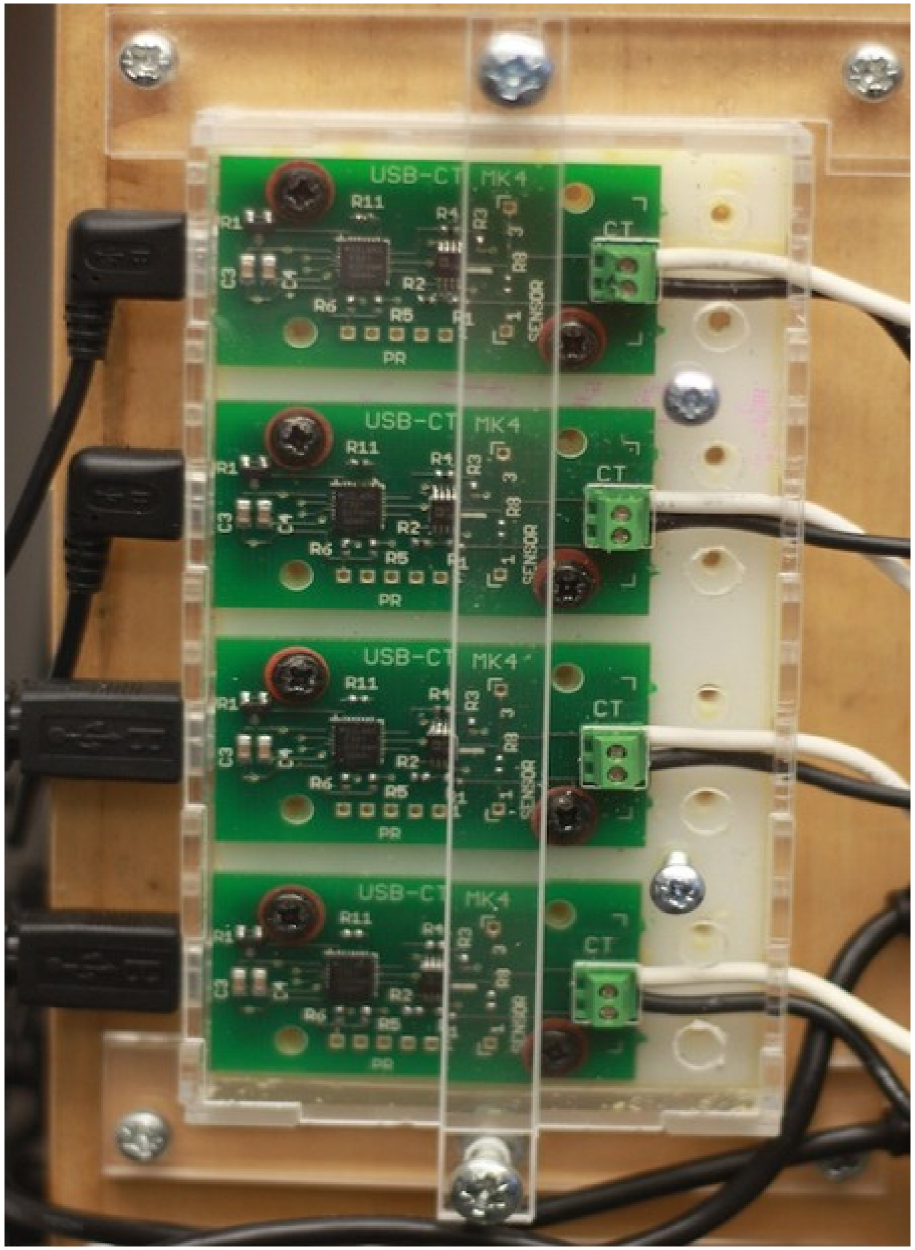
C-AWARE aims to build services to improve users' awareness of their personal energy consumption, and modify their energy demand.

Gates Building Power

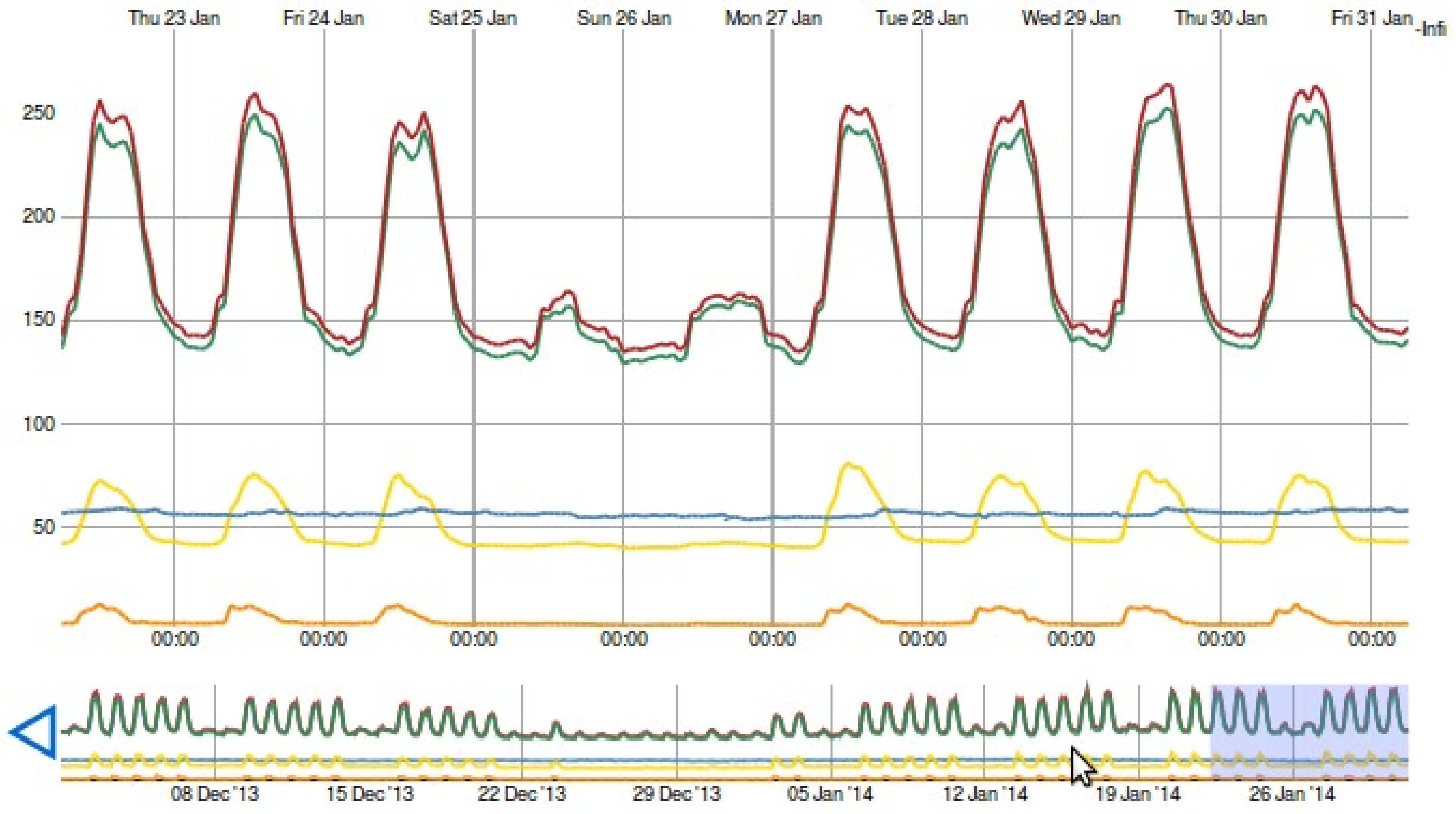
We have a log of nearly all the power used in our building in Cambridge.



That's in the *C-Aware Project* which has installed monitors on all the mains cables in the switch room.



The picture shows just four of many.

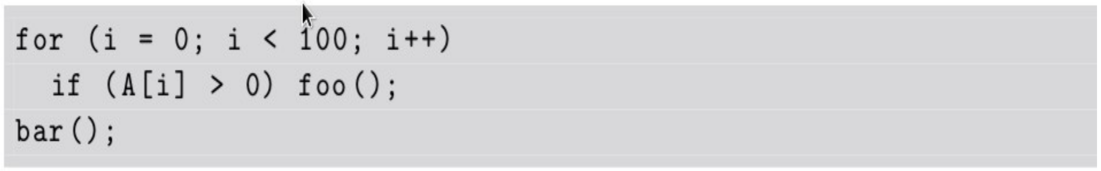


Col...	Description	Start	End	Avg kW Selected	Avg kW Entire	Total Energy (kWh)
🟥	Entire Building	Dec 2013	Jan 2014	179.78	161.02	236,542
🟩	Logical Sum of Sub Meters	Dec 2013	Jan 2014	172.72	154.53	227,007
🟦	Machine Rooms	Dec 2013	Jan 2014	56.46	57.44	84,381
🟨	Sockets	Dec 2013	Jan 2014	50.90	46.10	67,727
🟧	Miscellaneous	Dec 2013	Jan 2014	4.85	4.01	5,889

Computer Laboratory Research 2

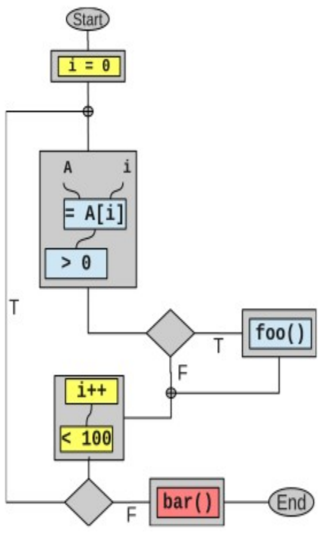
- TLM POWER 3: Power Estimation Methodology for SystemC TLM 2.0' (DJ Greaves & MM Yasin, FDL'12)

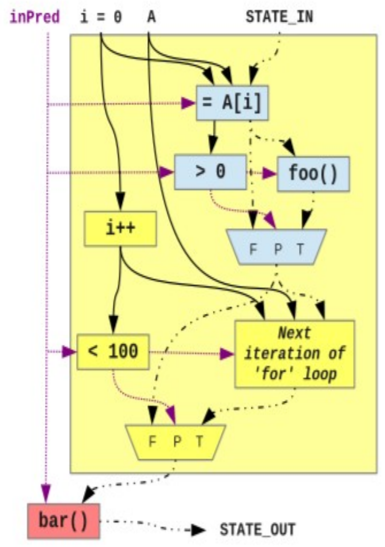
- KIWI – Compiling dotnet C# programs to FPGA for low energy execution (DJ Greaves + S Singh).



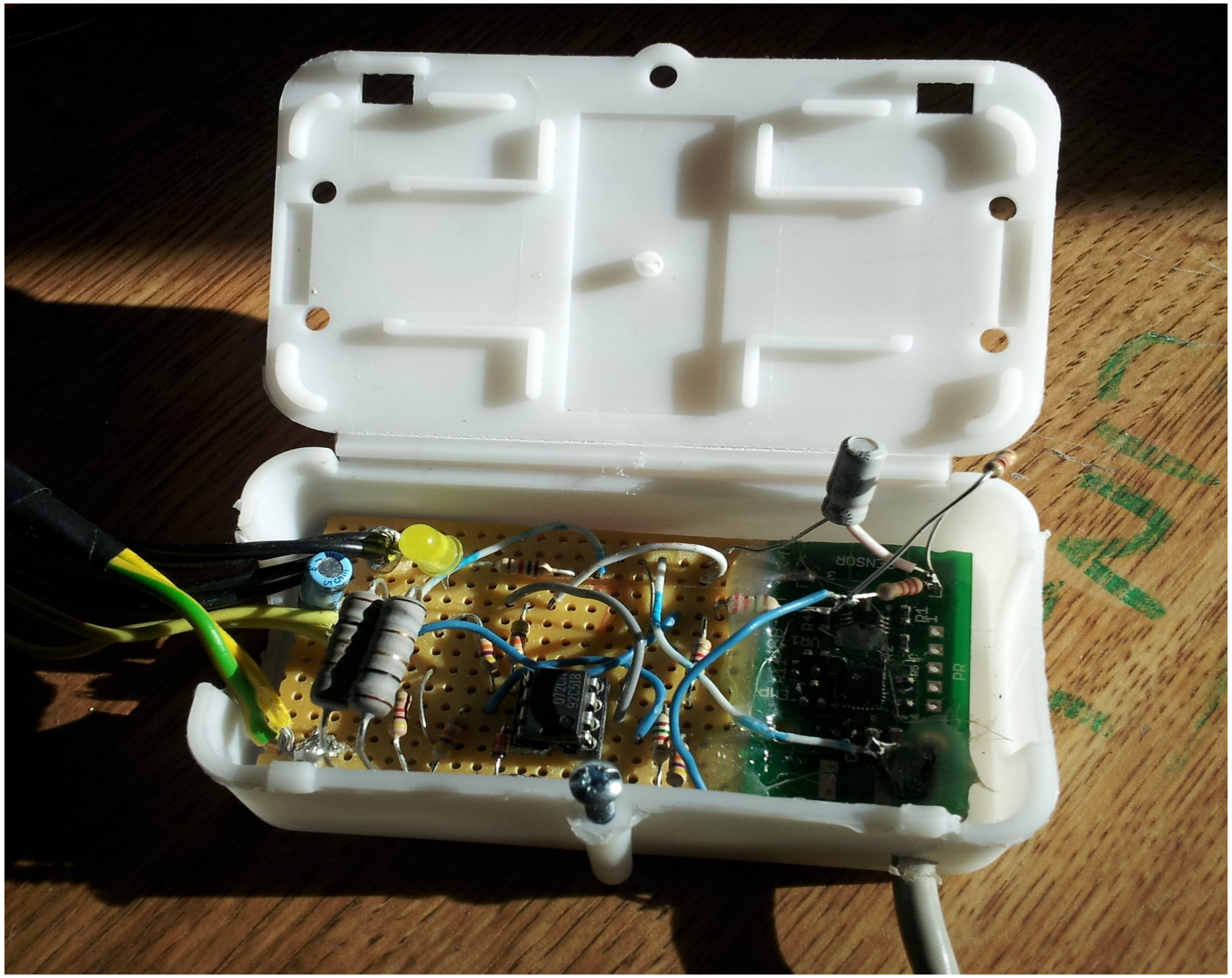
```
for (i = 0; i < 100; i++)
  if (A[i] > 0) foo();
bar();
```

- Achieving Superscalar Performance without Superscalar Overheads – A Dataflow Compiler IR for Custom Computing (AM Zaidi and DJ Greaves).





PC CPU Power Probe



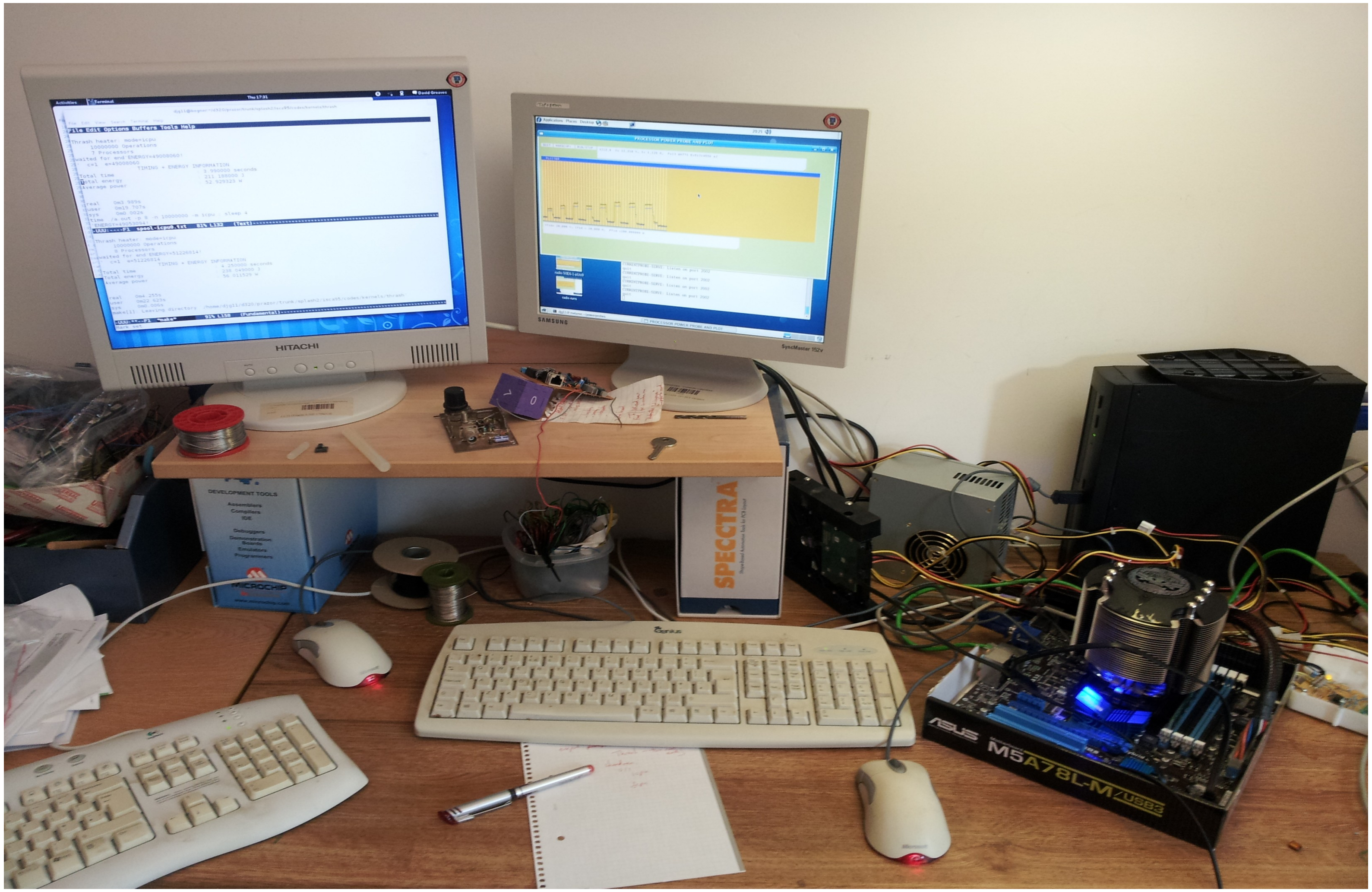
The same USB probe

Measures 12 volt rail to motherboard CPU socket.

Measures volts and amps at 10 Hz rate.

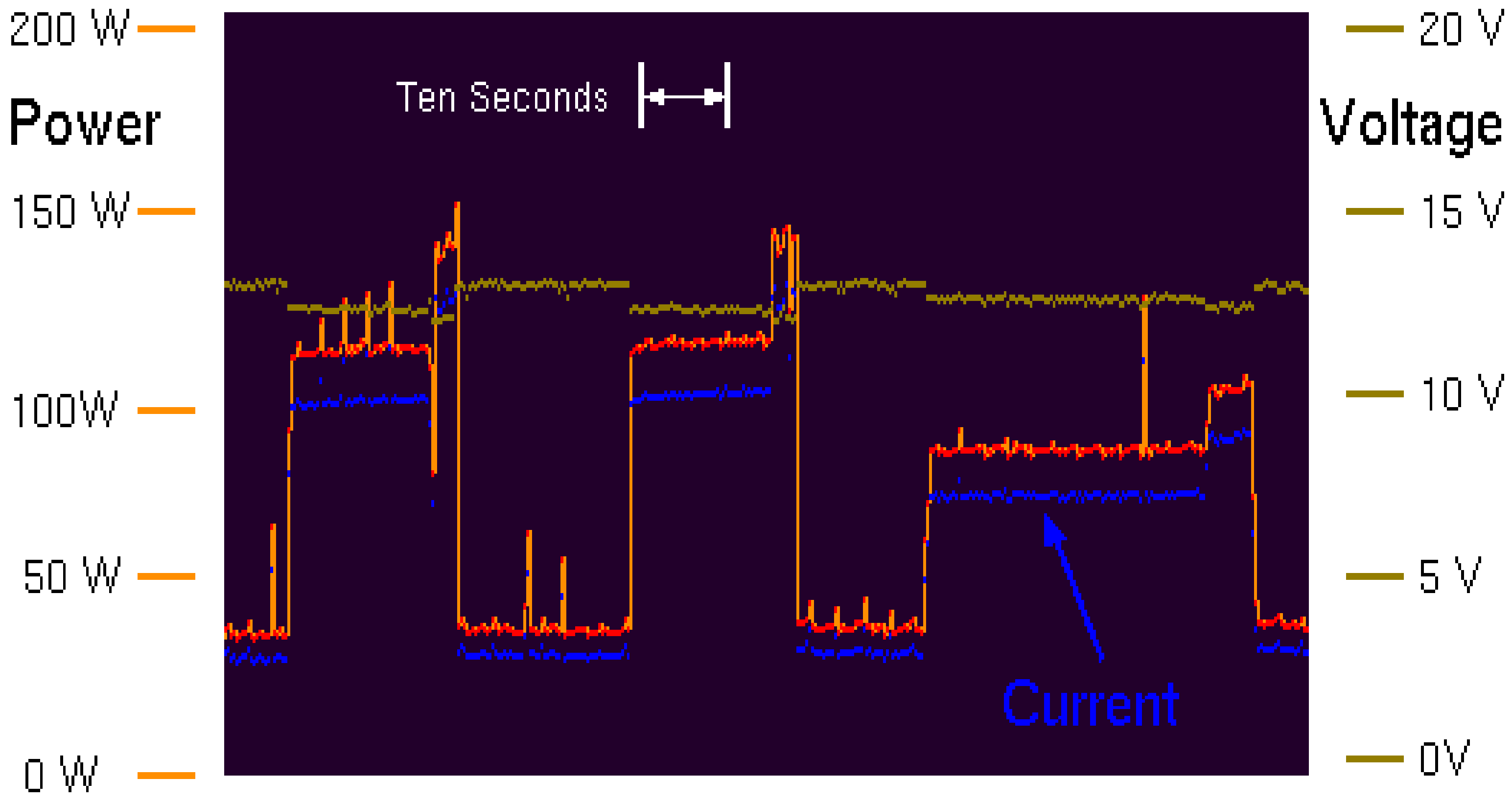
Accuracy:
  consistency of about 1 percent between runs.

Probed and Probing Machines



AMD 6-Core Phenom 64 Processor with TCP connection to power probe machine.

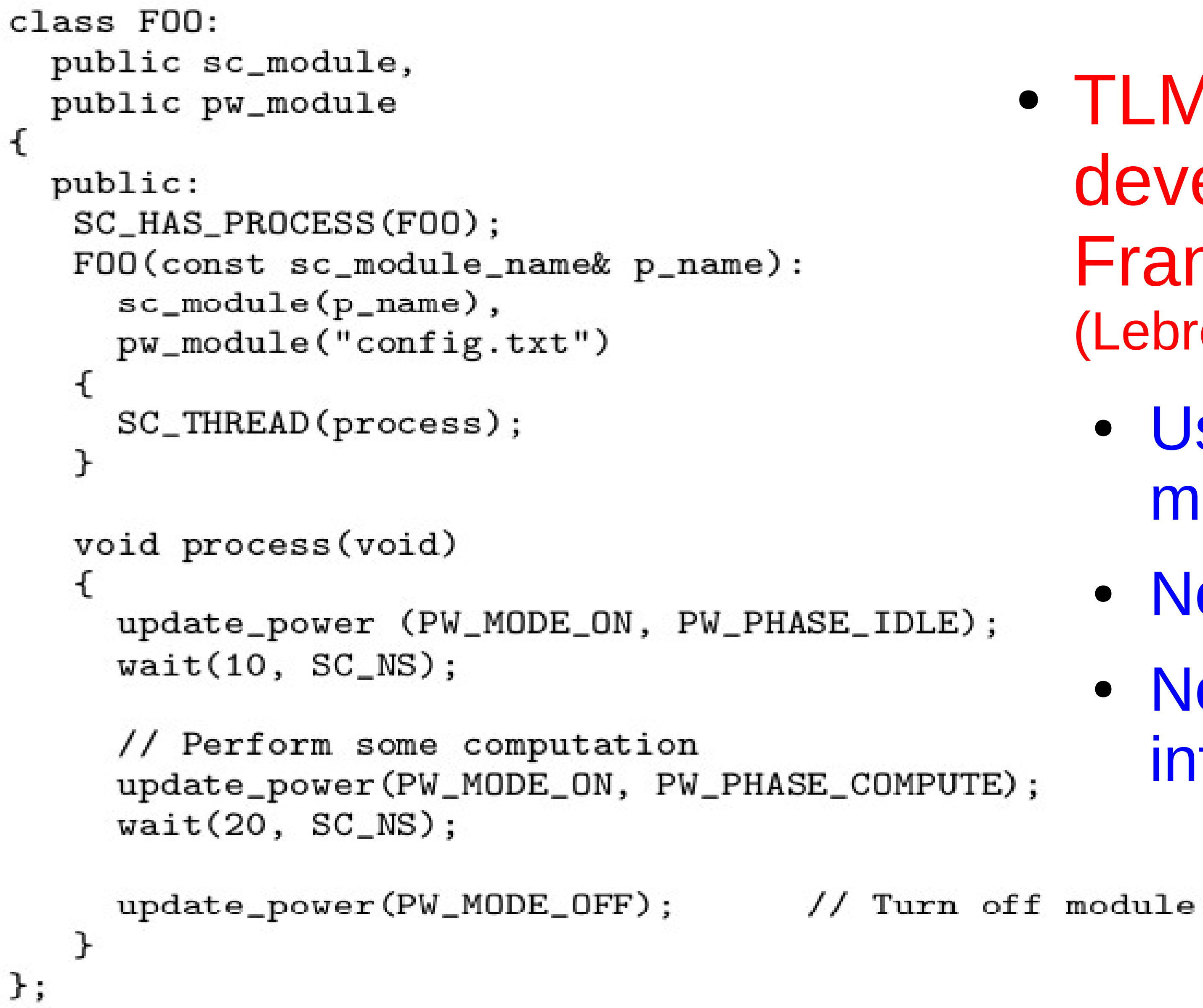
Splash-2 'RADIX' : First Test Setup



Plot shows two runs with two cores and then one run with one core.

Problem: Power probe was running on same machine (spikes).
Problem: Some spikes missed owing to aliasing (missing ADC LPF).
Fixed thereafter (use separate probe machine and add an RC filter).
Note: this older CPU used 3x power compared with Phenom...

TLM Power 2 Library



```
class FOO:
  public sc_module,
  public pw_module
{
  public:
   SC_HAS_PROCESS(FOO);
   FOO(const sc_module_name& p_name):
     sc_module(p_name),
     pw_module("config.txt")
   {
     SC_THREAD(process);
   }

   void process(void)
   {
     update_power (PW_MODE_ON, PW_PHASE_IDLE);
     wait(10, SC_NS);

     // Perform some computation
     update_power(PW_MODE_ON, PW_PHASE_COMPUTE);
     wait(20, SC_NS);

     update_power(PW_MODE_OFF);       // Turn off module
   }
};
```

- TLM POWER 2 developed at France CEA (Lebreton/Vivet)
  - Used phase/mode modelling
  - No LT
  - No TLM socket integration.

TLM POWER 3: Motivation

- Power estimation from high-level models.
- Rapid architectural exploration using SystemC.
- Absolute accuracy goal: correct order of magnitude at least!
- Relative accuracy goal: 30 percent or so.

- Want correct polarity of the parameter derivatives : *A change is better or worse*!

Physical Units

- SystemC provides overloaded sc_time units
- TLM POWER 2 added pw_energy and pw_power units with all appropriate overloads.
- TLM POWER 3 adds pw_voltage for F/V scaling.
- TLM POWER 3 also adds pw_length and pw_area.

Basic physics: energy divided by time ---> power

Basic physics: length times length ---> area

Setting Static Parameters

```
class FOO:
  public sc_module,
  public pw_module
{
  public:
   SC_HAS_PROCESS(FOO);
   FOO(const sc_module_name& p_name, int width):
     sc_module(p_name),
     pw_module("config.txt")
   {
     set_excess_area(pw_length(50.0 * width, PW_um), pw_length(5.0 * width, PW_um));

   }
};
```

Excess area: the local increment above the sum of the instantiated modules below.

Typically set the area and static power in the constructor.

Example: for a RAM, the area can be dependent on the number if bits.

LT b_transport energy annotation

```
tac_response tac_multiport_router::b_transport(tlm_generic_payload &trans, sc_time &delay)
{
  unsigned int len = trans.get_data_length();

   ... // Main body of the behavioural model

  sc_time activity_time = ...;

  delay += lt_activity_time; // Or use qk_inc to perform this addition

#ifdef TLM_POWER3
  // bit_width has been set in the constructor... etc
  sc_energy energy_cost = pw_energy((double) (5 * len), pw_energy_unit::PW_pJ);
  pw_module_base::update_energy(energy_cost, lt_activity_time);
#endif
}
```

Bad:

This shows computation of energy per transaction in the body of the transaction.

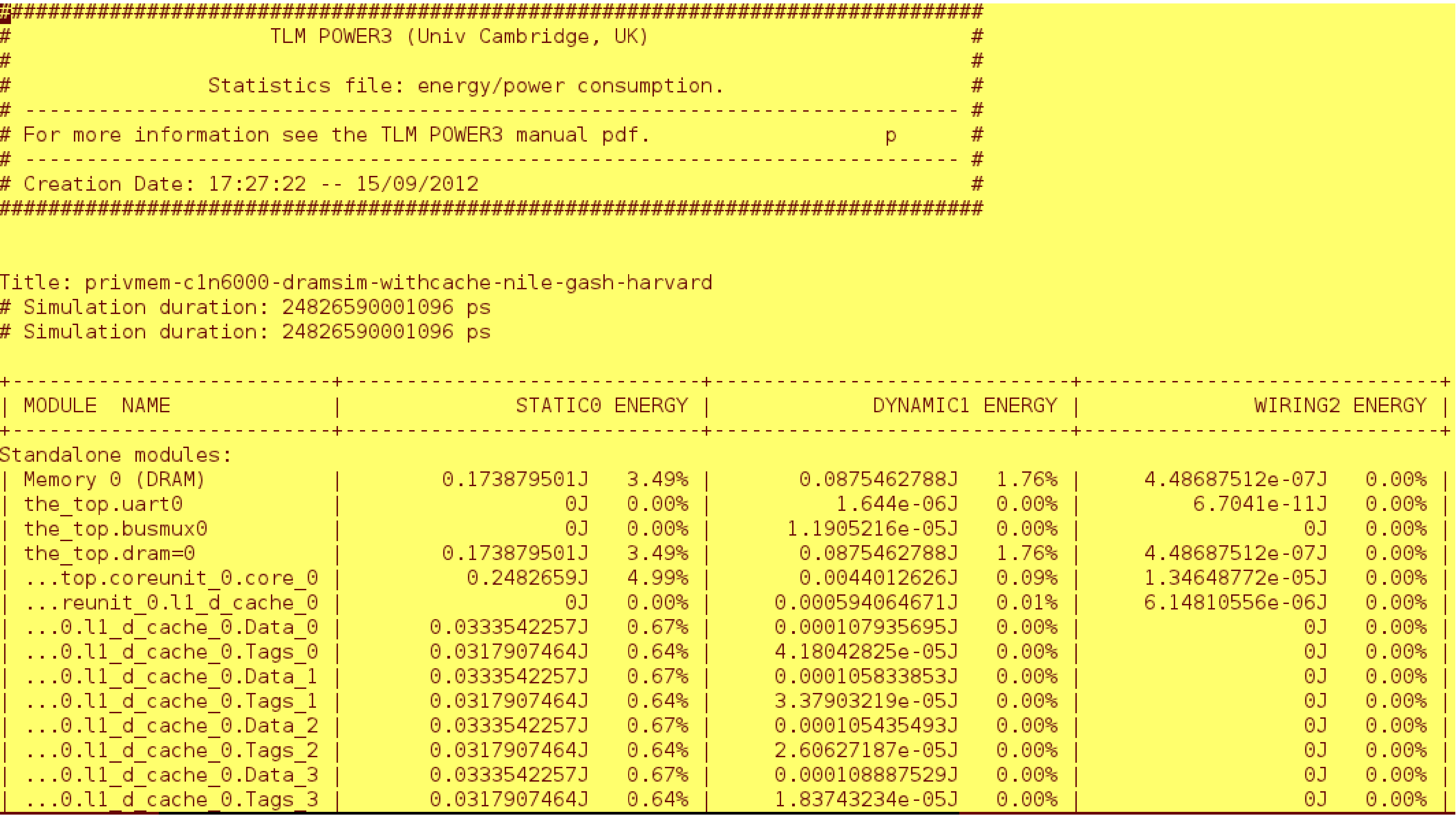
Better:

Energy and floating point computations done in RECOMPUTE_PVT callback.

Spatial Layout Support

- Every SC_MODULE has a chip/region designation.
- The area of a module is sum of
  - its children with the same chip/region name
  - its locally defined 'excess area'.
- Inter-module wiring lengths can be estimated using Rent's Rule on area of lowest-common-parent.
- Actual X-Y co-ordinates could be allocated by a placer.

Report Formats (2: Ascii-art text file)



```
################################################################################
#                     TLM POWER3 (Univ Cambridge, UK)                          #
#                                                                              #
#                Statistics file: energy/power consumption.                    #
# ---------------------------------------------------------------------------- #
# For more information see the TLM POWER3 manual pdf.                   p      #
# ---------------------------------------------------------------------------- #
# Creation Date: 17:27:22 -- 15/09/2012                                        #
################################################################################


Title: privmem-c1n6000-dramsim-withcache-nile-gash-harvard
# Simulation duration: 24826590001096 ps
# Simulation duration: 24826590001096 ps

+--------------------------+-----------------------------+-----------------------------+-----------------------------+
| MODULE  NAME             |               STATIC0 ENERGY |             DYNAMIC1 ENERGY |               WIRING2 ENERGY |
+--------------------------+-----------------------------+-----------------------------+-----------------------------+
Standalone modules:
| Memory 0 (DRAM)          |        0.173879501J   3.49% |       0.0875462788J   1.76% |     4.48687512e-07J   0.00% |
| the_top.uart0            |                  0J   0.00% |          1.644e-06J   0.00% |         6.7041e-11J   0.00% |
| the_top.busmux0          |                  0J   0.00% |      1.1905216e-05J   0.00% |                  0J   0.00% |
| the_top.dram=0           |        0.173879501J   3.49% |       0.0875462788J   1.76% |     4.48687512e-07J   0.00% |
| ...top.coreunit_0.core_0 |          0.2482659J   4.99% |       0.0044012626J   0.09% |     1.34648772e-05J   0.00% |
| ...reunit_0.l1_d_cache_0 |                  0J   0.00% |     0.000594064671J   0.01% |     6.14810556e-06J   0.00% |
| ...0.l1_d_cache_0.Data_0 |       0.0333542257J   0.67% |     0.000107935695J   0.00% |                  0J   0.00% |
| ...0.l1_d_cache_0.Tags_0 |       0.0317907464J   0.64% |     4.18042825e-05J   0.00% |                  0J   0.00% |
| ...0.l1_d_cache_0.Data_1 |       0.0333542257J   0.67% |     0.000105833853J   0.00% |                  0J   0.00% |
| ...0.l1_d_cache_0.Tags_1 |       0.0317907464J   0.64% |     3.37903219e-05J   0.00% |                  0J   0.00% |
| ...0.l1_d_cache_0.Data_2 |       0.0333542257J   0.67% |     0.000105435493J   0.00% |                  0J   0.00% |
| ...0.l1_d_cache_0.Tags_2 |       0.0317907464J   0.64% |     2.60627187e-05J   0.00% |                  0J   0.00% |
| ...0.l1_d_cache_0.Data_3 |       0.0333542257J   0.67% |     0.000108887529J   0.00% |                  0J   0.00% |
| ...0.l1_d_cache_0.Tags_3 |       0.0317907464J   0.64% |     1.83743234e-05J   0.00% |                  0J   0.00% |
```

spEEDO

- spEEDO: Energy Efficiency through Debug suppOrt
- University of Cambridge Computer Laboratory in Collaboration with Ultrasoc Limited.
- Funded for six months by the UK TSB
- Started October 2013

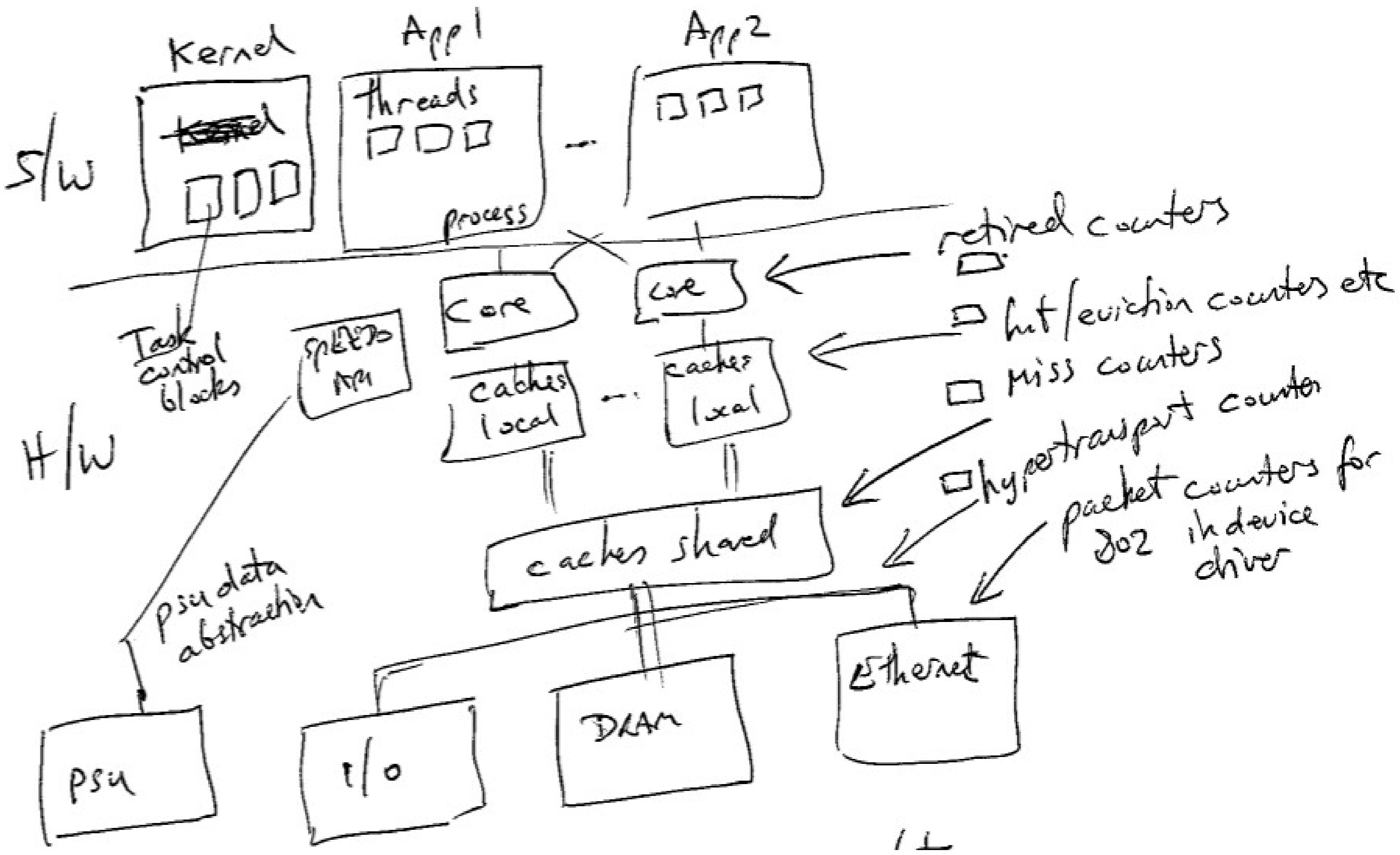
spEEDO

- Develop a power API for three purposes:
  - Embedded software energy reflection API
  - Remote debugger energy accounting and logging
  - Debug access to power-gated regions

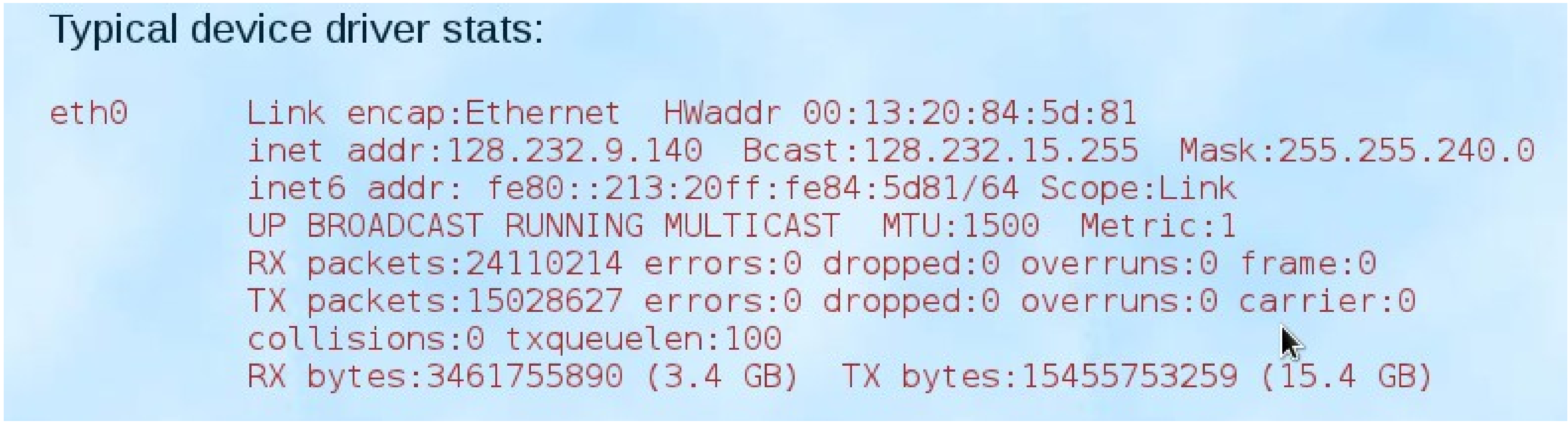
Current activities:

- Develop a strawman energy API for access to '*On Chip Analytics*'
- Trials on SystemC virtual SoC
- Extend GDB schemas for energy regs

Reference Architecture



Existing Power Events



Typical device driver stats:

```
eth0      Link encap:Ethernet  HWaddr 00:13:20:84:5d:81
          inet addr:128.232.9.140  Bcast:128.232.15.255  Mask:255.255.240.0
          inet6 addr: fe80::213:20ff:fe84:5d81/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:24110214 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15028627 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:3461755890 (3.4 GB)  TX bytes:15455753259 (15.4 GB)
```

Existing event counters in device drivers and hardware can be projected through a calibration matrix to give energy estimates.

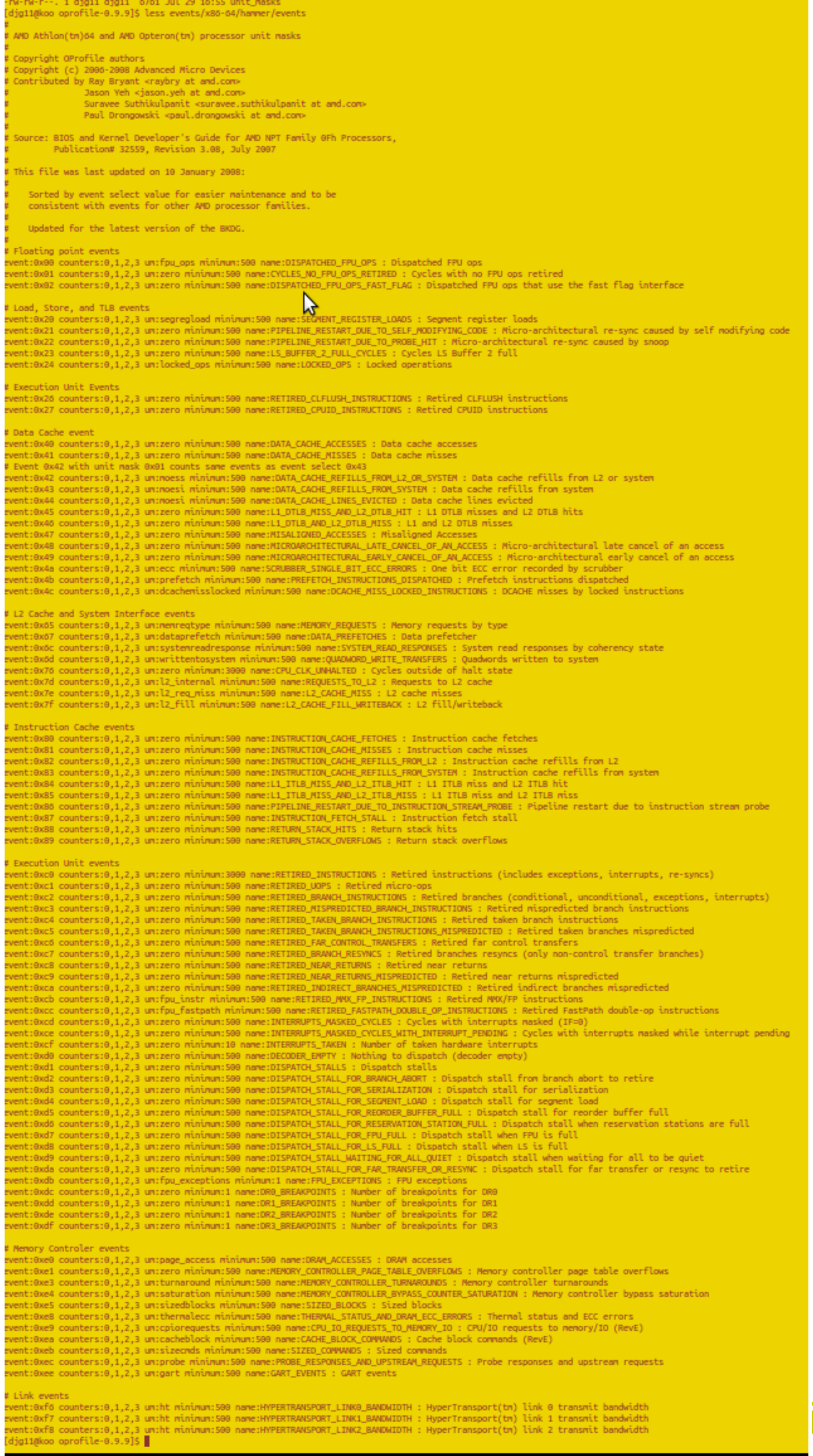
MSRs

Machine-Specific Registers:

Oprofile example.

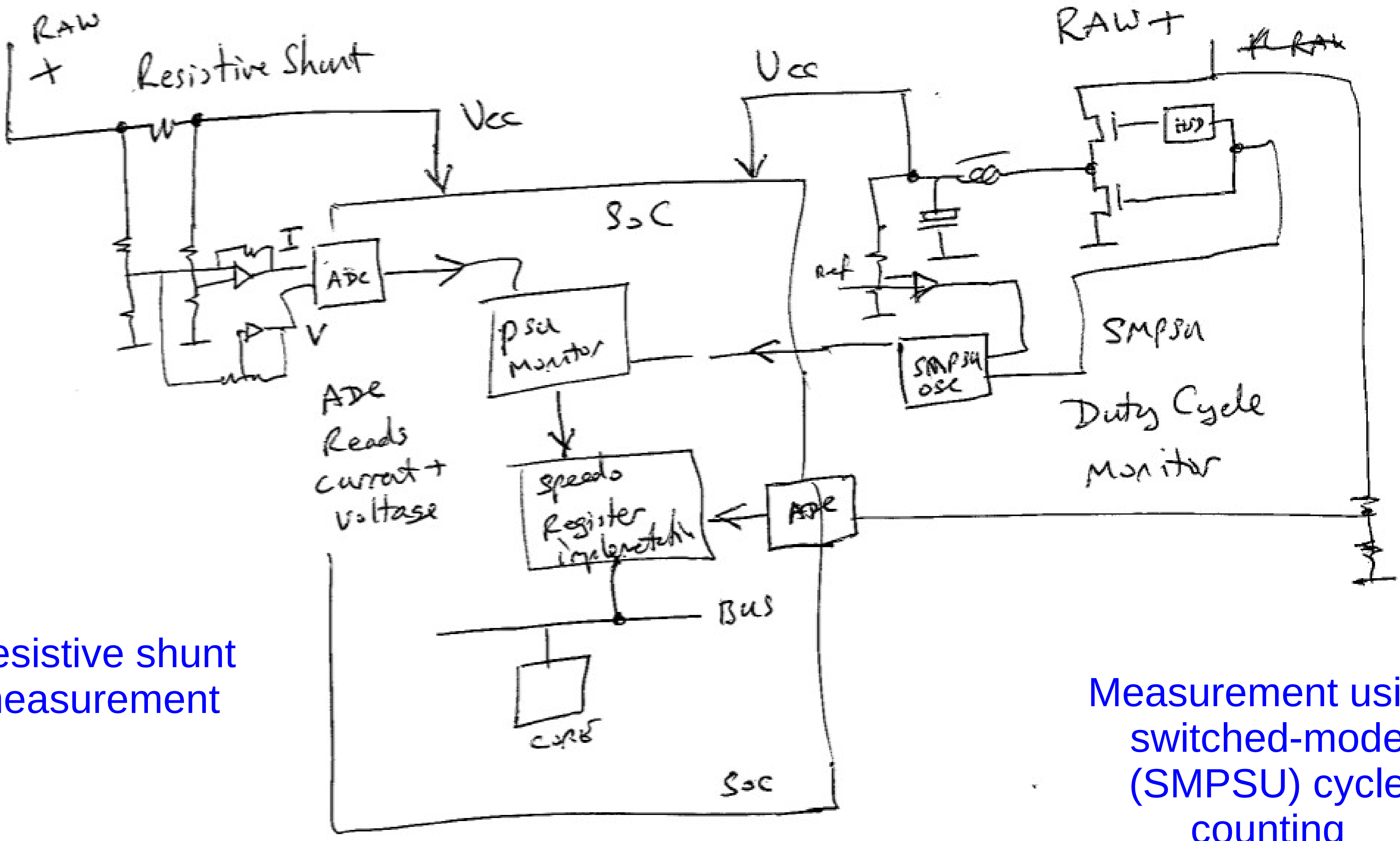
Oprofile gives
a uniform API to a wide
variety of hardware
platforms.

Listing shows monitorable
event counters on AMD
x86-Hammer



```
-rw-rw-r--. 1 djg11 djg11  6761 Jul 29 16:55 unit_masks
[djg11@koo oprofile-0.9.9]$ less events/x86-64/hammer/events
#
# AMD Athlon(tm)64 and AMD Opteron(tm) processor unit masks
#
# Copyright OProfile authors
# Copyright (c) 2006-2008 Advanced Micro Devices
# Contributed by Ray Bryant <raybry at amd.com>
#               Jason Yeh <jason.yeh at amd.com>
#               Suravee Suthikulpanit <suravee.suthikulpanit at amd.com>
#               Paul Drongowski <paul.drongowski at amd.com>
#
# Source: BIOS and Kernel Developer's Guide for AMD NPT Family 0Fh Processors,
#         Publication# 32559, Revision 3.08, July 2007
#
# This file was last updated on 10 January 2008:
#
#    Sorted by event select value for easier maintenance and to be
#    consistent with events for other AMD processor families.
#
#    Updated for the latest version of the BKDG.
#
# Floating point events
event:0x00 counters:0,1,2,3 um:fpu_ops minimum:500 name:DISPATCHED_FPU_OPS : Dispatched FPU ops
event:0x01 counters:0,1,2,3 um:zero minimum:500 name:CYCLES_NO_FPU_OPS_RETIRED : Cycles with no FPU ops retired
event:0x02 counters:0,1,2,3 um:zero minimum:500 name:DISPATCHED_FPU_OPS_FAST_FLAG : Dispatched FPU ops that use the fast flag interface

# Load, Store, and TLB events
event:0x20 counters:0,1,2,3 um:segregload minimum:500 name:SEGMENT_REGISTER_LOADS : Segment register loads
event:0x21 counters:0,1,2,3 um:zero minimum:500 name:PIPELINE_RESTART_DUE_TO_SELF_MODIFYING_CODE : Micro-architectural re-sync caused by self modifying code
event:0x22 counters:0,1,2,3 um:zero minimum:500 name:PIPELINE_RESTART_DUE_TO_PROBE_HIT : Micro-architectural re-sync caused by snoop
event:0x23 counters:0,1,2,3 um:zero minimum:500 name:LS_BUFFER_2_FULL_CYCLES : Cycles LS Buffer 2 full
event:0x24 counters:0,1,2,3 um:locked_ops minimum:500 name:LOCKED_OPS : Locked operations

# Execution Unit Events
event:0x26 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_CLFLUSH_INSTRUCTIONS : Retired CLFLUSH instructions
event:0x27 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_CPUID_INSTRUCTIONS : Retired CPUID instructions

# Data Cache event
event:0x40 counters:0,1,2,3 um:zero minimum:500 name:DATA_CACHE_ACCESSES : Data cache accesses
event:0x41 counters:0,1,2,3 um:zero minimum:500 name:DATA_CACHE_MISSES : Data cache misses
# Event 0x42 with unit mask 0x01 counts same events as event select 0x43
event:0x42 counters:0,1,2,3 um:moess minimum:500 name:DATA_CACHE_REFILLS_FROM_L2_OR_SYSTEM : Data cache refills from L2 or system
event:0x43 counters:0,1,2,3 um:moesi minimum:500 name:DATA_CACHE_REFILLS_FROM_SYSTEM : Data cache refills from system
event:0x44 counters:0,1,2,3 um:moesi minimum:500 name:DATA_CACHE_LINES_EVICTED : Data cache lines evicted
event:0x45 counters:0,1,2,3 um:zero minimum:500 name:L1_DTLB_MISS_AND_L2_DTLB_HIT : L1 DTLB misses and L2 DTLB hits
event:0x46 counters:0,1,2,3 um:zero minimum:500 name:L1_DTLB_AND_L2_DTLB_MISS : L1 and L2 DTLB misses
event:0x47 counters:0,1,2,3 um:zero minimum:500 name:MISALIGNED_ACCESSES : Misaligned Accesses
event:0x48 counters:0,1,2,3 um:zero minimum:500 name:MICROARCHITECTURAL_LATE_CANCEL_OF_AN_ACCESS : Micro-architectural late cancel of an access
event:0x49 counters:0,1,2,3 um:zero minimum:500 name:MICROARCHITECTURAL_EARLY_CANCEL_OF_AN_ACCESS : Micro-architectural early cancel of an access
event:0x4a counters:0,1,2,3 um:ecc minimum:500 name:SCRUBBER_SINGLE_BIT_ECC_ERRORS : One bit ECC error recorded by scrubber
event:0x4b counters:0,1,2,3 um:prefetch minimum:500 name:PREFETCH_INSTRUCTIONS_DISPATCHED : Prefetch instructions dispatched
event:0x4c counters:0,1,2,3 um:dcachemisslocked minimum:500 name:DCACHE_MISS_LOCKED_INSTRUCTIONS : DCACHE misses by locked instructions

# L2 Cache and System Interface events
event:0x65 counters:0,1,2,3 um:memreqtype minimum:500 name:MEMORY_REQUESTS : Memory requests by type
event:0x67 counters:0,1,2,3 um:dataprefetch minimum:500 name:DATA_PREFETCHES : Data prefetcher
event:0x6c counters:0,1,2,3 um:systemreadresponse minimum:500 name:SYSTEM_READ_RESPONSES : System read responses by coherency state
event:0x6d counters:0,1,2,3 um:writtentosystem minimum:500 name:QUADWORD_WRITE_TRANSFERS : Quadwords written to system
event:0x76 counters:0,1,2,3 um:zero minimum:3000 name:CPU_CLK_UNHALTED : Cycles outside of halt state
event:0x7d counters:0,1,2,3 um:l2_internal minimum:500 name:REQUESTS_TO_L2 : Requests to L2 cache
event:0x7e counters:0,1,2,3 um:l2_req_miss minimum:500 name:L2_CACHE_MISS : L2 cache misses
event:0x7f counters:0,1,2,3 um:l2_fill minimum:500 name:L2_CACHE_FILL_WRITEBACK : L2 fill/writeback

# Instruction Cache events
event:0x80 counters:0,1,2,3 um:zero minimum:500 name:INSTRUCTION_CACHE_FETCHES : Instruction cache fetches
event:0x81 counters:0,1,2,3 um:zero minimum:500 name:INSTRUCTION_CACHE_MISSES : Instruction cache misses
event:0x82 counters:0,1,2,3 um:zero minimum:500 name:INSTRUCTION_CACHE_REFILLS_FROM_L2 : Instruction cache refills from L2
event:0x83 counters:0,1,2,3 um:zero minimum:500 name:INSTRUCTION_CACHE_REFILLS_FROM_SYSTEM : Instruction cache refills from system
event:0x84 counters:0,1,2,3 um:zero minimum:500 name:L1_ITLB_MISS_AND_L2_ITLB_HIT : L1 ITLB miss and L2 ITLB hit
event:0x85 counters:0,1,2,3 um:zero minimum:500 name:L1_ITLB_MISS_AND_L2_ITLB_MISS : L1 ITLB miss and L2 ITLB miss
event:0x86 counters:0,1,2,3 um:zero minimum:500 name:PIPELINE_RESTART_DUE_TO_INSTRUCTION_STREAM_PROBE : Pipeline restart due to instruction stream probe
event:0x87 counters:0,1,2,3 um:zero minimum:500 name:INSTRUCTION_FETCH_STALL : Instruction fetch stall
event:0x88 counters:0,1,2,3 um:zero minimum:500 name:RETURN_STACK_HITS : Return stack hits
event:0x89 counters:0,1,2,3 um:zero minimum:500 name:RETURN_STACK_OVERFLOWS : Return stack overflows

# Execution Unit events
event:0xc0 counters:0,1,2,3 um:zero minimum:3000 name:RETIRED_INSTRUCTIONS : Retired instructions (includes exceptions, interrupts, re-syncs)
event:0xc1 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_UOPS : Retired micro-ops
event:0xc2 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_BRANCH_INSTRUCTIONS : Retired branches (conditional, unconditional, exceptions, interrupts)
event:0xc3 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_MISPREDICTED_BRANCH_INSTRUCTIONS : Retired mispredicted branch instructions
event:0xc4 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_TAKEN_BRANCH_INSTRUCTIONS : Retired taken branch instructions
event:0xc5 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_TAKEN_BRANCH_INSTRUCTIONS_MISPREDICTED : Retired taken branches mispredicted
event:0xc6 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_FAR_CONTROL_TRANSFERS : Retired far control transfers
event:0xc7 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_BRANCH_RESYNCS : Retired branches resyncs (only non-control transfer branches)
event:0xc8 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_NEAR_RETURNS : Retired near returns
event:0xc9 counters:0,1,2,3 um:zero minimum:500 name:RETIRED_NEAR_RETURNS_MISPREDICTED : Retired near returns mispredicted
event:0xca counters:0,1,2,3 um:zero minimum:500 name:RETIRED_INDIRECT_BRANCHES_MISPREDICTED : Retired indirect branches mispredicted
event:0xcb counters:0,1,2,3 um:fpu_instr minimum:500 name:RETIRED_MMX_FP_INSTRUCTIONS : Retired MMX/FP instructions
event:0xcc counters:0,1,2,3 um:fpu_fastpath minimum:500 name:RETIRED_FASTPATH_DOUBLE_OP_INSTRUCTIONS : Retired FastPath double-op instructions
event:0xcd counters:0,1,2,3 um:zero minimum:500 name:INTERRUPTS_MASKED_CYCLES : Cycles with interrupts masked (IF=0)
event:0xce counters:0,1,2,3 um:zero minimum:500 name:INTERRUPTS_MASKED_CYCLES_WITH_INTERRUPT_PENDING : Cycles with interrupts masked while interrupt pending
event:0xcf counters:0,1,2,3 um:zero minimum:10 name:INTERRUPTS_TAKEN : Number of taken hardware interrupts
event:0xd0 counters:0,1,2,3 um:zero minimum:500 name:DECODER_EMPTY : Nothing to dispatch (decoder empty)
event:0xd1 counters:0,1,2,3 um:zero minimum:500 name:DISPATCH_STALLS : Dispatch stalls
event:0xd2 counters:0,1,2,3 um:zero minimum:500 name:DISPATCH_STALL_FOR_BRANCH_ABORT : Dispatch stall from branch abort to retire
event:0xd3 counters:0,1,2,3 um:zero minimum:500 name:DISPATCH_STALL_FOR_SERIALIZATION : Dispatch stall for serialization
event:0xd4 counters:0,1,2,3 um:zero minimum:500 name:DISPATCH_STALL_FOR_SEGMENT_LOAD : Dispatch stall for segment load
event:0xd5 counters:0,1,2,3 um:zero minimum:500 name:DISPATCH_STALL_FOR_REORDER_BUFFER_FULL : Dispatch stall for reorder buffer full
event:0xd6 counters:0,1,2,3 um:zero minimum:500 name:DISPATCH_STALL_FOR_RESERVATION_STATION_FULL : Dispatch stall when reservation stations are full
event:0xd7 counters:0,1,2,3 um:zero minimum:500 name:DISPATCH_STALL_FOR_FPU_FULL : Dispatch stall when FPU is full
event:0xd8 counters:0,1,2,3 um:zero minimum:500 name:DISPATCH_STALL_FOR_LS_FULL : Dispatch stall when LS is full
event:0xd9 counters:0,1,2,3 um:zero minimum:500 name:DISPATCH_STALL_WAITING_FOR_ALL_QUIET : Dispatch stall when waiting for all to be quiet
event:0xda counters:0,1,2,3 um:zero minimum:500 name:DISPATCH_STALL_FOR_FAR_TRANSFER_OR_RESYNC : Dispatch stall for far transfer or resync to retire
event:0xdb counters:0,1,2,3 um:fpu_exceptions minimum:1 name:FPU_EXCEPTIONS : FPU exceptions
event:0xdc counters:0,1,2,3 um:zero minimum:1 name:DR0_BREAKPOINTS : Number of breakpoints for DR0
event:0xdd counters:0,1,2,3 um:zero minimum:1 name:DR1_BREAKPOINTS : Number of breakpoints for DR1
event:0xde counters:0,1,2,3 um:zero minimum:1 name:DR2_BREAKPOINTS : Number of breakpoints for DR2
event:0xdf counters:0,1,2,3 um:zero minimum:1 name:DR3_BREAKPOINTS : Number of breakpoints for DR3

# Memory Controler events
event:0xe0 counters:0,1,2,3 um:page_access minimum:500 name:DRAM_ACCESSES : DRAM accesses
event:0xe1 counters:0,1,2,3 um:zero minimum:500 name:MEMORY_CONTROLLER_PAGE_TABLE_OVERFLOWS : Memory controller page table overflows
event:0xe3 counters:0,1,2,3 um:turnaround minimum:500 name:MEMORY_CONTROLLER_TURNAROUNDS : Memory controller turnarounds
event:0xe4 counters:0,1,2,3 um:saturation minimum:500 name:MEMORY_CONTROLLER_BYPASS_COUNTER_SATURATION : Memory controller bypass saturation
event:0xe5 counters:0,1,2,3 um:sizedblocks minimum:500 name:SIZED_BLOCKS : Sized blocks
event:0xe8 counters:0,1,2,3 um:thermalecc minimum:500 name:THERMAL_STATUS_AND_DRAM_ECC_ERRORS : Thermal status and ECC errors
event:0xe9 counters:0,1,2,3 um:cpiorequests minimum:500 name:CPU_IO_REQUESTS_TO_MEMORY_IO : CPU/IO requests to memory/IO (RevE)
event:0xea counters:0,1,2,3 um:cacheblock minimum:500 name:CACHE_BLOCK_COMMANDS : Cache block commands (RevE)
event:0xeb counters:0,1,2,3 um:sizecmds minimum:500 name:SIZED_COMMANDS : Sized commands
event:0xec counters:0,1,2,3 um:probe minimum:500 name:PROBE_RESPONSES_AND_UPSTREAM_REQUESTS : Probe responses and upstream requests
event:0xee counters:0,1,2,3 um:gart minimum:500 name:GART_EVENTS : GART events

# Link events
event:0xf6 counters:0,1,2,3 um:ht minimum:500 name:HYPERTRANSPORT_LINK0_BANDWIDTH : HyperTransport(tm) link 0 transmit bandwidth
event:0xf7 counters:0,1,2,3 um:ht minimum:500 name:HYPERTRANSPORT_LINK1_BANDWIDTH : HyperTransport(tm) link 1 transmit bandwidth
event:0xf8 counters:0,1,2,3 um:ht minimum:500 name:HYPERTRANSPORT_LINK2_BANDWIDTH : HyperTransport(tm) link 2 transmit bandwidth
[djg11@koo oprofile-0.9.9]$ 
```

New Power Supply Monitors



Resistive shunt measurement

Measurement using switched-mode (SMPSU) cycle counting measurement

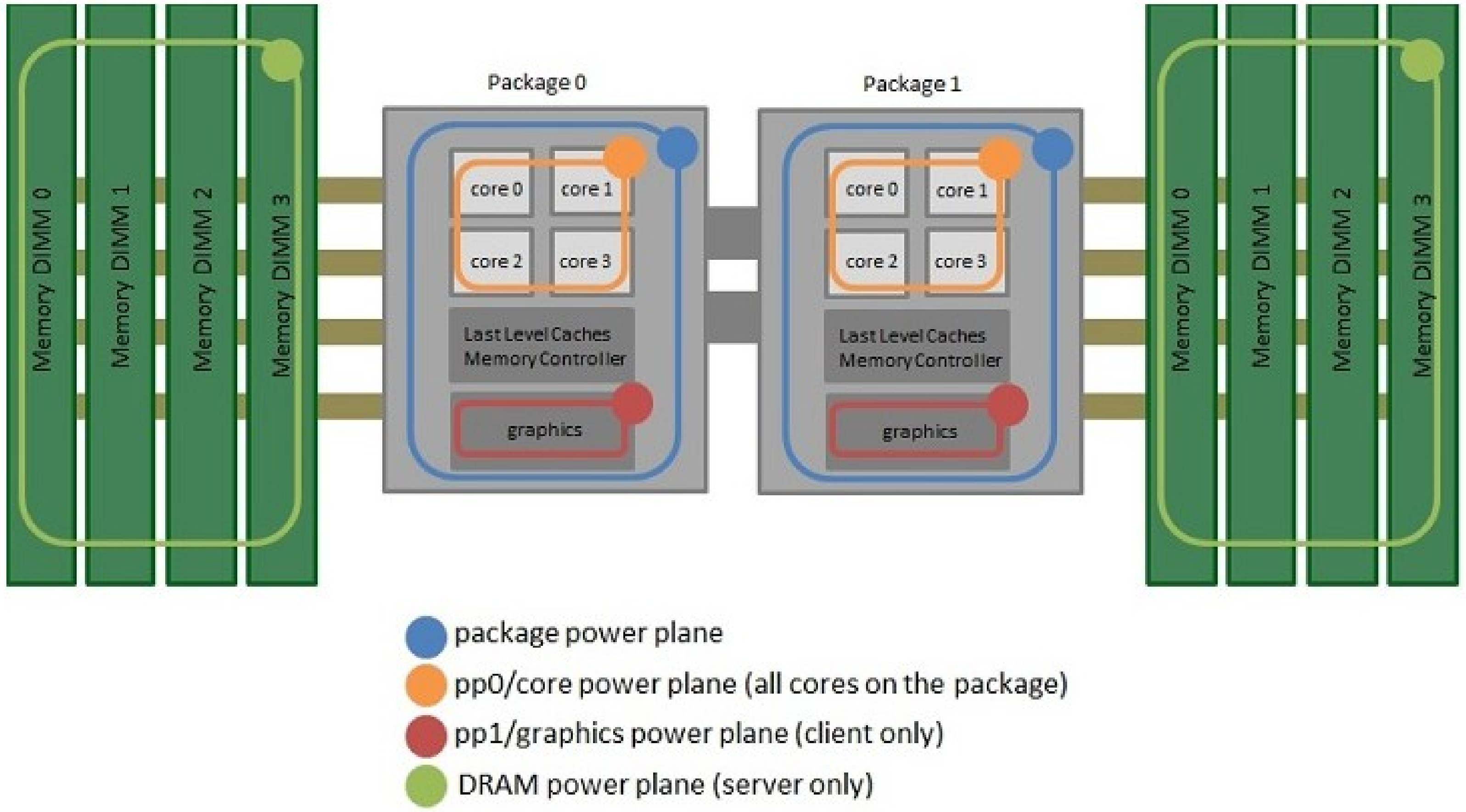
Intel's Power Gadget MSRs

Intel has implemented a Running Average Power Limit (RAPL) on Sandybridge processors.

A number of machine-specific registers are defined containing energy information:

```
SandyBridge:
    MSR_RAPL_POWER_UNIT   MSR_PKG_POWER_LIMIT    MSR_PKG_ENERGY_STATUS   MSR_PP0_POLICY
    MSR_PP0_PERF_STATUS   MSR_PKG_POWER_INFO    MSR_PP0_POWER_LIMIT    MSR_PP0_ENERGY_STATUS
```

»Measuring Energy Consumption for Short Code Paths Using RAPL. Hähnel 2012



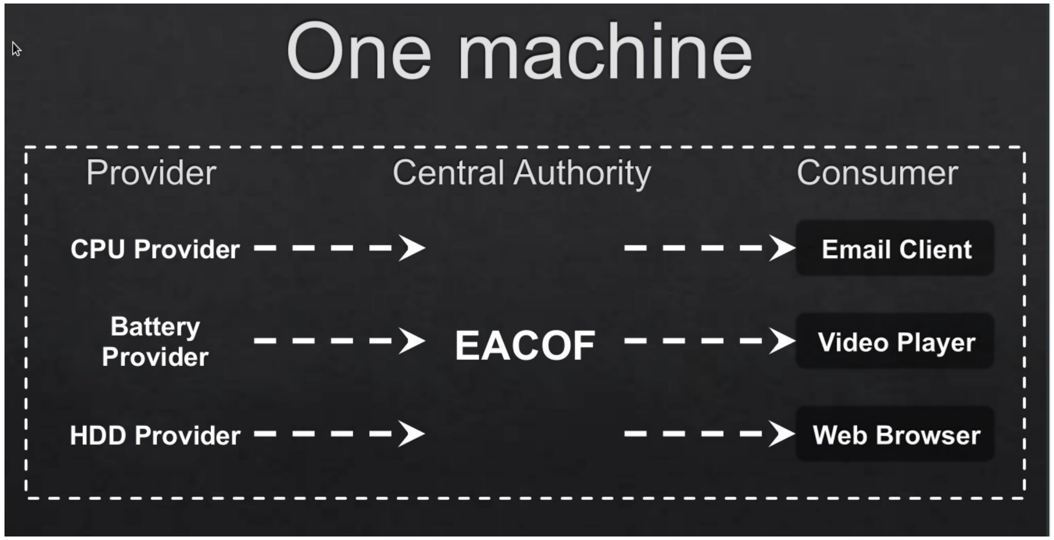
Energy Aware COmputing Framework (EACOF)

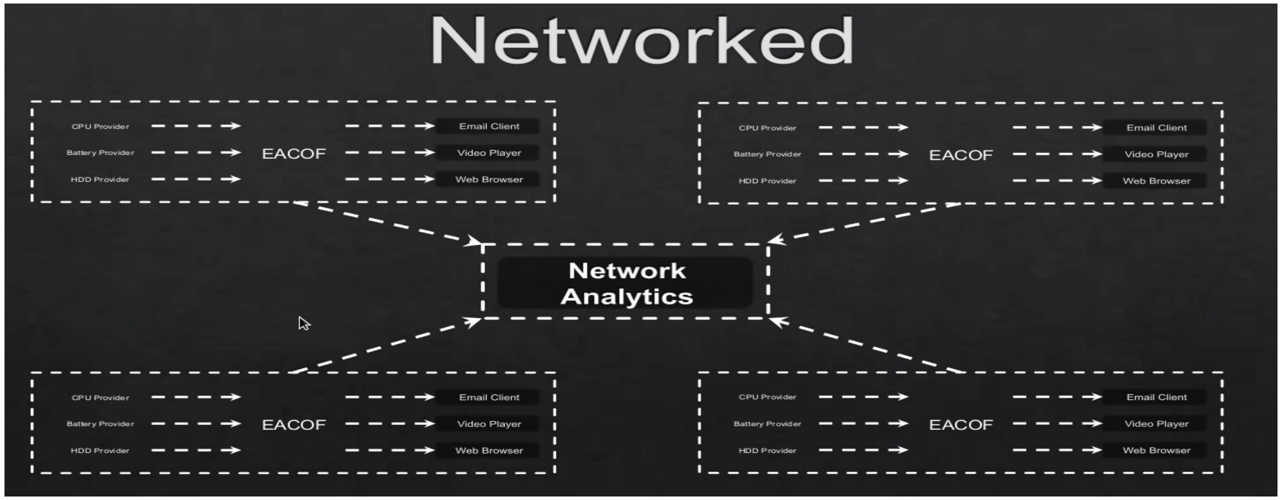
Hayden Field / James Pedlingham – University of Bristol

Basically an SQL networked server where:

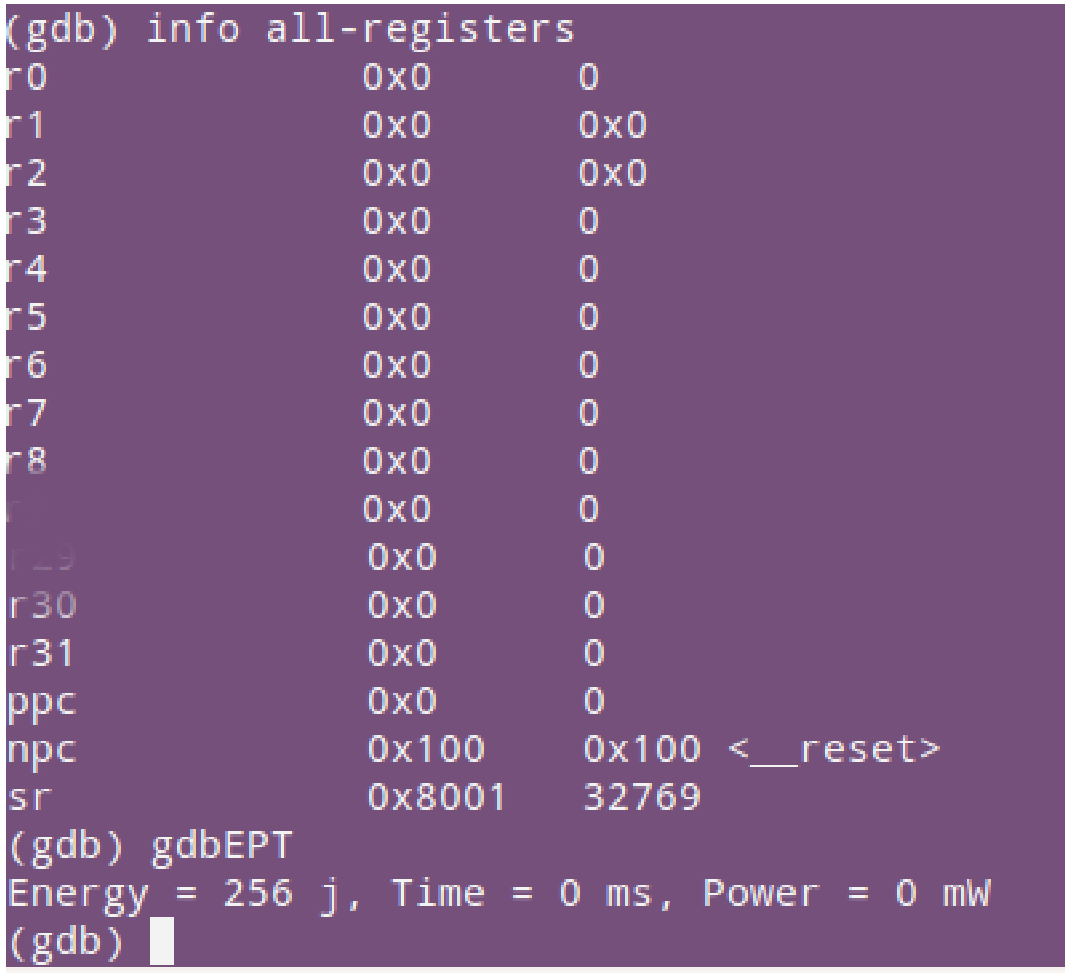
- Multiple sensors and other providers can log energy use

- Multiple customers and analytics can inspect.





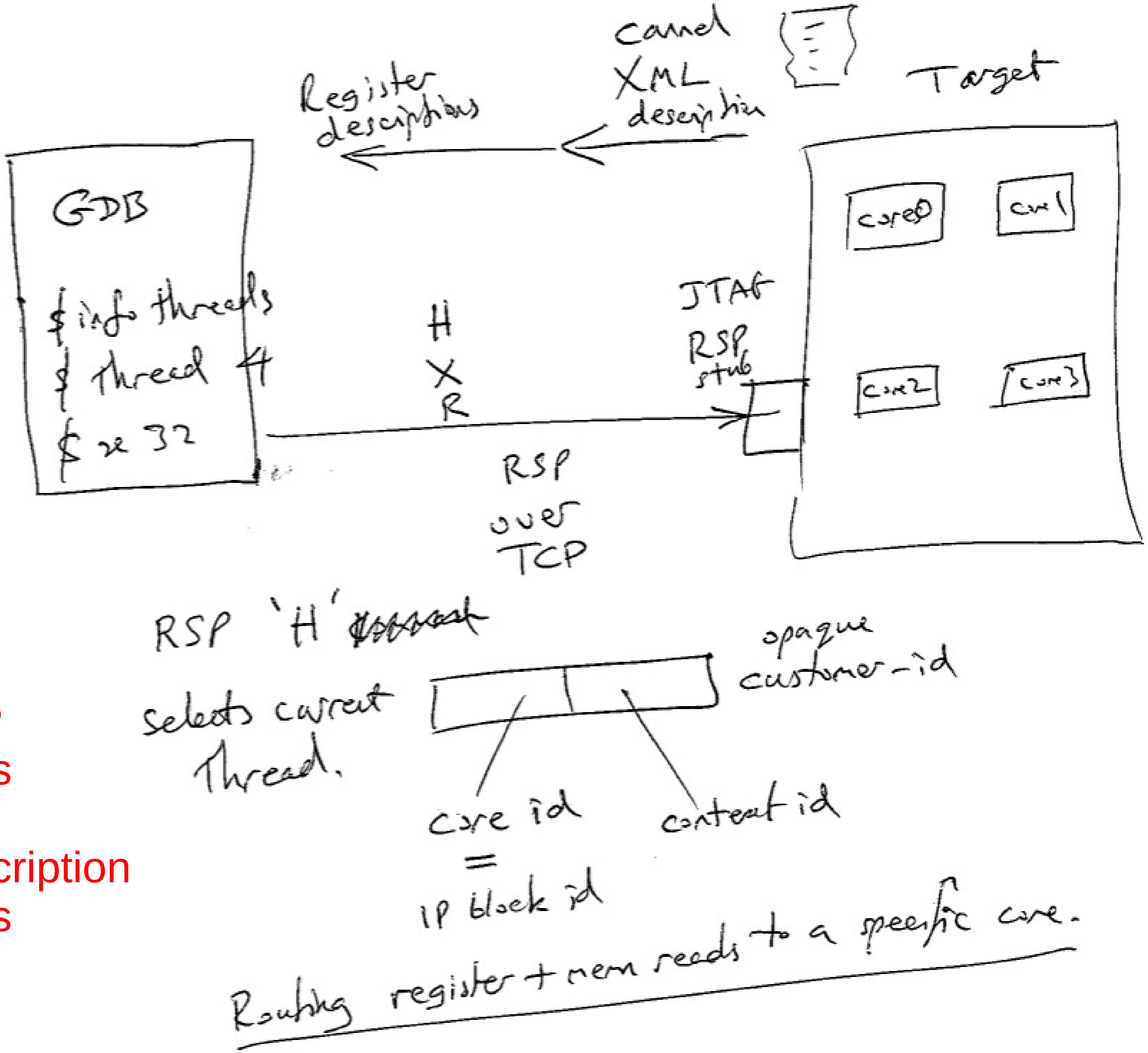
Existing GDB Energy Capability ...



```
(gdb) info all-registers
r0             0x0      0
r1             0x0      0x0
r2             0x0      0x0
r3             0x0      0
r4             0x0      0
r5             0x0      0
r6             0x0      0
r7             0x0      0
r8             0x0      0
r              0x0      0
r29            0x0      0
r30            0x0      0
r31            0x0      0
ppc            0x0      0
npc            0x100    0x100 <__reset>
sr             0x8001   32769
(gdb) gdbEPT
Energy = 256 j, Time = 0 ms, Power = 0 mW
(gdb)
```

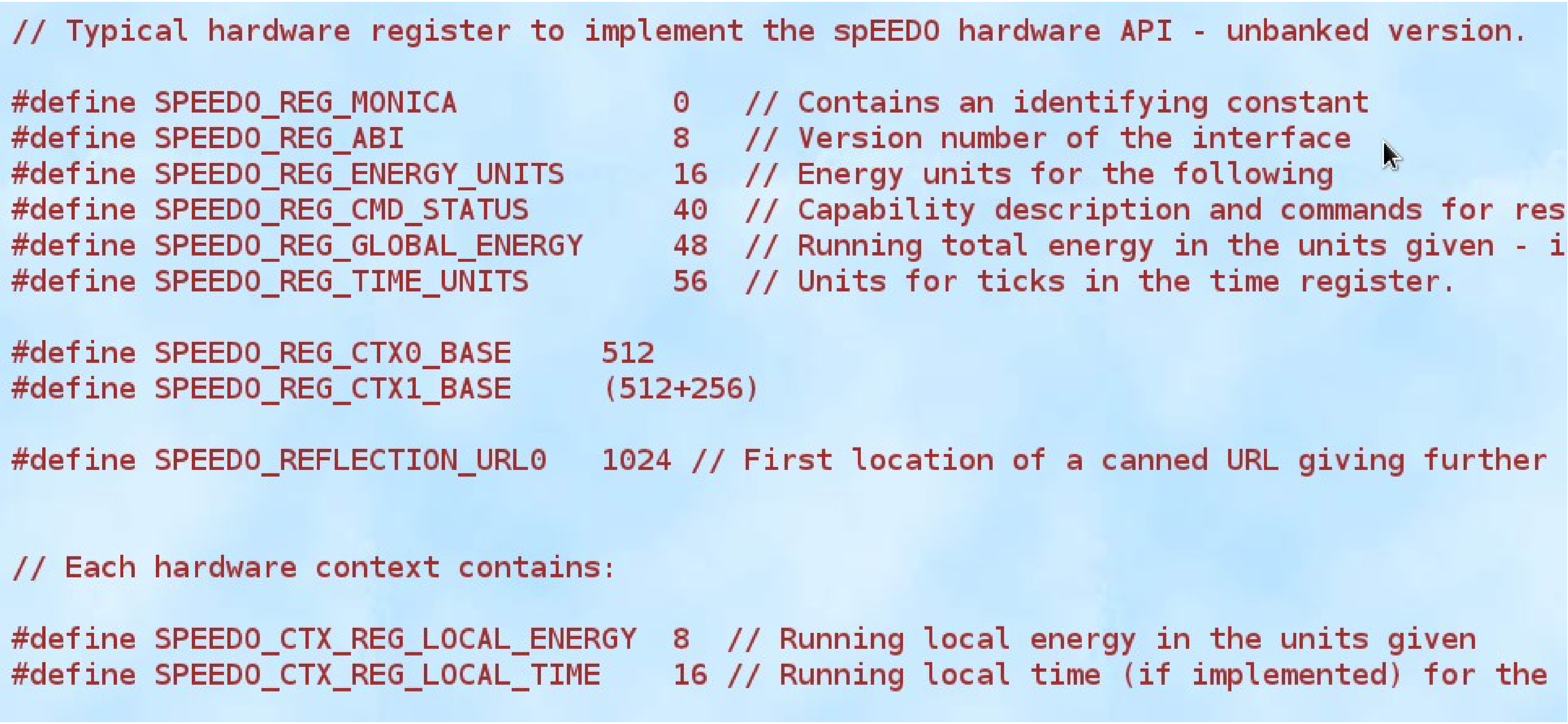
...

is inadequate !



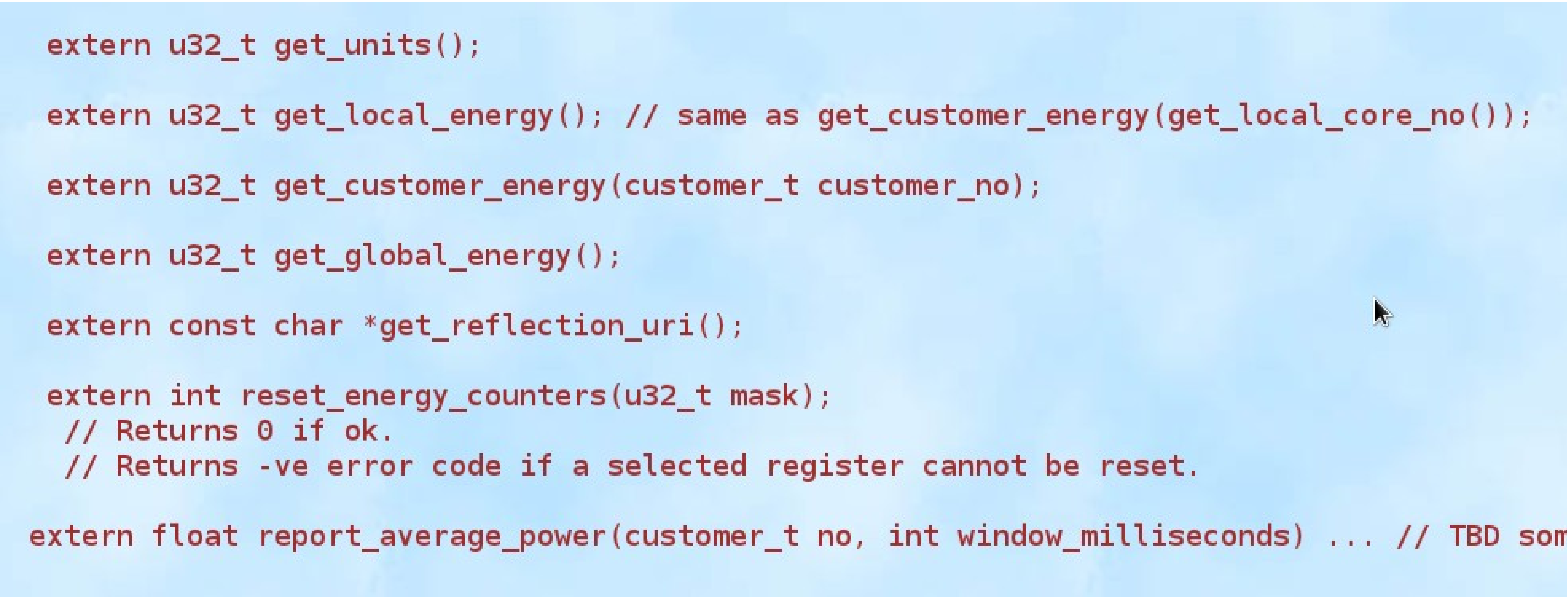
GDB RSP Extensions & XML Target Description Extensions

Register Power ABI Strawman



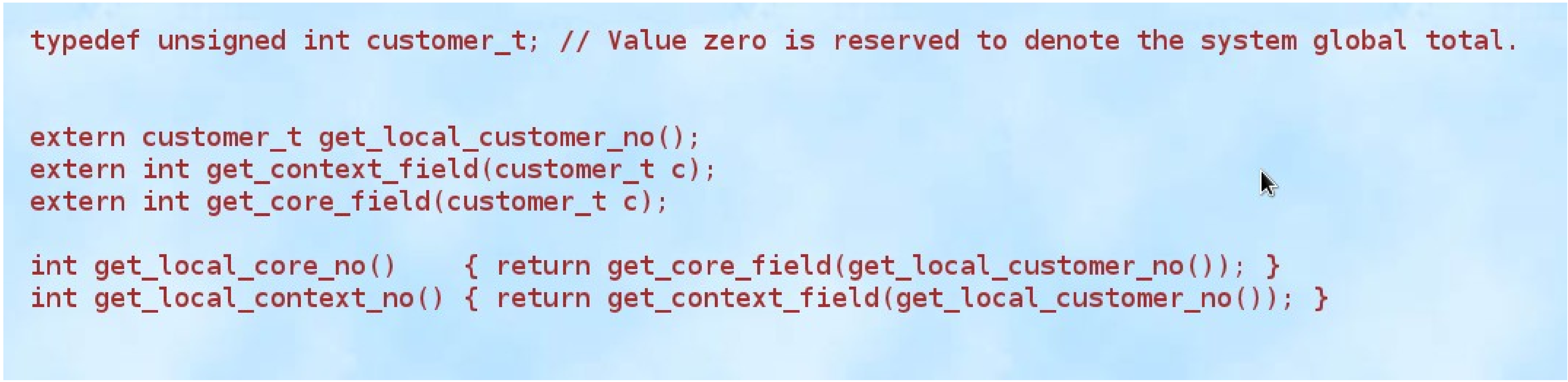
```
// Typical hardware register to implement the spEEDO hardware API - unbanked version.

#define SPEEDO_REG_MONICA              0   // Contains an identifying constant
#define SPEEDO_REG_ABI                 8   // Version number of the interface
#define SPEEDO_REG_ENERGY_UNITS        16  // Energy units for the following
#define SPEEDO_REG_CMD_STATUS          40  // Capability description and commands for res
#define SPEEDO_REG_GLOBAL_ENERGY       48  // Running total energy in the units given - i
#define SPEEDO_REG_TIME_UNITS          56  // Units for ticks in the time register.

#define SPEEDO_REG_CTX0_BASE     512
#define SPEEDO_REG_CTX1_BASE     (512+256)

#define SPEEDO_REFLECTION_URL0   1024 // First location of a canned URL giving further


// Each hardware context contains:

#define SPEEDO_CTX_REG_LOCAL_ENERGY  8  // Running local energy in the units given
#define SPEEDO_CTX_REG_LOCAL_TIME    16 // Running local time (if implemented) for the
```

C API – Registers via HAL



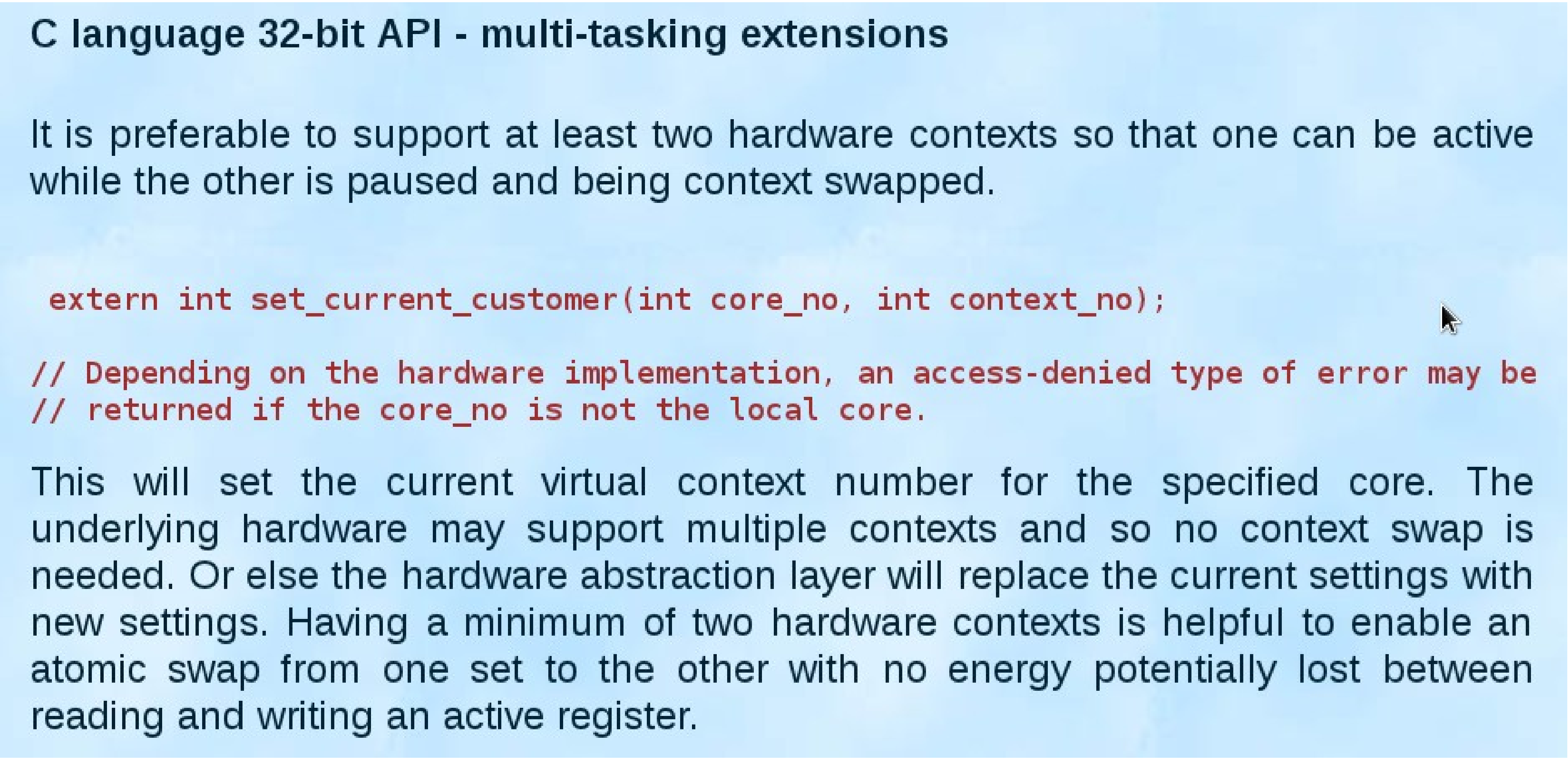
```
 extern u32_t get_units();

 extern u32_t get_local_energy(); // same as get_customer_energy(get_local_core_no());

 extern u32_t get_customer_energy(customer_t customer_no);

 extern u32_t get_global_energy();

 extern const char *get_reflection_uri();

 extern int reset_energy_counters(u32_t mask);
  // Returns 0 if ok.
  // Returns -ve error code if a selected register cannot be reset.

extern float report_average_power(customer_t no, int window_milliseconds) ... // TBD som
```

Customer Number



```
typedef unsigned int customer_t; // Value zero is reserved to denote the system global total.


extern customer_t get_local_customer_no();
extern int get_context_field(customer_t c);
extern int get_core_field(customer_t c);

int get_local_core_no()    { return get_core_field(get_local_customer_no()); }
int get_local_context_no() { return get_context_field(get_local_customer_no()); }
```

Context Swap H/W Energy Bank



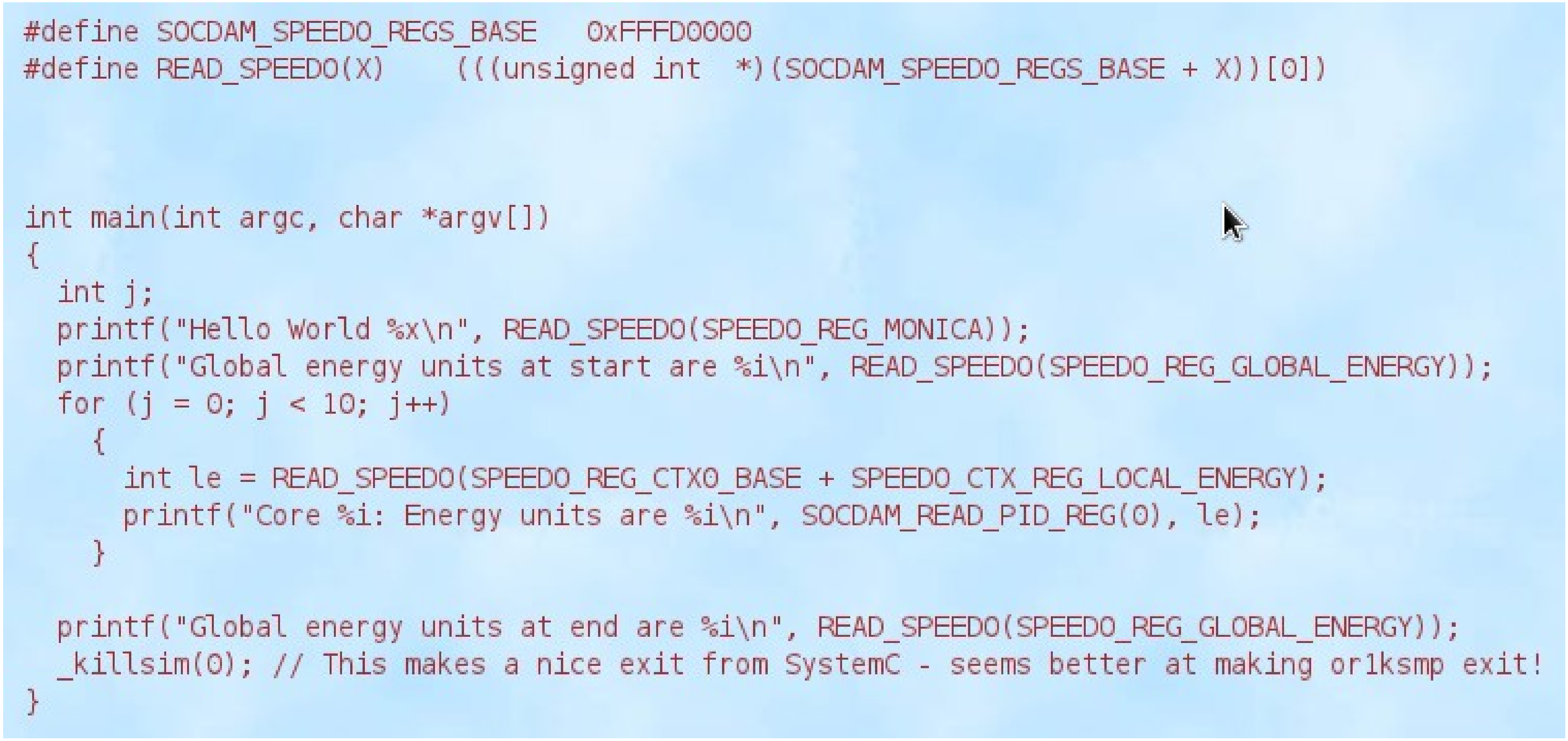
C language 32-bit API - multi-tasking extensions

It is preferable to support at least two hardware contexts so that one can be active while the other is paused and being context swapped.

```
 extern int set_current_customer(int core_no, int context_no);

// Depending on the hardware implementation, an access-denied type of error may be
// returned if the core_no is not the local core.
```

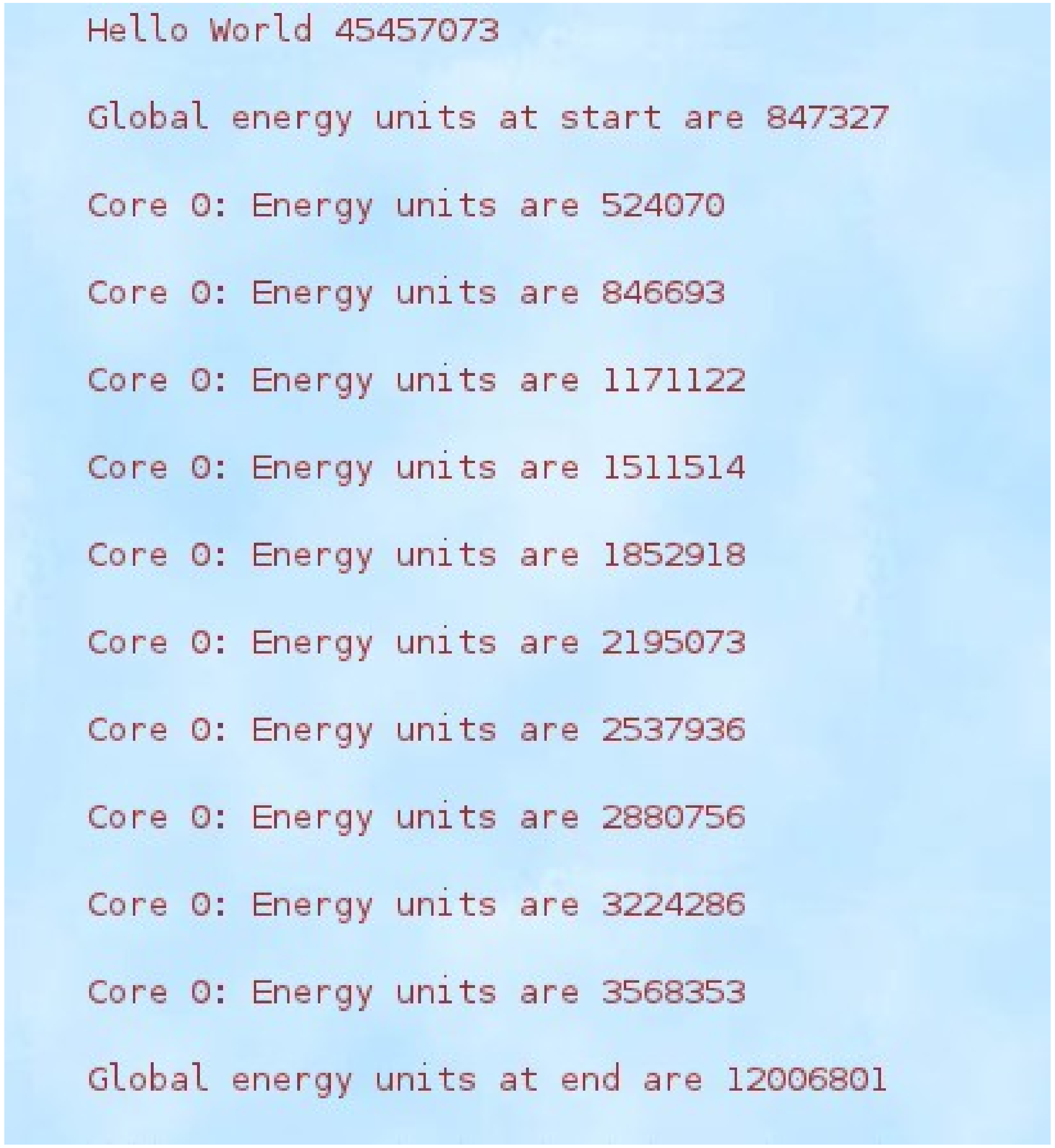
This will set the current virtual context number for the specified core. The underlying hardware may support multiple contexts and so no context swap is needed. Or else the hardware abstraction layer will replace the current settings with new settings. Having a minimum of two hardware contexts is helpful to enable an atomic swap from one set to the other with no energy potentially lost between reading and writing an active register.

A Hello World, very-simple C app.



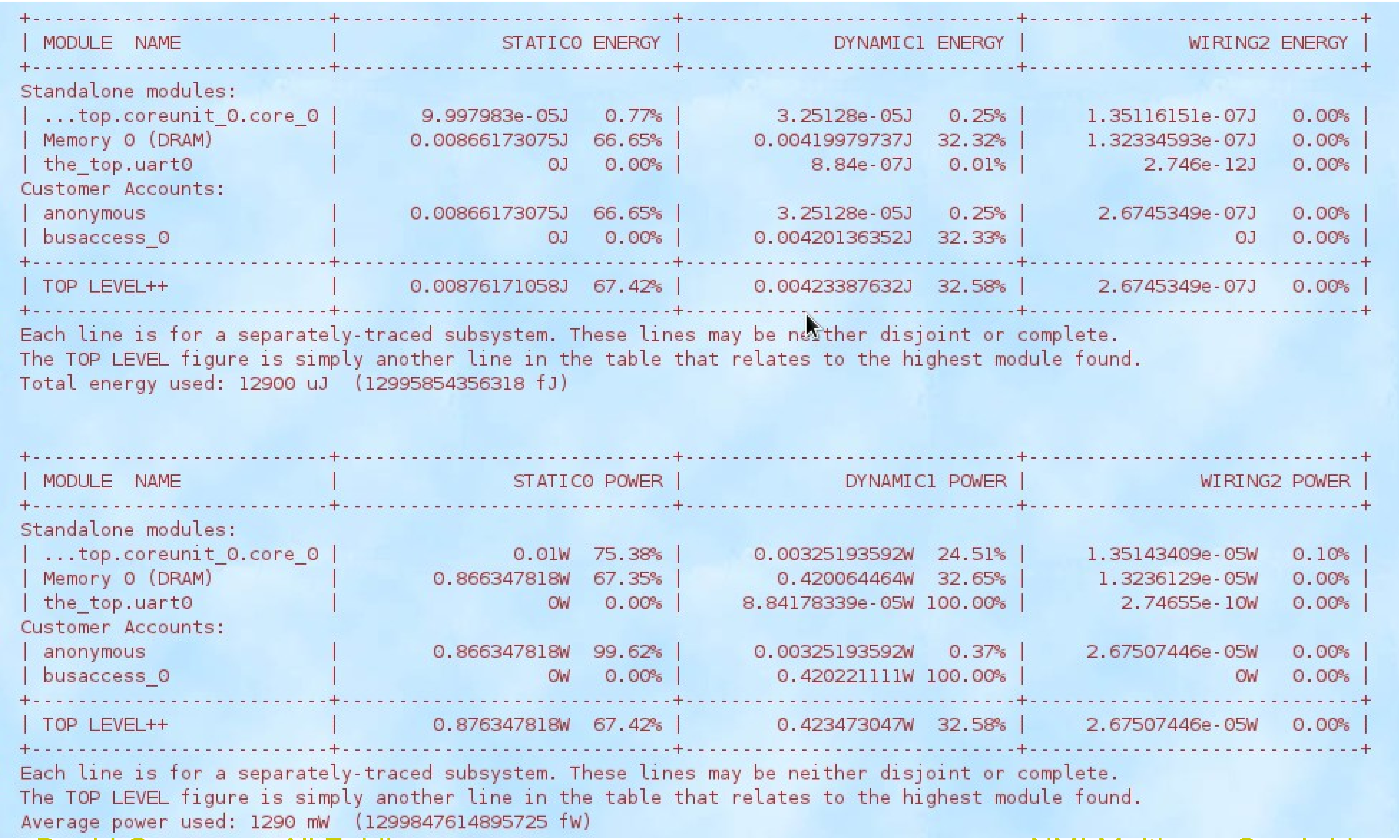
```
#define SOCDAM_SPEEDO_REGS_BASE   0xFFFD0000
#define READ_SPEEDO(X)    (((unsigned int  *)(SOCDAM_SPEEDO_REGS_BASE + X))[0])



int main(int argc, char *argv[])
{
  int j;
  printf("Hello World %x\n", READ_SPEEDO(SPEEDO_REG_MONICA));
  printf("Global energy units at start are %i\n", READ_SPEEDO(SPEEDO_REG_GLOBAL_ENERGY));
  for (j = 0; j < 10; j++)
    {
      int le = READ_SPEEDO(SPEEDO_REG_CTX0_BASE + SPEEDO_CTX_REG_LOCAL_ENERGY);
      printf("Core %i: Energy units are %i\n", SOCDAM_READ_PID_REG(0), le);
    }

  printf("Global energy units at end are %i\n", READ_SPEEDO(SPEEDO_REG_GLOBAL_ENERGY));
  _killsim(0); // This makes a nice exit from SystemC - seems better at making or1ksmp exit!
}
```

Output from the very-simple C Program



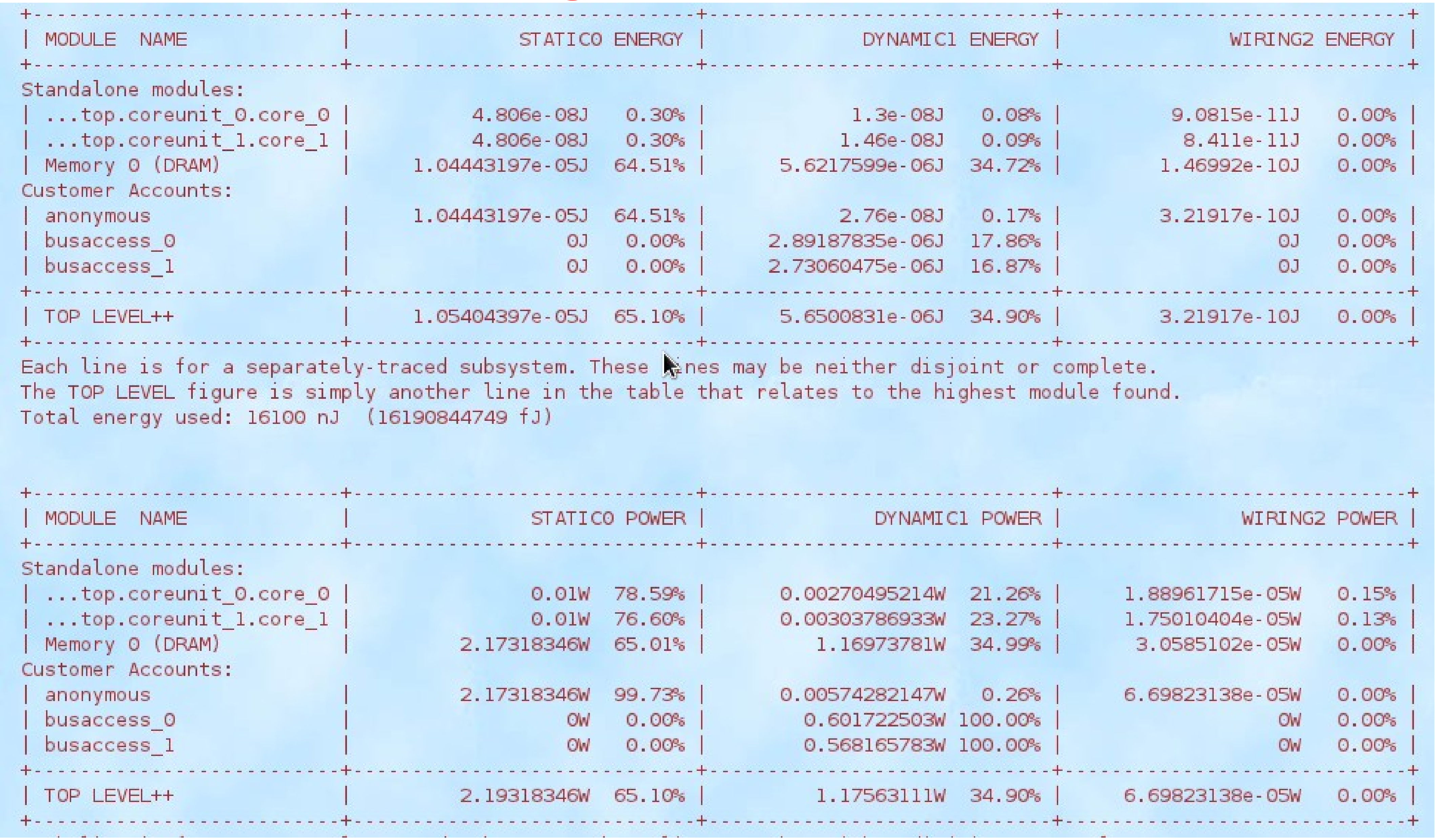
```
Hello World 45457073

Global energy units at start are 847327

Core 0: Energy units are 524070

Core 0: Energy units are 846693

Core 0: Energy units are 1171122

Core 0: Energy units are 1511514

Core 0: Energy units are 1852918

Core 0: Energy units are 2195073

Core 0: Energy units are 2537936

Core 0: Energy units are 2880756

Core 0: Energy units are 3224286

Core 0: Energy units are 3568353

Global energy units at end are 12006801
```

Energy Report With Customer Nos



```
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
| MODULE  NAME              |              STATIC0 ENERGY |             DYNAMIC1 ENERGY |              WIRING2 ENERGY |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
Standalone modules:
| ...top.coreunit_0.core_0 |       9.997983e-05J   0.77% |        3.25128e-05J   0.25% |     1.35116151e-07J   0.00% |
| Memory 0 (DRAM)          |     0.00866173075J  66.65% |      0.00419979737J  32.32% |     1.32334593e-07J   0.00% |
| the_top.uart0            |                 0J   0.00% |           8.84e-07J   0.01% |          2.746e-12J   0.00% |
Customer Accounts:
| anonymous                |     0.00866173075J  66.65% |        3.25128e-05J   0.25% |      2.6745349e-07J   0.00% |
| busaccess_0              |                 0J   0.00% |      0.00420136352J  32.33% |                 0J   0.00% |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
| TOP LEVEL++              |     0.00876171058J  67.42% |      0.00423387632J  32.58% |      2.6745349e-07J   0.00% |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
Each line is for a separately-traced subsystem. These lines may be neither disjoint or complete.
The TOP LEVEL figure is simply another line in the table that relates to the highest module found.
Total energy used: 12900 uJ  (12995854356318 fJ)


+---------------------------+-----------------------------+-----------------------------+-----------------------------+
| MODULE  NAME              |               STATIC0 POWER |              DYNAMIC1 POWER |               WIRING2 POWER |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
Standalone modules:
| ...top.coreunit_0.core_0 |             0.01W  75.38% |      0.00325193592W  24.51% |     1.35143409e-05W   0.10% |
| Memory 0 (DRAM)          |       0.866347818W  67.35% |        0.420064464W  32.65% |      1.3236129e-05W   0.00% |
| the_top.uart0            |                0W   0.00% |    8.84178339e-05W 100.00% |        2.74655e-10W   0.00% |
Customer Accounts:
| anonymous                |       0.866347818W  99.62% |      0.00325193592W   0.37% |     2.67507446e-05W   0.00% |
| busaccess_0              |                0W   0.00% |        0.420221111W 100.00% |                 0W   0.00% |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
| TOP LEVEL++              |       0.876347818W  67.42% |        0.423473047W  32.58% |     2.67507446e-05W   0.00% |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
Each line is for a separately-traced subsystem. These lines may be neither disjoint or complete.
The TOP LEVEL figure is simply another line in the table that relates to the highest module found.
Average power used: 1290 mW  (1299847614895725 fW)
```

Running on two cores...



```
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
| MODULE  NAME              |              STATIC0 ENERGY |             DYNAMIC1 ENERGY |               WIRING2 ENERGY |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
Standalone modules:
| ...top.coreunit_0.core_0 |        4.806e-08J    0.30% |           1.3e-08J    0.08% |        9.0815e-11J    0.00% |
| ...top.coreunit_1.core_1 |        4.806e-08J    0.30% |          1.46e-08J    0.09% |         8.411e-11J    0.00% |
| Memory 0 (DRAM)          |   1.04443197e-05J   64.51% |    5.6217599e-06J   34.72% |       1.46992e-10J    0.00% |
Customer Accounts:
| anonymous                |   1.04443197e-05J   64.51% |          2.76e-08J    0.17% |       3.21917e-10J    0.00% |
| busaccess_0              |               0J    0.00% |  2.89187835e-06J   17.86% |                0J    0.00% |
| busaccess_1              |               0J    0.00% |  2.73060475e-06J   16.87% |                0J    0.00% |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
| TOP LEVEL++              |   1.05404397e-05J   65.10% |    5.6500831e-06J   34.90% |       3.21917e-10J    0.00% |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
Each line is for a separately-traced subsystem. These lines may be neither disjoint or complete.
The TOP LEVEL figure is simply another line in the table that relates to the highest module found.
Total energy used: 16100 nJ  (16190844749 fJ)


+---------------------------+-----------------------------+-----------------------------+-----------------------------+
| MODULE  NAME              |               STATIC0 POWER |              DYNAMIC1 POWER |                WIRING2 POWER |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
Standalone modules:
| ...top.coreunit_0.core_0 |            0.01W   78.59% |    0.00270495214W   21.26% |    1.88961715e-05W    0.15% |
| ...top.coreunit_1.core_1 |            0.01W   76.60% |    0.00303786933W   23.27% |    1.75010404e-05W    0.13% |
| Memory 0 (DRAM)          |      2.17318346W   65.01% |       1.16973781W   34.99% |     3.0585102e-05W    0.00% |
Customer Accounts:
| anonymous                |      2.17318346W   99.73% |    0.00574282147W    0.26% |    6.69823138e-05W    0.00% |
| busaccess_0              |              0W    0.00% |      0.601722503W  100.00% |                0W    0.00% |
| busaccess_1              |              0W    0.00% |      0.568165783W  100.00% |                0W    0.00% |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
| TOP LEVEL++              |      2.19318346W   65.10% |       1.17563111W   34.90% |    6.69823138e-05W    0.00% |
+---------------------------+-----------------------------+-----------------------------+-----------------------------+
```

Thankyou for listening

David Greaves
Ali Zaidi
Klaus McDonald Maier

University of Cambridge
Computer Laboratory

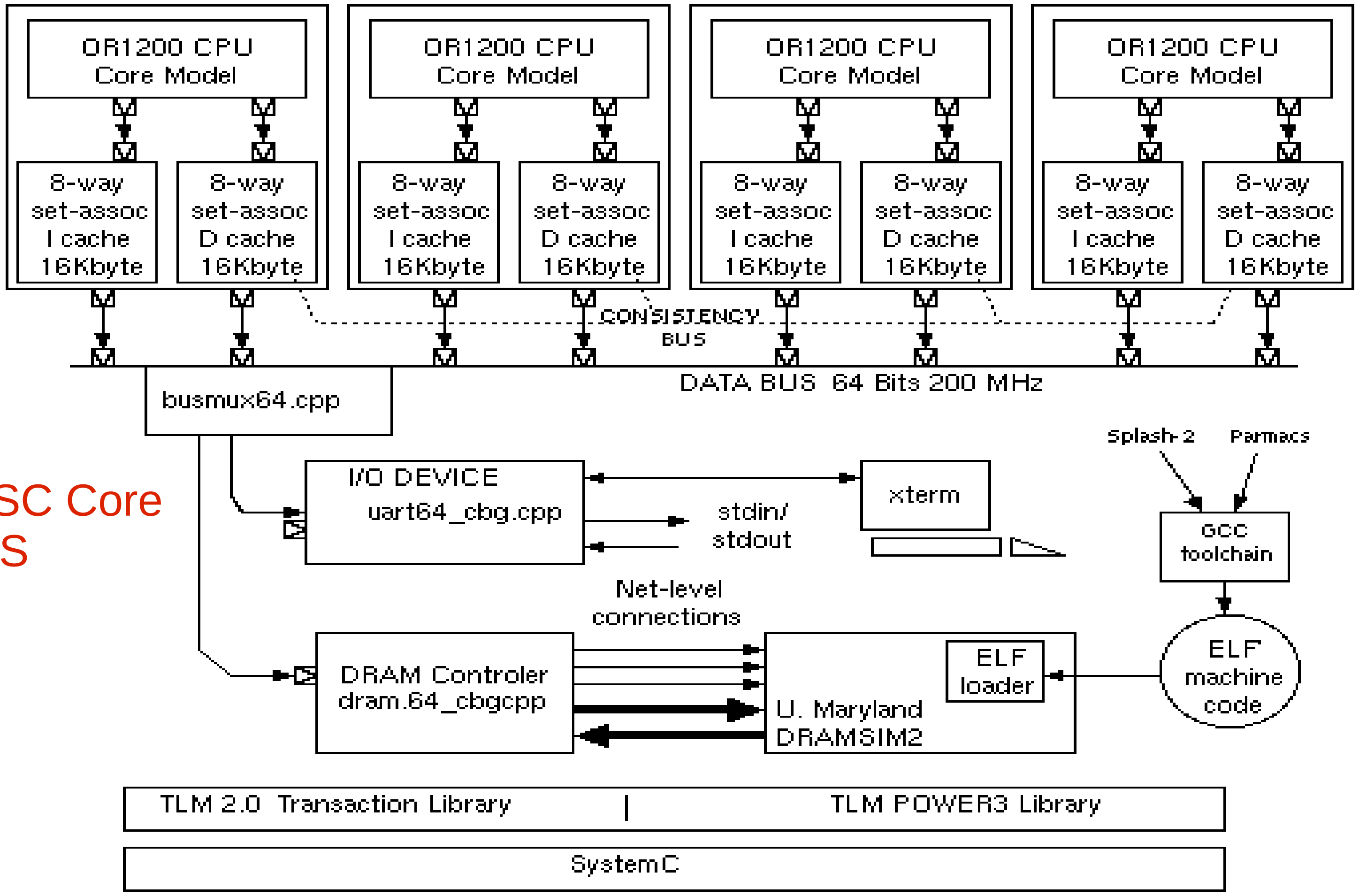
FOSDEM'14
Energy Efficient Computing.

BACKUP SLIDES NOW FOLLOW

....

TLM Modelling and TLM POWER 3

SMP OpenRISC Demo Platform



Verilated OpenRISC Core
Or fast ORSIM ISS
(Or MIPS64)

1 to 64 cores (four shown)
Shared or split or no L1 Cache
Flexible cache architectures
L2 and L3 caches easily added

Each cache has power-annotated tag and data RAMs
SRAM parameters from CACTI
DRAM modelled by Univ Maryland DRAMSIM2

SystemC

A free C++ library that provides:

- A hardware module description system where a module is a C++ class,
- An eventing and threading kernel,
- Compute/commit signals as well as other forms of channel,
- A library of fixed-precision integers,
- Plotting and logging facilities for generating output,
- Two transactional modelling libraries.

Originally aimed as an RTL replacement, for low-level hardware modelling.

Now being used for high-level (esp. transactional) modelling for architectural exploration.

Also now being used as an implementation language with its own synthesis tools.

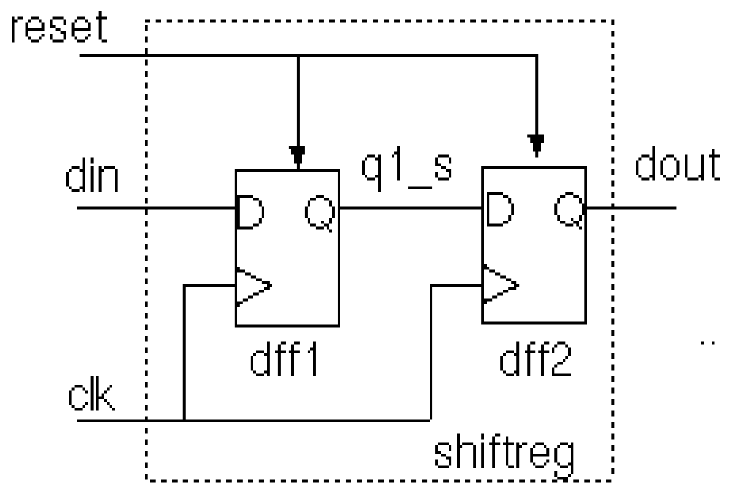
SystemC: Example Module

In this example a C++ class is defined using the the SC_MODULE macro.

```
SC_MODULE(mycounter)
{
   sc_in  < bool       > clk, enable, reset;
   sc_out < sc_int<10> > sum;

   void m() // Behaviour
   {
      if (reset) sum = 0;
      else if (enable) sum = sum.read()+1;
      // Use .read() since sc_out makes a signal.
   }

   SC_CTOR(mycounter) // constructor
     { SC_METHOD(m);
       sensitive << clk.pos();
     }
}
```

Modules inherit various attributes appropriate for an hierarchic hardware design including an instance name field and a channel binding capability.

SystemC: Structural Netlist



```
// Example of structural hierarchy and wiring
// between levels:

SC_MODULE(shiftreg)  // Two-bit shift register
{   sc_in  < bool >  clk, reset, din;
    sc_out < bool >  dout;

    sc_signal < bool > q1_s;
    dff dff1, dff2;        // Instantiate FFs

    SC_CTOR(shiftreg) :
                 dff1("dff1"), dff2("dff2")
    {   dff1.clk(clk);
        dff1.reset(reset);
        dff1.d(din);
        dff1.q(q1_s);

        dff2.clk(clk);
        dff2.reset(reset);
        dff2.d(q1_s);
        dff2.q(dout);
    }
};
```

The sc_signal (extends sc_channel) should be used to obtain the compute/commit paradigm. Avoids non-determinacy from races on zero-delay flip-flops.

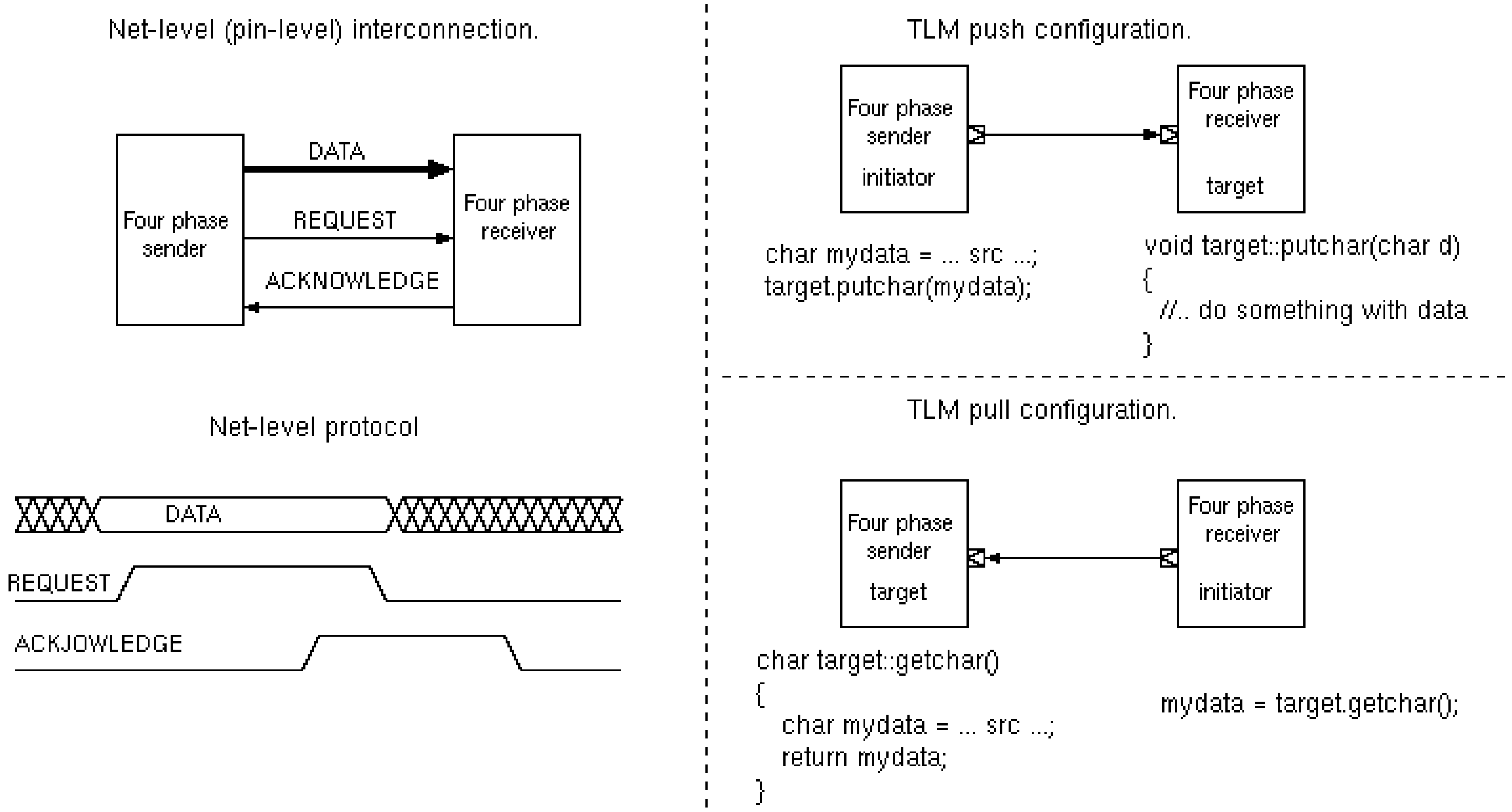
sc_in and sc_out extend sc_channel.

General SystemC channel provides general purpose interface between components.

Other SystemC channel types include FIFOs and semaphores.

sc_port and sc_export needed for TLM modelling.

Transaction Level Modelling



```
char mydata = ... src ...;
target.putchar(mydata);
```

```
void target::putchar(char d)
{
  //.. do something with data
}
```

```
char target::getchar()
{
   char mydata = ... src ...;
   return mydata;
}
```

```
mydata = target.getchar();
```

Note that the roles of initiator and target do not necessarily relate to the sources and sinks of the data.

Infact, an initiator can commonly make both a read and a write transaction on a given target and so the direction of data transfer is dynamic.

TLM: Loose Timing

Naive Coding Style

```
b_putbyte(char d)
{
  printf(“Byte '%c'\n”, d);
  wait(250, SC_NS);
}
```

Loosely-Timed Coding Style

Have a local variable 'delay' associated with each thread.

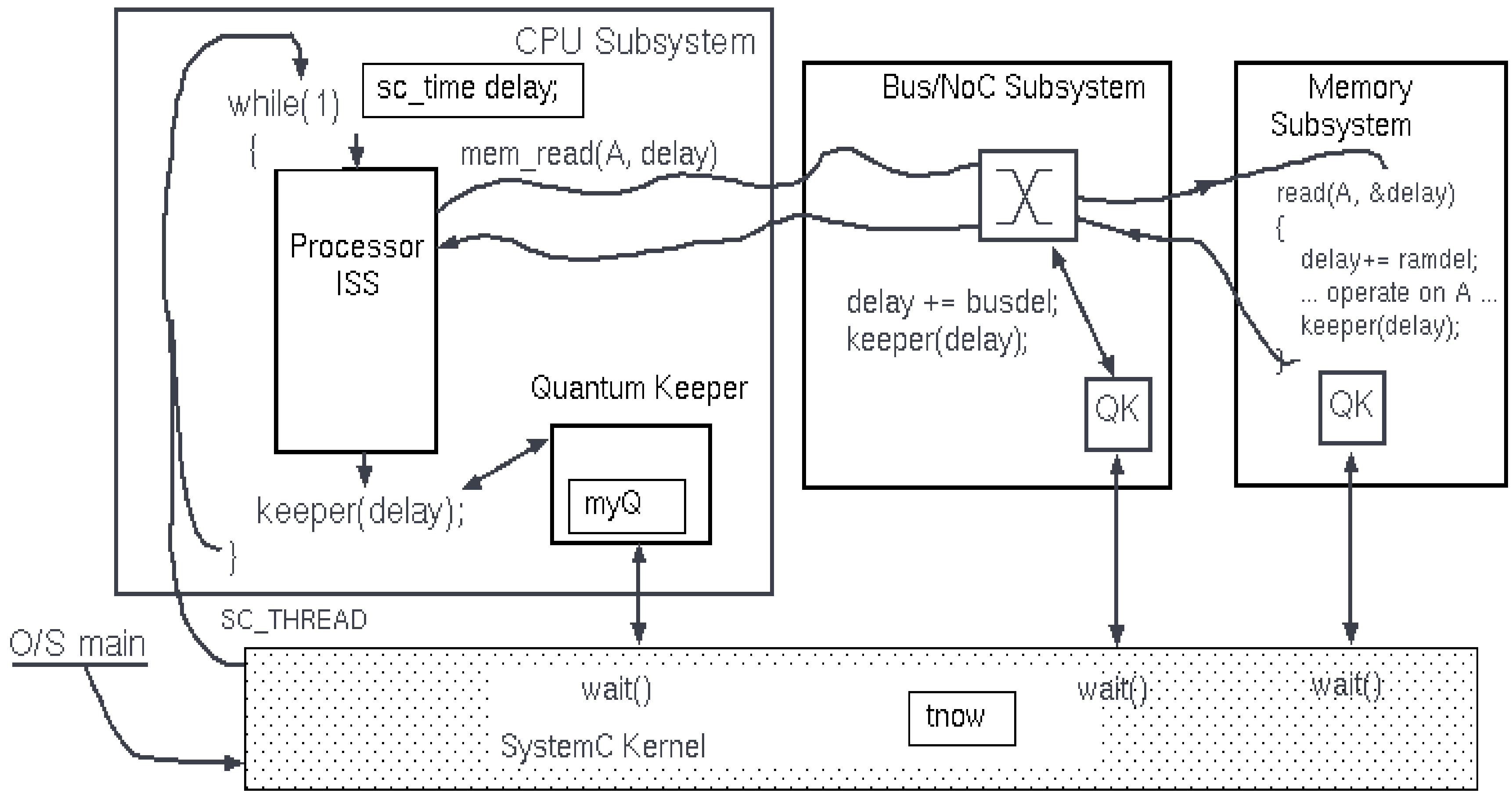
```
b_putbyte(char d, sc_time &delay)
{
  sc_time del(250, SC_NS);
  printf(“Byte '%c'\n”, d);
  delay += del;
}
```

But, at any point, any thread can resynch itself with the kernel by performing:

```
// Resynch idiomatic form:
sc_wait(delay);
Delay = 0;
```

Simulation performance is reduced when there are frequent resynchs, but true transaction ordering will be modelled correctly.

Loosely-timed TLM Modelling: General Structure



Records, Accounts and Observers

- Every monitored module is tied to a *power record*
  - by inheritance or
  - by SystemC attribute.
- Every power record contains a set of accounts.
- Accounts have common (user-defined) names and purposes across the system. Typically:
  - A1 Static power
  - A2 Dynamic energy
  - A3 Wiring energy
- Each account can track both energy and power.
- An *observer* sums activity in a collection of records keeping accounts separate.
- A report file has one observer per line.

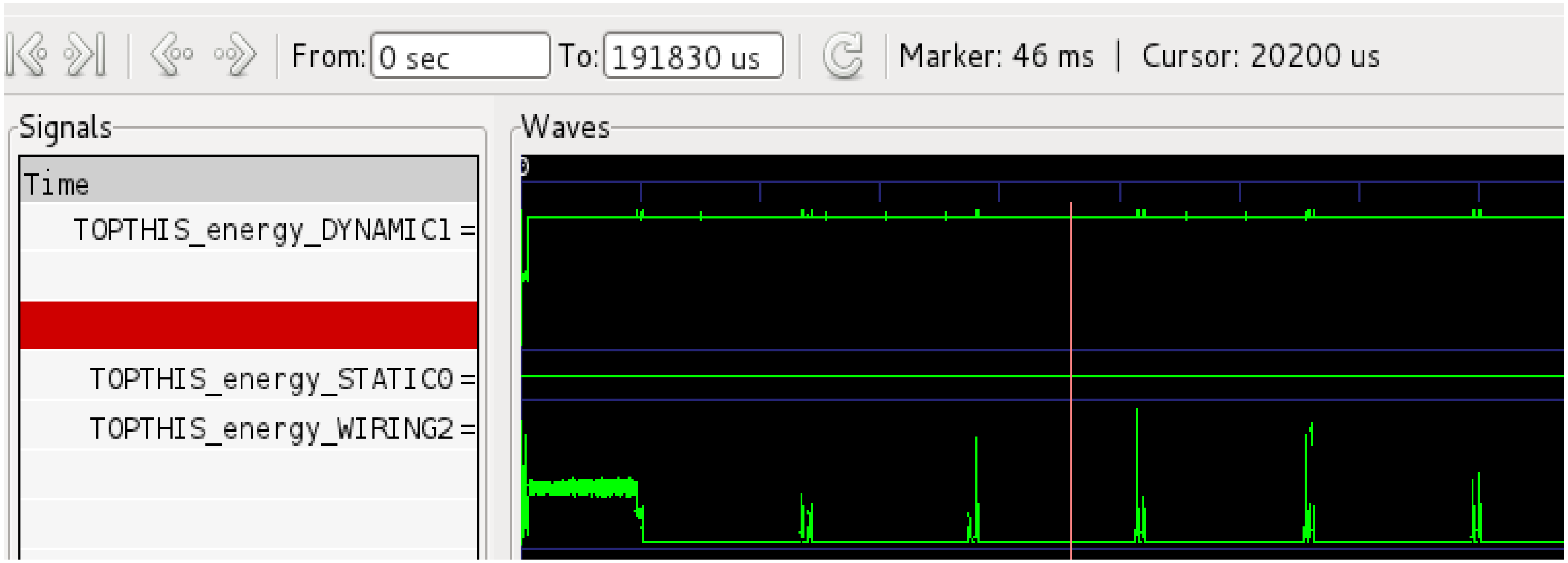
Hop Tracking: Origin/Hop/Terminus.

Option 1: Track transaction trajectory to get distance travelled.

```
trans.pw_set_origin(this, PW_TGP_ADDRESS | PW_TGP_ACCT_SRC, &frontside_bus);
initiator_socket->b_transport(trans, delay);
trans.pw_terminus(this);
```

- Initiator makes the origin and terminus calls.
- Intermediate nodes (cache and bus models) call log_hop.
- Flags enable energy to be logged at src or dest.
- Options 1+2:
  - For additional transition counting, need to know which bus transaction is on and which fields in TLM payload are active.

Report Formats (3: VCD)



- Each account and their summations can be plotted in various forms
  - 1: Ascii-art table format
  - 2: SYLK or CSV spreadsheet format
  - 3: VCD temporal display (using dirac impulse response or average over interval)
- A physical layout file is also written.

An OpenRISC Core in TLM Form

Two approaches to getting an OpenRISC core:

1. Verilated:
   - Use OR1200 in verilog and pass through Verilator to create net-level SystemC.
   - Manually write a TLM 2.0 wrapper for it.

2. ORSIM ISS:
   - Take the (auto-generated?) sim.C code from orsim
   - Add some backdoor nops
     (e.g. atomic prefix for load-linked bus transaction)
   - Manually write a SystemC TLM wrapper for it.

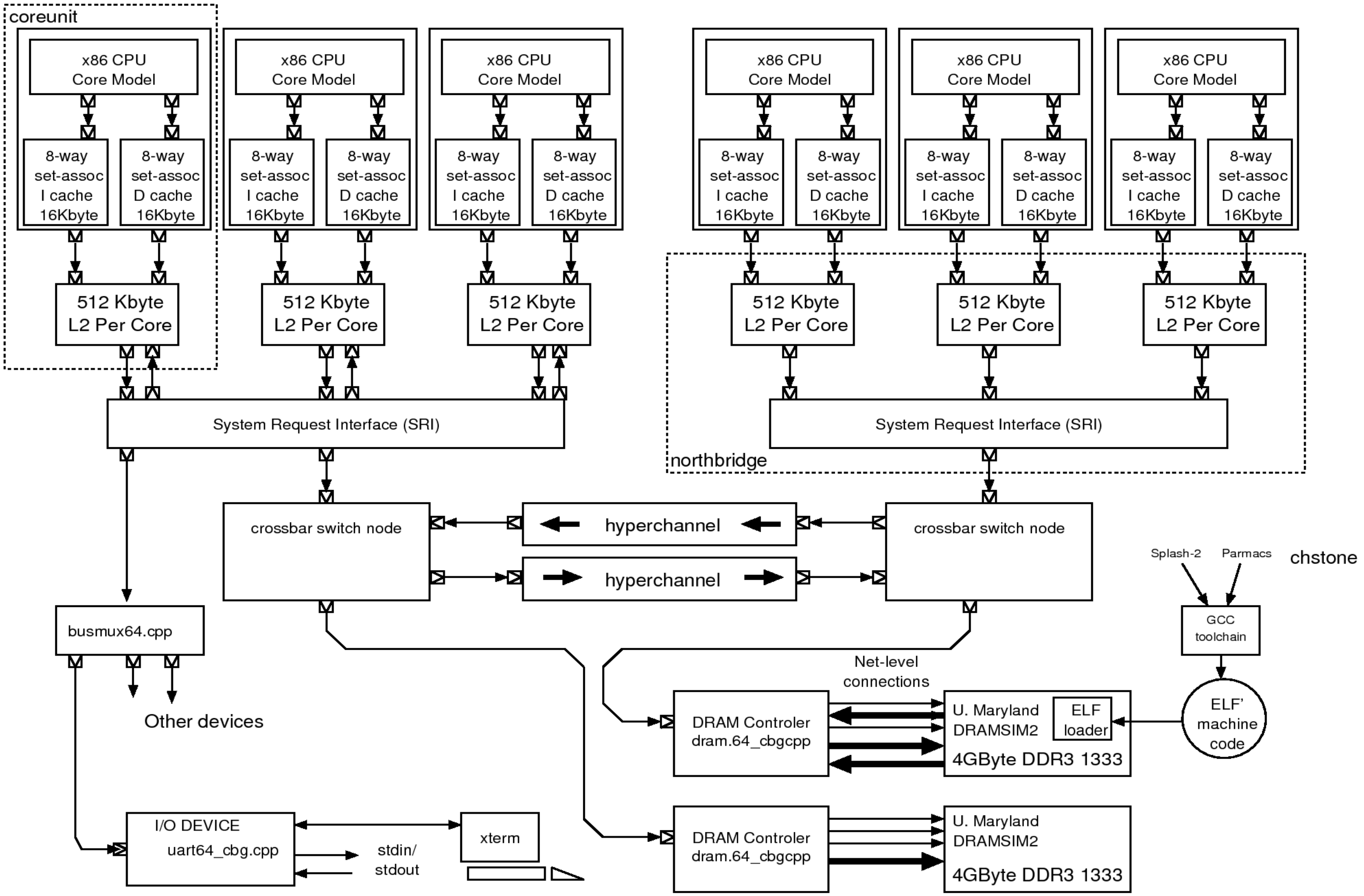
OpenRISC Core Power Annotation

Two approaches to getting an OpenRISC core:

1. Verilated:
  - Add a static power consumption in the constructor.
  - Modify Verilator's net update macros to debit energy quanta according to hamming distance (TODO).

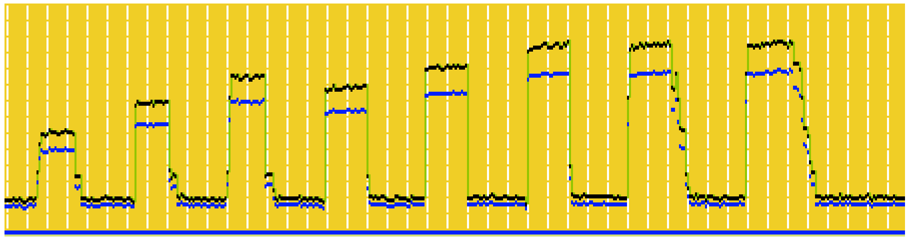
2. ORSIM ISS:
   - Add a static power consumption in constructor.
   - Adjust static power mode on any sleep modes.
   - Add an array giving the complexity of each instruction.
   - On each instruction, debit dynamic energy proportional to complexity.

AMD Phenom 6 Core Model



Phenom Corner Cases: 1 to 8 threads

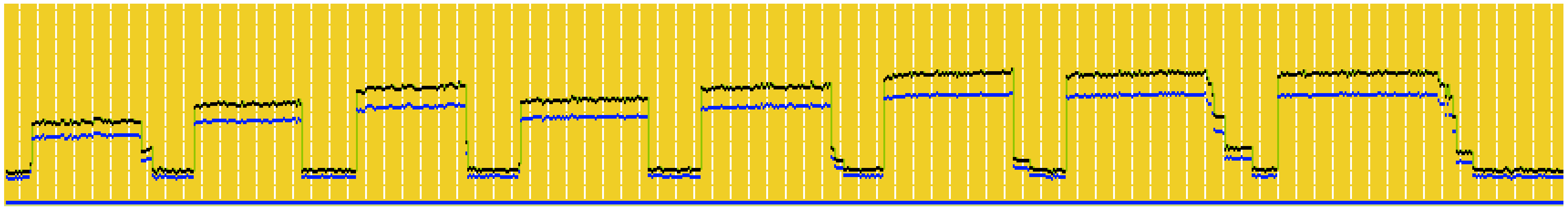
Integer ALU



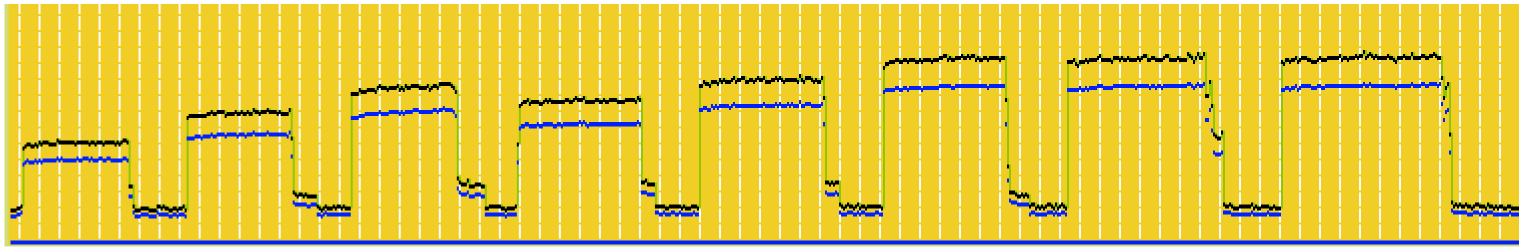
Vertical bar -> 1 second.
Horizontal scale -> 100 Watts.

System has 6 cores sharing one DRAM bank.

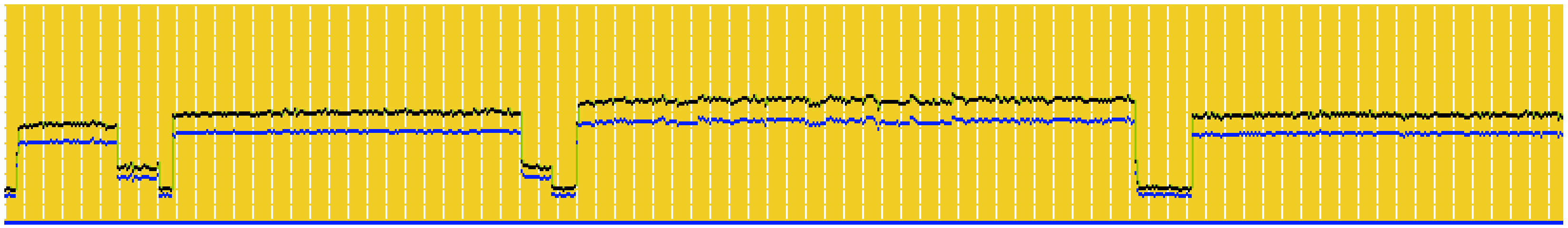
Floating Point ALU



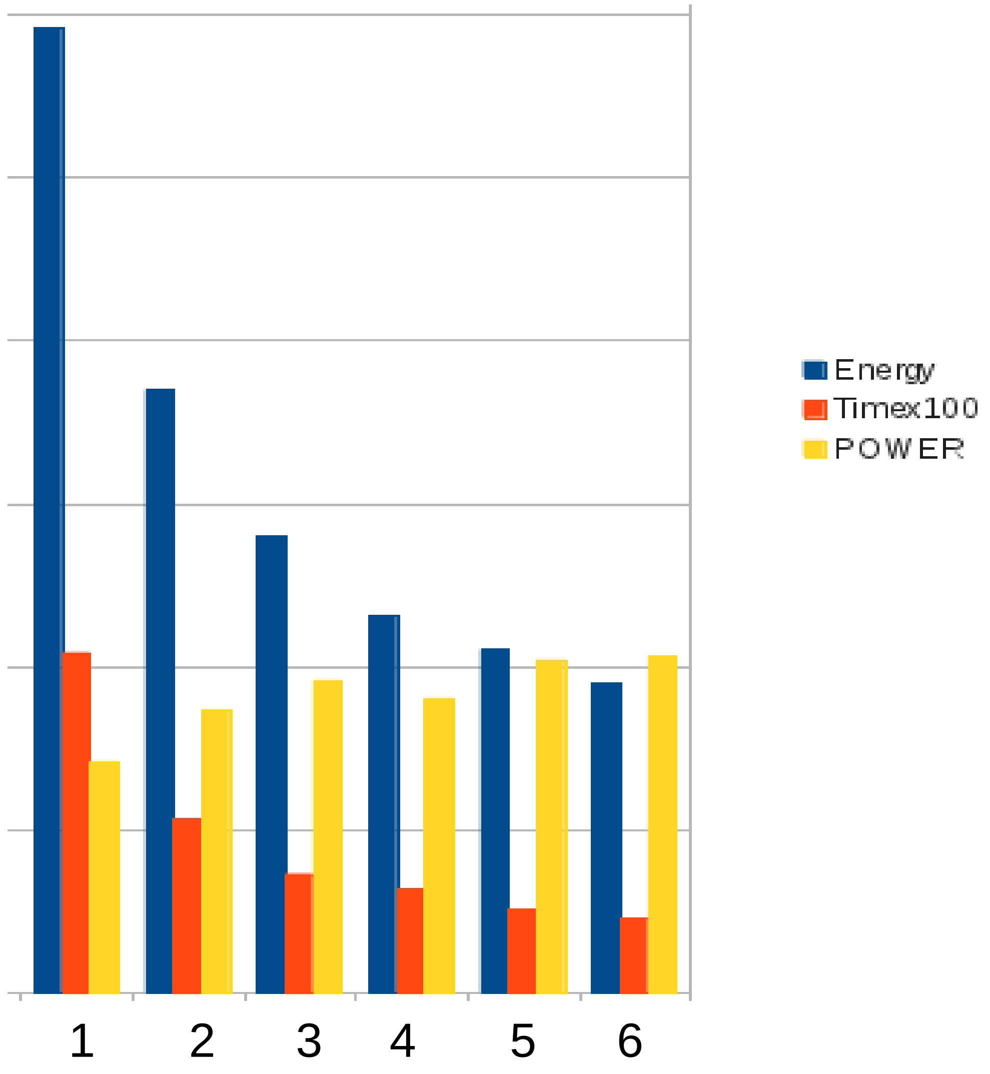
Memory Access: Disjoint Footprints



Memory Access: Overlapping Footprint



Splash-2 'RADIX' : Power + Energy



Running the RADIX test on
n = 1 to 6 cores.

Program modified to suit n not a power of 2.

Increasing n ---> increased performance.

Increasing n ---> better efficiency.

Strange power humps !

One DRAM DIMM shared.

Phenom Energy Coefficients

Instruction	1 nJ
I Cache Miss	50 nJ
D Cache Miss	15 uJ
D Cache Snoop Read	4 mJ
D Cache Evict	7 mJ

Values obtained from curve fitting

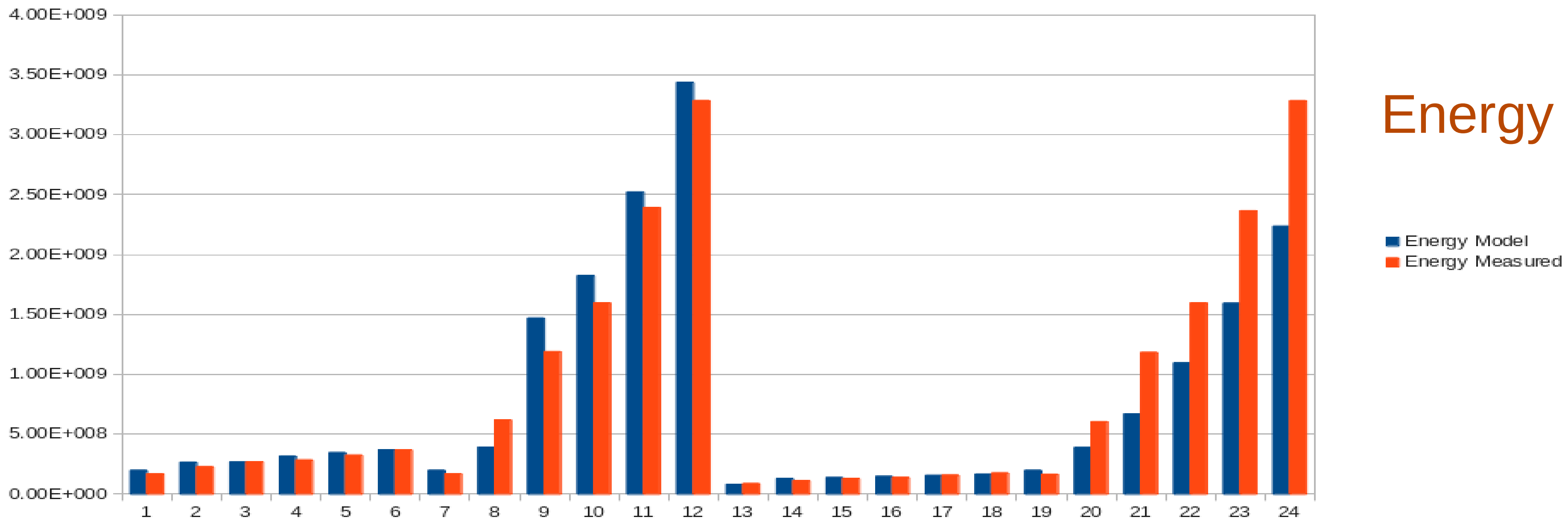
CPU + Caches only.

DRAM excluded.

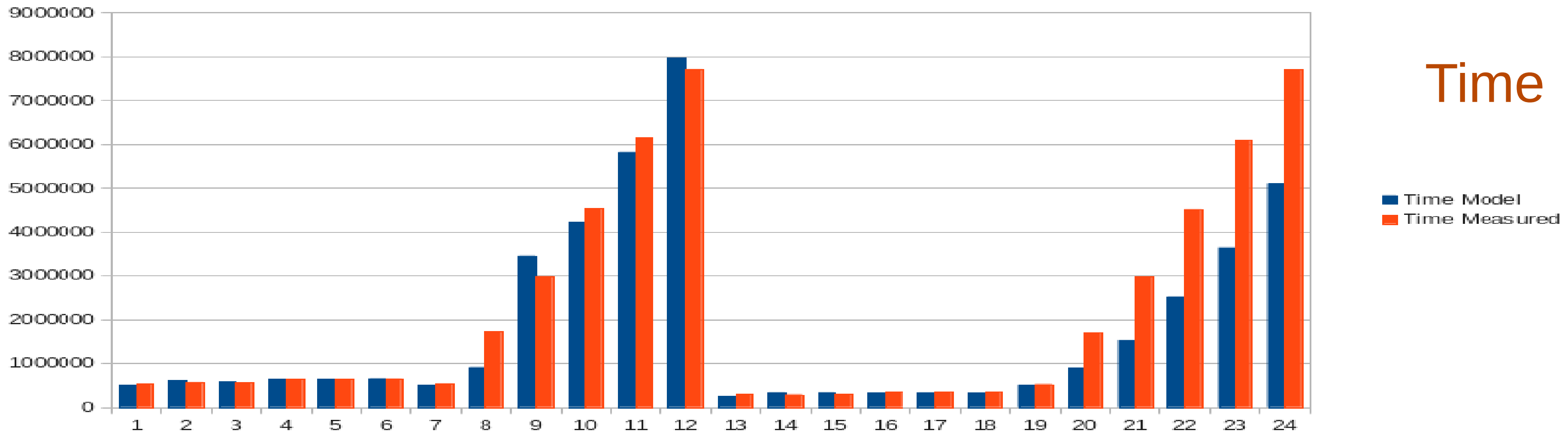
Measured v Predicted:

Runs 19-24 extrapolated from data fitting on 1-18.

Energy



Time

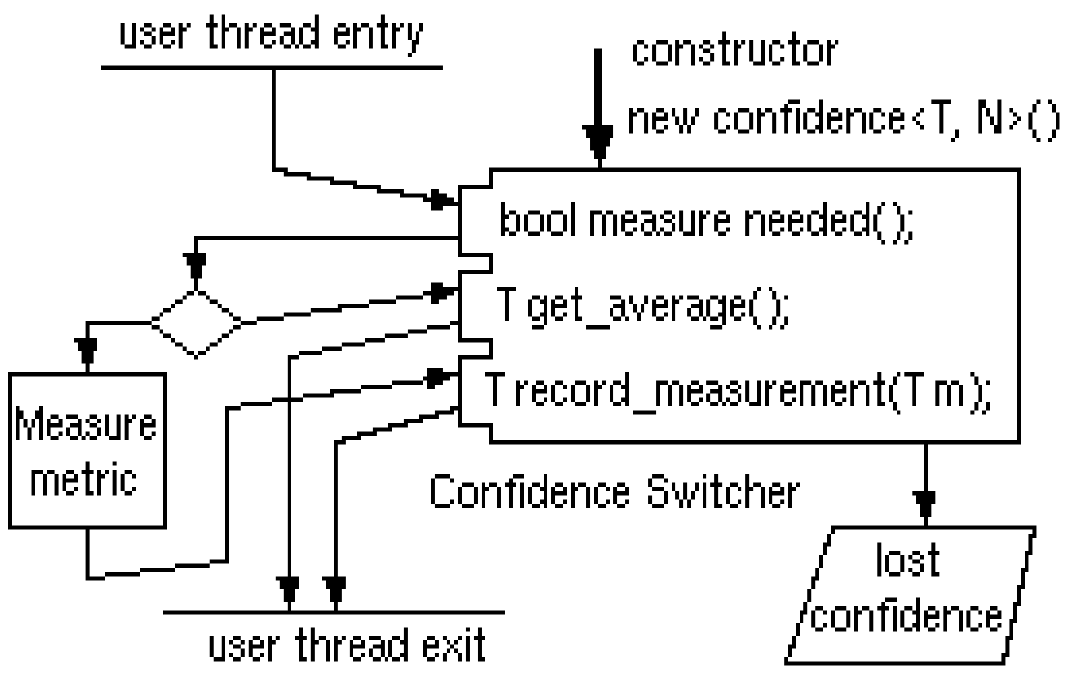


Static or Initial Parameters (2)

- Set up static parameters in constructor:
    - Excess or actual area or dimensions
    - Static power consumption
    - Chip/region name
    - VCC supply voltage

- Optional per-instance or per-kind technology file (XML) can be accessed (defines phases and modes and default VCC ...).
- Some are less static:
    - Set these in PVT change callback (virtual function).
    - Call that yourself from constructor.
- PVT called-back when VCC changes.

Confidence Switcher



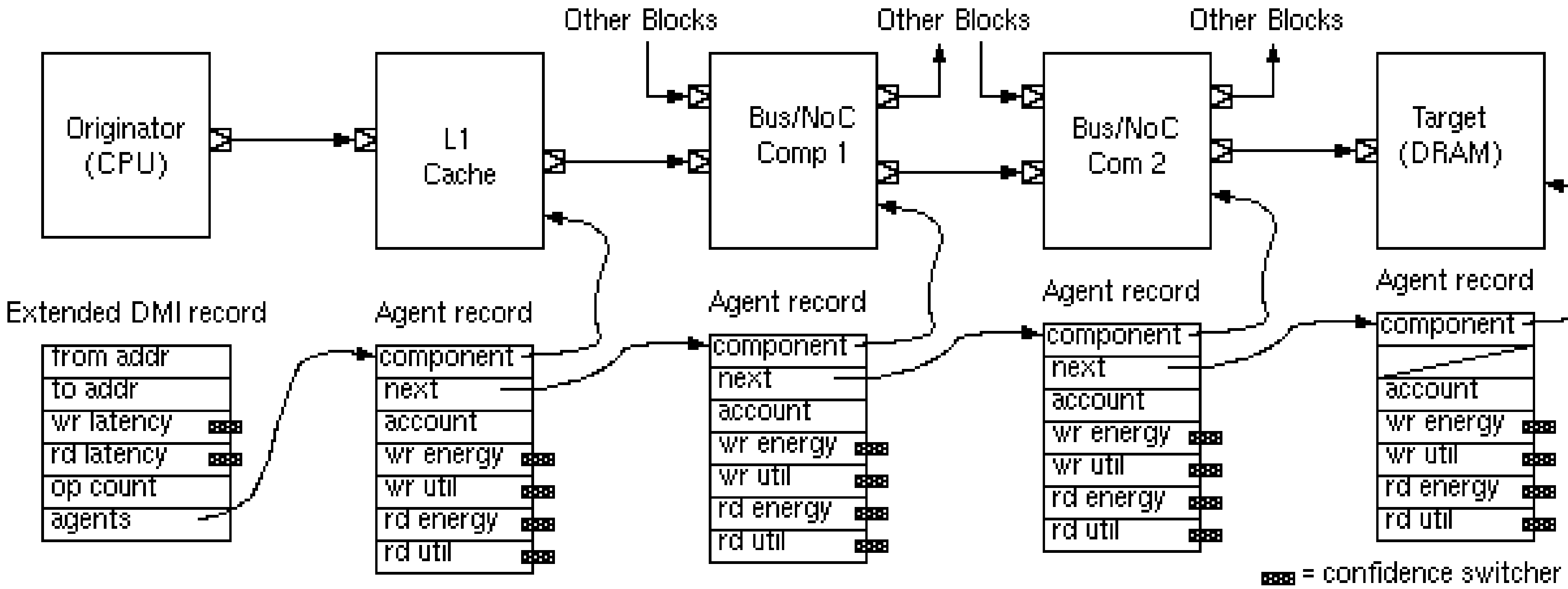
Generic API for a measuring and estimating component.

Use for time, energy, transition count and so on ...

Very simple implementation if we just want an estimate of the average metric:

Discard first N measurements, average next N, return this value while making an actual measurement one in every N to check for LOSS OF CONFIDENCE.

Augmented DMI Flow



Latency can be credited to the initiating thread's 'delay' as always.

Energy *should* be credited to the intermediate components:

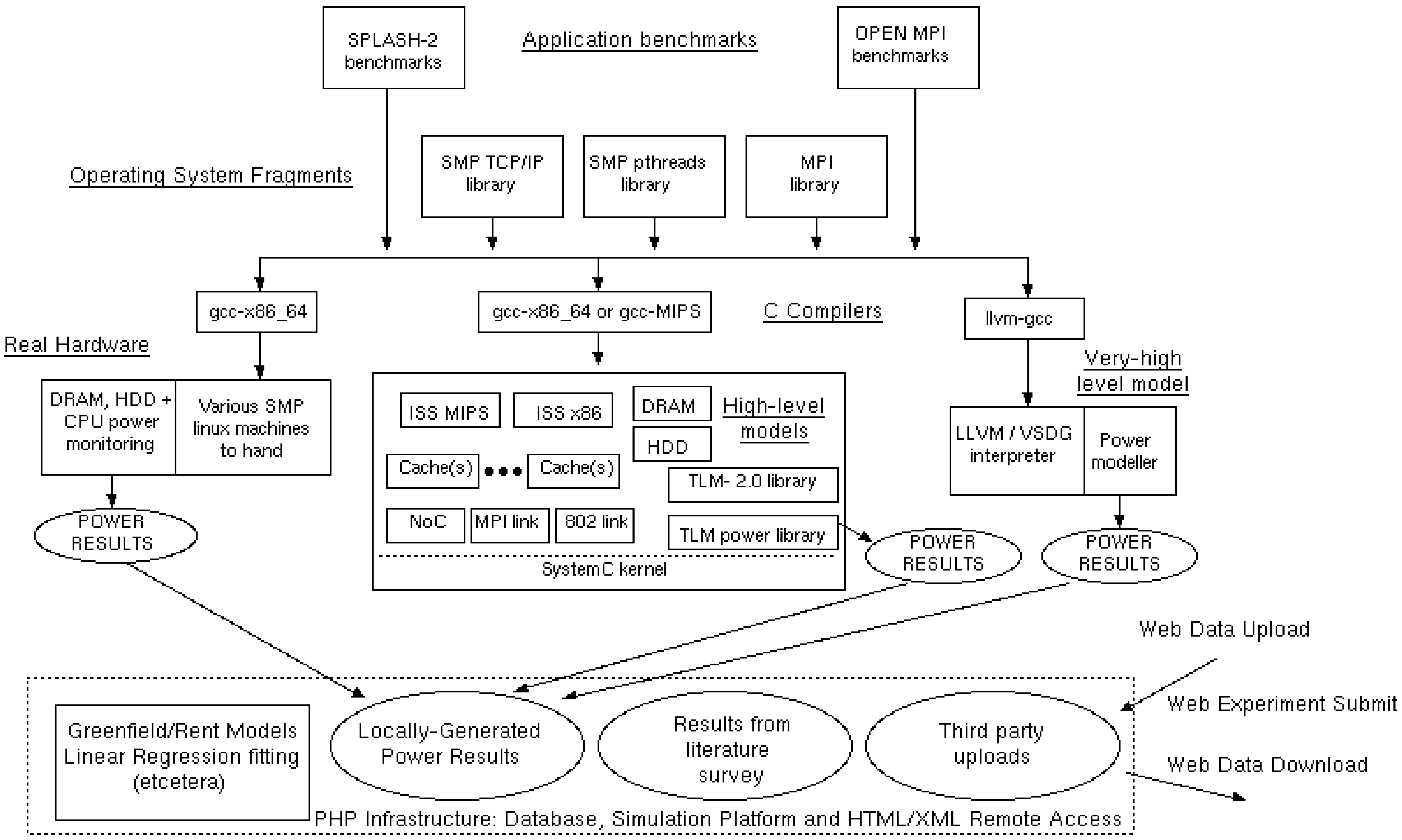
so DMI record at initiator is extended with either

a) a list of intermediate agents that have their own records

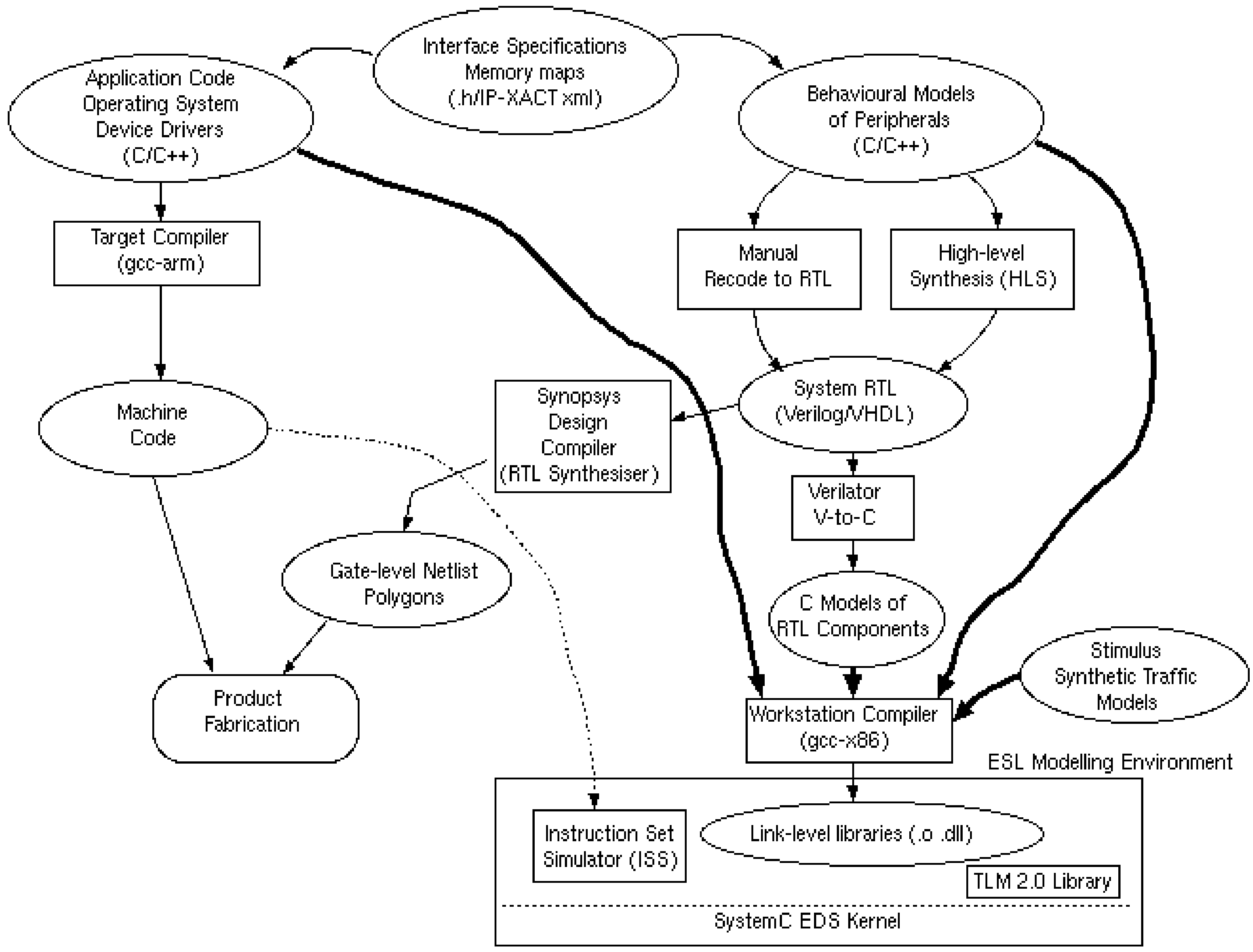
or

b) bulk read and write energy records (simpler, not shown).

Power Estimation: Project Flow



Backup Slide: ESL Modelling Flow



Interface Specifications Memory maps (.h/IP-XACT xml)
Application Code Operating System Device Drivers (C/C++)
Behavioural Models of Peripherals (C/C++)
Target Compiler (gcc-arm)
Manual Recode to RTL
High-level Synthesis (HLS)
Machine Code
Synopsys Design Compiler (RTL Synthesiser)
System RTL (Verilog/VHDL)
Verilator V-to-C
Gate-level Netlist Polygons
C Models of RTL Components
Stimulus Synthetic Traffic Models
Product Fabrication
Workstation Compiler (gcc-x86)
ESL Modelling Environment
Instruction Set Simulator (ISS)
Link-level libraries (.o .dll)
TLM 2.0 Library
SystemC EDS Kernel

Talk Overview

- SystemC + TLM Introduction
- TLM POWER 2
- TLM POWER 3
  - Loose timing
  - Energy based
  - Layout aware
  - Bit transition counting
- Splash-2 benchmarks, power probed.
- Data fit x86_64 to OpenRISC !

Loosely-Timed: Effect of Quantum

Two cores running: main() { for(i=0;i<5;i++) puts("Hello World"); }

Core clock Is 200 MHz (5ns period).

```
Sim Start: cores=2
HHelleol lWoo rWlodr
ld
HHeellllloo  WWoorrlldd

HHeellllloo  Wwoorrlldd

HHeellllloo  WWoorrlldd
H
eHlellol oW oWrolrd
ld
CPU 0 exit: insns #717
CPU 1 exit: insns #717
```

Global Q = 5ns
Lock-step execution

```
Sim Start: cores=2
Hello World
HeHello World
Hello World

Hello Woolo World
Hello rld
Hello World
World
Hello Wor
Hello World
CPU 0 exit : insns #717
CPU 1 exit:  insns #717
```

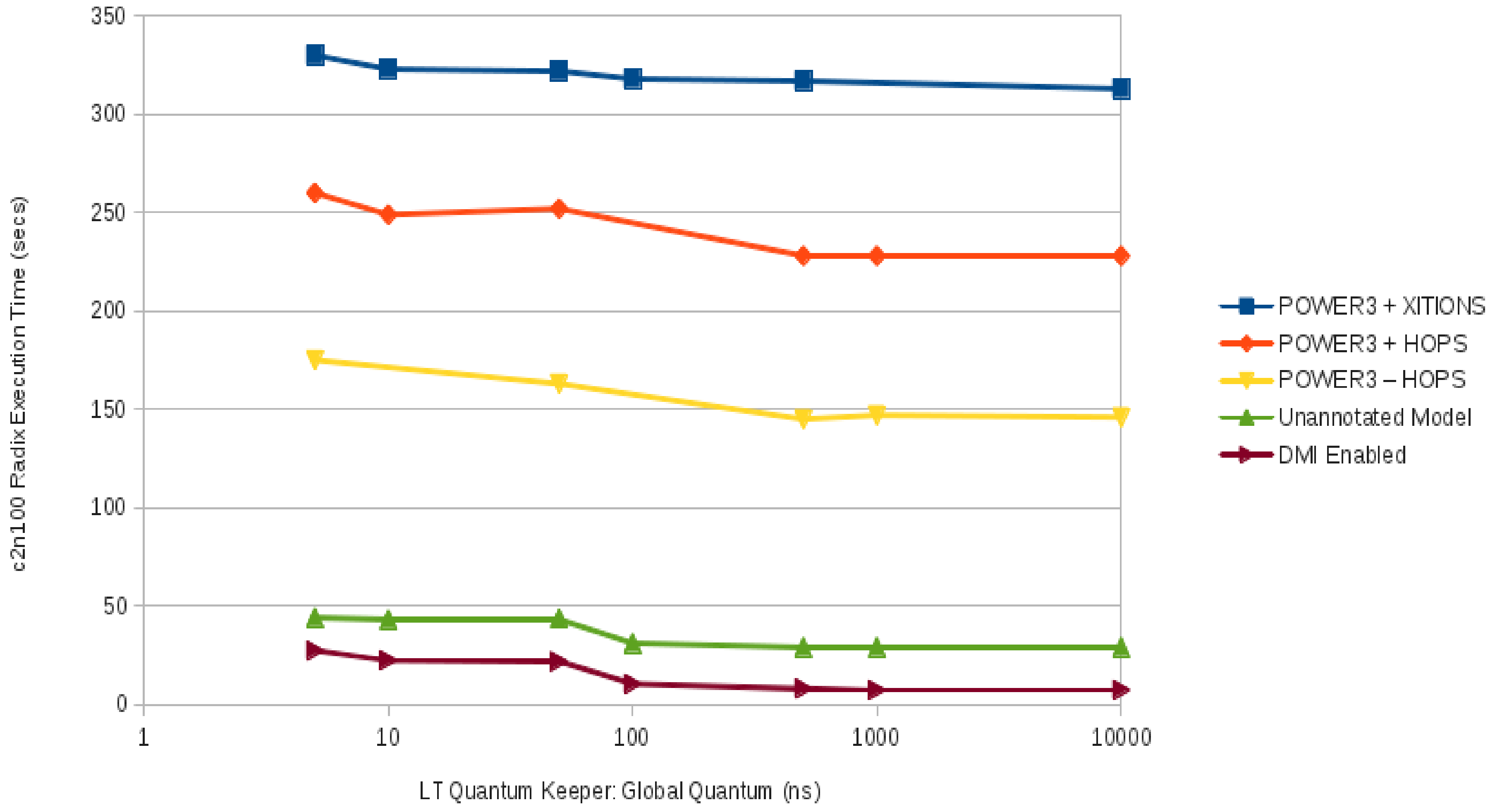
Global Q = 1us
Finely interleaved

```
Sim Start: cores=2
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
CPU 0 exit: insns #717
CPU 1 exit: insns #717
```

Global Q = 100us
Coarsely interleaved

Three different settings of the global quantum.

Loosely-Timed Performance Lost



Relative performance of LT TLM Model (2 cores, running SPLASH-2 Radix Sort n=100)