Performance Monitor PowerPC Perspective

Alex Mericas
February 12, 2005

Acknowledgements

- **No list would be complete….**
- **Evelyn Duesterwald**
- **Maynard Johnson**
- **Luc Smolders**
- **Peter Sweeney**
- **Bob Wisniewski**

Requirements

- **Easy to use, easy to program**
- **Robust, full feature**
  - Wide variety of users
    - HW Performance
    - SW Performance
      - Applications
      - OS
      - Compiler
- **Accurate**
- **Efficient**
  - Power
  - Area
  - Design and Implementation resources
  - Verification

Power5 Performance Monitor Features

- **Third Generation Design**
- **Detailed coverage of performance sensitive events**
  - Core, Cache, Outboard units (e.g. memory controller)
  - Unit usage, utilization
  - Queue allocation, occupancy
  - Cache Reload source (L2, L3, Memory)
  - Latency measurement by cache level
  - Cache contention indicators (state transitions)
  - Architectural hazards
  - Completion Delay
  - SMT, SMP effects
- **Profiling Support**
  - Uniquely identify cause of events (when known)
  - Threshold events for long latency conditions

Why

- **Why did you do *that*?**
- **Why don't you do *this*?**
- **Why is it so hard do to what I want?**

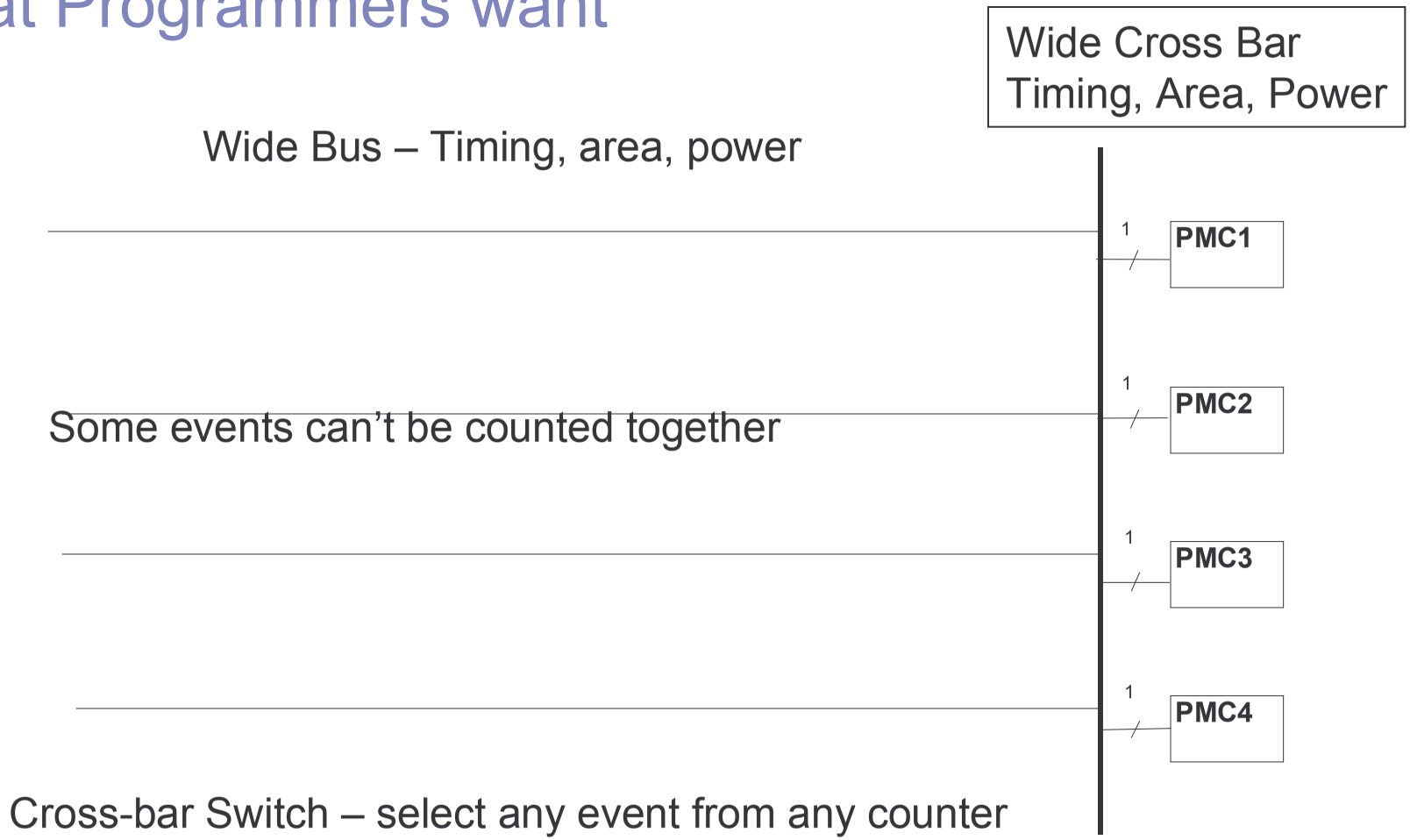
Example: Event Selection

- **I just want to count ….**
  - PPC604e had ~ 128 total events
    - ~ 110 unique
    - ~ 32 events per counter
  - POWER4 had ~ 900 total events
    - ~ 300 unique events
    - ~ 115 events per counter
  - POWER5 has ~ 900 total events
    - ~ 500 unique events
    - ~ 230 events per counter

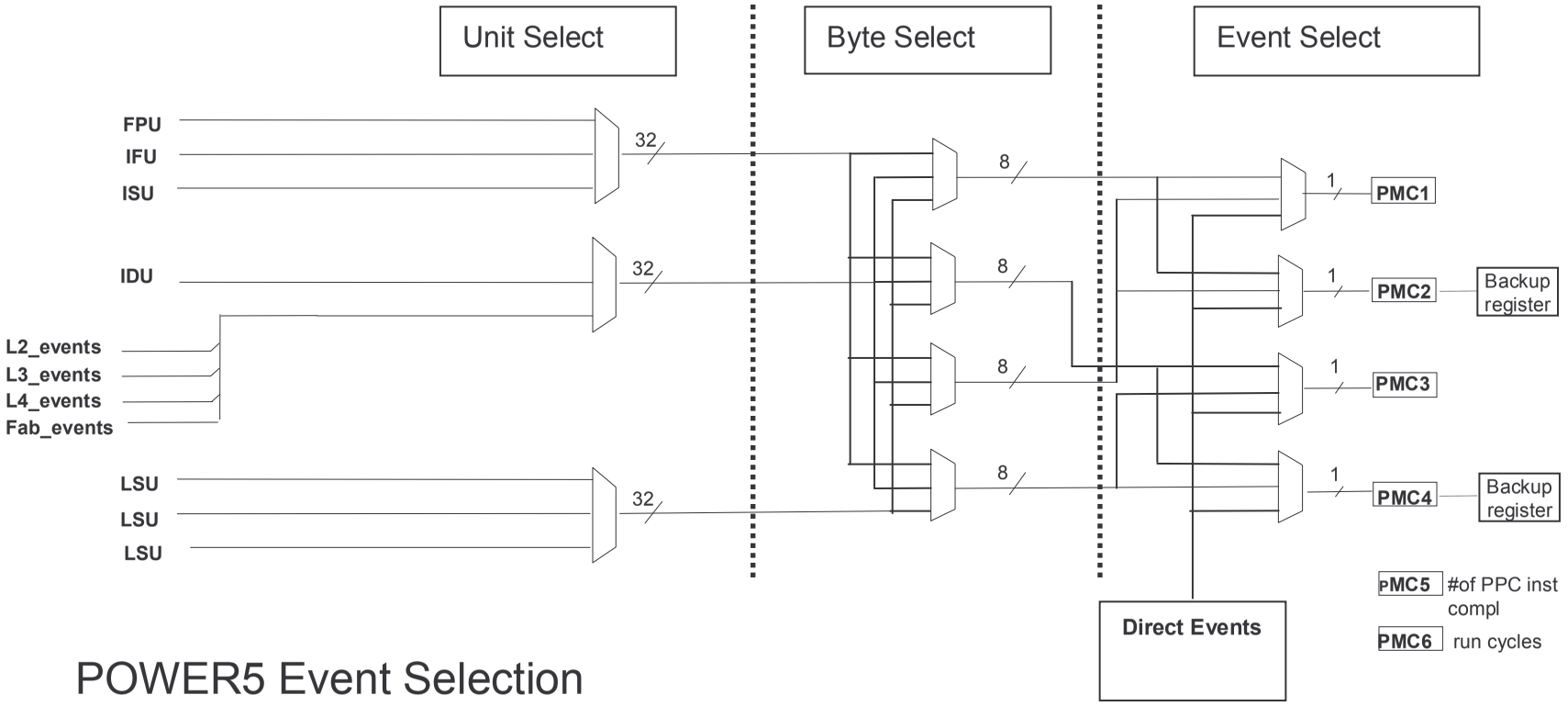
What Engineers Want

- **Generic interfaces**
  - Mux
  - Adders
  - Decoders
- **Little or no “special logic”**
  - All events are treated the same
    - Signals are signals
- **Low-cost**
  - Power
  - Area (total, location)
  - Wires/pins
  - Verification

What Programmers want



Hierarchical Event Routing



POWER5 Event Selection

Event Selection Challenges

- **The event you configure for PMC1 will affect the events available for PMC2-n**
  - Events on same byte but different units
  - Events on different byte but same unit mux
- **Simple solution is to pre-defined legal combinations**

Example: Profiling

- **Identify hotspots in code/data**
- **Find performance sensitive areas**
- **Identify problem instructions and/or data areas**
- **POWER5 makes this challenging**
  - 5 instructions/group
  - 20 groups past dispatch
  - 32 outstanding loads
  - 16 outstanding misses
  - 2 independent threads
  - Decoupled nest/core
- **Close isn't good enough**

Which Instruction?

- **Continuous**
  - Capture every instruction, data address
  - RS64, POWER5
- **Queue based**
  - One slot is instrumented, instruction in that slot is *marked*
  - Instruction and Data address captured for marked instructions
  - Events attributable to marked instructions are called *marked events*
  - PPC604, POWER3, PPC750
- **Random**
  - Randomly pick an instruction to mark
  - Capture instruction, data address
    - Instruction address is 1st instruction in dispatch group
  - POWER4, PPC970, POWER5
- **Event Based**
  - Tag instruction that causes monitored event(s)
  - Capture instruction, data address of tagged instruction

POWER4 Features
Sampling

- **Introduced concept of “eligible” instruction**
  - A single instruction is sampled from eligible
  - POWER4 could only count eligible instructions
- **Sampling eligibility**
  - Random
  - OpCode Match
  - Load/Store
- **Sampled instructions tracked from dispatch to completion**
  - Ability to apply threshold checking on pipeline stages

New Features in POWER5
Sampling

- **Slot offset within Sampled Group**
- **Continuous Sampling**
- **Limited “look back” ability**

Slot Offset

- **Instructions are dispatched and completed in groups**
- **POWER4 captured Instruction Address of first instruction in group that contained a sampled instruction**
- **POWER5 captures offset within group**
- **Better granularity for profiling**

Continuous Sampling

- **Capture Instruction Address for every completion**
- **Capture Data Address for every**
  - Data Cache Reload (cache line address)
  - TLB Reload (page address)
- **Better resolution for hotspot analysis**

Look Back Capability

- **Add an bit in completion table to indicate that group suffered a branch mispredict or an icache miss**
  - Pick one

New Features in POWER5

- **PowerPC Instructions now counted**
  - Eligible instructions still counted as on POWER4
- **Data Source for translation requests**
- **More outboard events (memory, bus)**
- **Chained Counters**
- **Speculative Counters**
- **Simultaneous Multi-threading support**

Speculative Counters

- **Sometimes we don't know why we're waiting until after we're done**
- **At beginning of a wait**
  - Checkpoint counter value
  - Speculatively count cycles for a particular reason
- **At end check reason**
  - Correct: keep count
  - Incorrect: restore to checkpointed value

POWER5 CPI Stack

<table>
<tr><td rowspan="11">Total Cycles</td><td rowspan="2">Completion cycles <A:group complete cycles></td><td colspan="3">PPC Base completion cycles
<A1: One or more PowerPC instructions completed this cycle></td></tr>
<tr><td colspan="3">overhead of cracking/microcoding
<A2:(A)-(A1)></td></tr>
<tr><td rowspan="3">Completion Table empty <B></td><td colspan="3">I-cache miss penalty <B1></td></tr>
<tr><td colspan="3">Branch redirection (branch misprediction) penalty
<B2> PMC4SEL=0x38</td></tr>
<tr><td colspan="3">others (Flush penalty etc)
<B4: (B)-(B1)-(B2></td></tr>
<tr><td rowspan="6">Completion Stall cycles <C: total-(A)-(B)></td><td rowspan="3">Stall by LSU inst
<C1></td><td rowspan="2">Stall by reject
<C1A></td><td>Stall by Translation (rejected by ERAT miss)
<C1A1></td></tr>
<tr><td>other reject
<C1A2: (C1A)-(C1A1)></td></tr>
<tr><td colspan="2">Stall by D-cache miss
<C1B></td></tr>
<tr><td colspan="3">Stall by LSU basic latency, LSU Flush penalty
<C1C: (C1)-(C1A)-(C1B)></td></tr>
<tr><td rowspan="2">Stall by FXU inst
<C2></td><td colspan="2">Stall by any form of DIV/MTSPR/MFSPR inst
<C2A></td></tr>
<tr><td colspan="2">Stall by FXU basic latency
<C2C: (C2)-(C2A)></td></tr>
<tr><td rowspan="2">Stall by FPU inst
<C3></td><td colspan="2">Stall by any form of FDIV/FSQRT inst
<C3A></td></tr>
<tr><td colspan="2">Stall by FPU basic latency
<C3B: (C3)-(C3A)></td></tr>
<tr><td colspan="3">others (Stall by BRU/CRU inst , flush penalty (except LSU flush), etc)
<C4: (completion stall cycles)-(C1)-(C2)-(C3) ></td></tr>
</table>

Simultaneous Multi-Threading support

- **Units are pipelined**
  - On any given cycle could be executing two threads
  - Event signals had to be replicated
  - Some events are shared between threads (cycles)
- **Each thread is a logical processor**
- **Simple solution was to replicate PMU**
  - 4 programmable counters per thread
  - 2 fixed counters per thread (cycles, instructions)
- **Thread interaction events**

More Challenges

- **Signal timing**
  - Globally
  - Within unit
- **Speculative Execution**
  - Tracking to completion is expensive
    - A single event requires 100 bits in POWER5!
- **Number of events**

Looking Forward

- **PowerPC PMU design is scalable**
  - More/fewer counters
  - More/fewer events
- **Very core-centric**
  - Outboard events can be routed to core(s)
  - How to correlate nest event to particular instruction?
- **Complexity**
  - Would a simpler design be more useful?

Software Perspective

- **Documented/shipping interfaces**
  - Kept as simple as possible
    - Table driven
      - Hiding Power4/PowerPC970/Power5 event selection complexity from users
    - To be able to tolerate processor differences
      - Current code supports 11 types of processors
  - Hides some hardware features
    - Added several over time
    - Currently no plans to propagate interrupt on overflows.

Software Perspectives (cont)

- Exposing 64bit software counters
  - Virtualized mode
    - Support both kernel thread and pthreads
  - System-wide mode
    - Automatic overflow accumulations.
- Third part caller interfaces available
  - Support both ptrace and /proc based debuggers

Software Plans

- **Support libhpm/hpmcount/hpmstat**
  - Using table based derived metrics
    - Struggling to provide same set on all processors
    - Need standardized events!
- **Ship hardware event based profiling tool**
- **Counter timeslicing**
- **User mode 64bit counters reading**