

Power Architecture™ '07
DEVELOPER CONFERENCE



Integrated Multi-Application Safety Critical and Secure Systems

Tim Skutt, GE Aviation

25 Sept. 2007

Power.ORG ™

Introduction



» In this presentation, we'll look at:

- **Challenges** in multi-application safety and/or security certified systems
- **Unique barriers** to efficient processor utilization in these systems
- **Solutions** to **break through** the barriers

» The discussion comes from an aerospace background, BUT

- Other safety/security critical domains including automotive, industrial, and medical are (or will be) experiencing the same issues
- Domains with mission critical systems are impacted by these issues as well

Challenges



» First – what's the problem?

- You have multiple applications that need to fit into a system
- Your size, power, weight, and/or cost constraints prohibit a separate processor per application
- What do you do? (a **challenge**)

» An easy answer – share processors among multiple applications

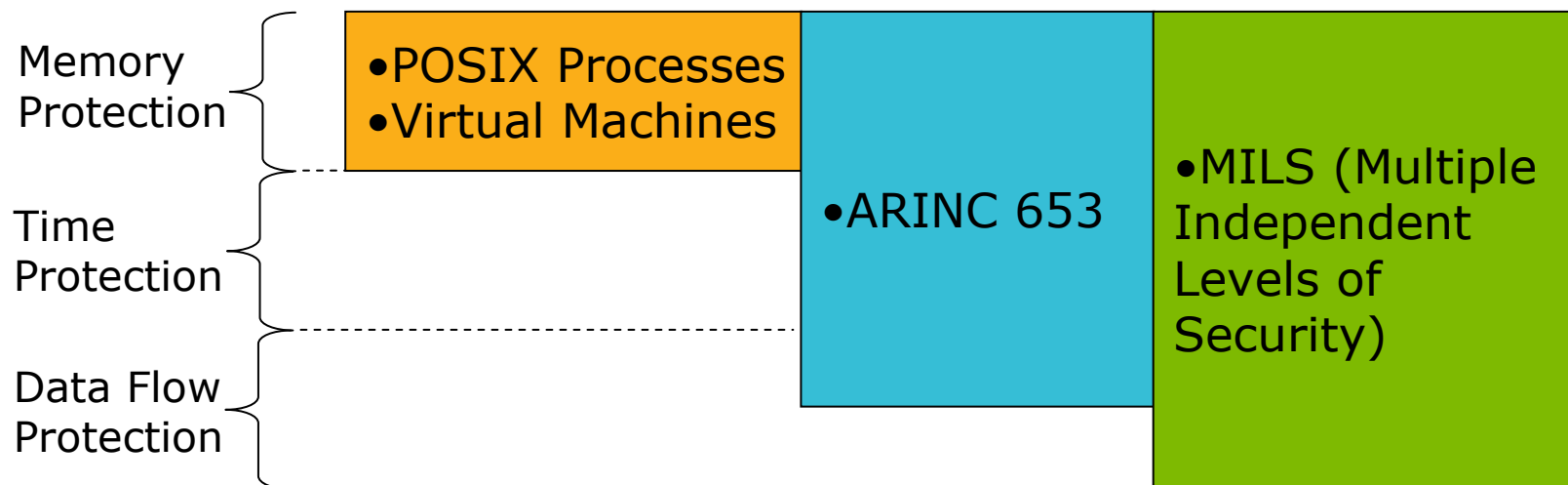
- What if your applications are different levels of safety criticality or security?
- What if your applications come from different sources and may not play well together?

Challenges



» A revised answer – use an Integration Framework to share the processor

- What's an Integration Framework?
 - A software infrastructure designed to provide a level of protection between applications
- Examples:



Challenges



- » Safety or security certified projects require safety and/or security analysis (driven by certifying agency – FAA, FDA, NSA)
 - Identify risks/threats
 - Identify risk/threat probabilities
 - Identify risk/threat impacts
- » Integration introduces new risks/threats and increases the impact of some previous risks/threats (another **challenge**)
 - Example: New risk/threat – one application's actions can indirectly cause a failure in another application (e.g., denial of service attack on a framework or hardware resource)
 - Example: Increased impact – one hardware failure can now cause a failure in multiple applications

Barriers



- » New or increased impact risks/threats drive designs that hit **barriers** to efficient processor utilization

- » We'll discuss two categories of these barriers:
 - Performance loss due to processor architecture choices
 - Inefficiencies resulting from processor vulnerabilities

Performance Loss Barrier



» When multiple applications are integrated on the same processor performance is lost on shared resources

- Pipelines
- MMUs
- Caches

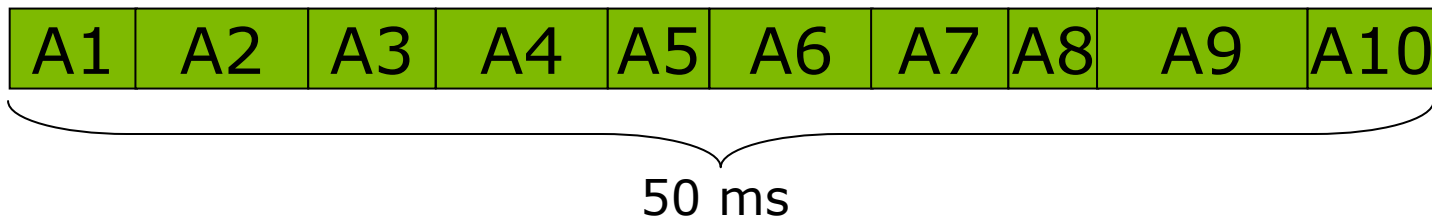
» How big are the potential losses?

- It depends on the applications and the framework
- Assume that a typical system has
 - 10 applications on a processor
 - Each application has a 50 ms fundamental period

Performance Loss Barrier



» BEST CASE: Each application is activated once in a 50 ms window



» Implications

- 10 pipeline flushes per 50 ms
- 10 MMU context switches per 50 ms
- 10 applications filling and flushing different cache lines

» Note: This will be worse if applications preempt each other and cause additional application activations

Performance Loss Barrier



» Impacts

- Pipeline flush – small ($< 1 \text{ us} \rightarrow < 1\% \text{ overhead}$)
 - A handful of cycles for each flush
 - Mostly hardware implementation dependent
- MMU context switch – medium ($1 - 100 \text{ us} \rightarrow \leq 2\% \text{ overhead}$)
 - Number of cycles depends on framework implementation
 - Strong hardware implementation influence (e.g., Power QUICC II Pro vs. Power QUICC III)
- Cache – hmmm... let's look at this in more detail

» Cache Impact Analysis

- Measurement: L1 data cache fill and flush takes $\sim 290 \text{ us}$ for contiguous data
 - Hardware platform: 800 MHz PPC 7457, 100 MHz SDRAM

Performance Loss Barrier



» Cache Impact

- 290 us per application context switch (~6% overhead)
- Why?
 - If the execution during A2 uses none of the cache lines loaded during A1, then all of the L1 data must be flushed for A1, and then filled with cache lines for A2
 - This sounds very pessimistic. Characteristics:
 - All cache lines contain modified data and must be flushed
 - No re-use of cache lines between A1 and A2 (including framework)
 - BUT, while the above characteristics may be unrealistic, they are offset by characteristics of the measurement:
 - Made with contiguous data – real-world data will not be contiguous forcing latency inefficiencies with RAM devices
 - Made by first filling then flushing – real-world activity mixes fills and flushes forcing latency inefficiencies with RAM devices

Performance Loss Barrier



» How do faster processors/memory subsystems impact this?

- The jury is still out, but consider that
 - Faster processing will likely result in more applications on a processor
 - Vehicle systems are requiring faster and faster application rates as vehicles rely more heavily on software control for stability
- Result – utilization factors will likely keep percentages from changing

Performance Loss Barrier



» Conclusion – Cache effects on integrated systems can be a performance loss barrier

- Clearly, mileage will vary, but the order of magnitude is in the right ballpark

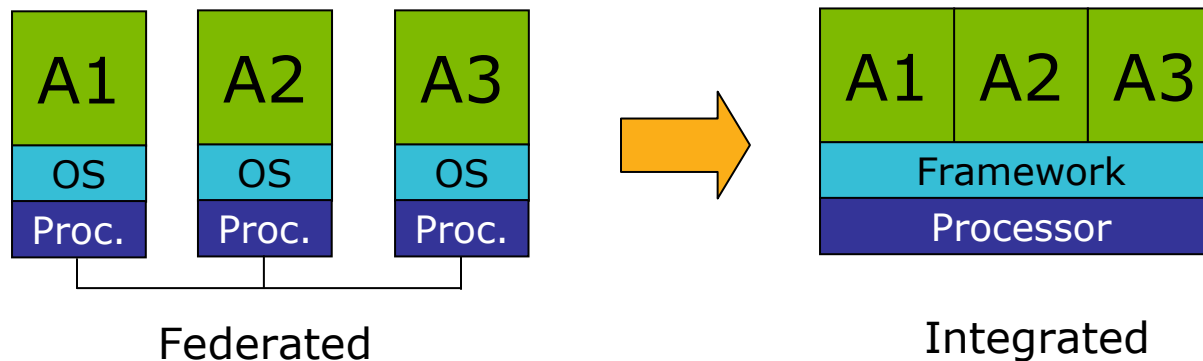
» Note:

- The same issues causing performance loss also impact Worst Case Execution Time (WCET) – a factor that must be analyzed for safety certification

Vulnerability Inefficiency Barrier



» As multiple applications are integrated onto processors, the impact of failures in the processor or framework increase



» Impact of a processor failure

- Federated System - one application unavailable
- Integrated System - three applications unavailable

Vulnerability Inefficiency Barrier



- » An increase in impact often results in
 - **Tighter requirements**, which lead to
 - **Higher scrutiny** in safety/security analyses, which lead to
 - **Identification of vulnerabilities** in the design, which lead to
 - **Mitigation requirements**
- » Mitigation requirements add features into the system that typically make the system less efficient
 - Use more CPU
 - Introduce wait states
 - Etc.
- » Where do these vulnerabilities surface?

Vulnerability Inefficiency Barrier



» Vulnerability sources

- Features whose functionality cannot be verified
- Features that cannot support the required integrity

» Verification barriers

- Simply put, in a safety or security critical environment if you can't verify a feature's functionality, you have to assume it doesn't work
 - Assuming features don't work typically means adding functionality elsewhere to provide the equivalent
- Example: The PPC 7457 does not provide a way to verify the functionality of the L2 cache checksum feature
 - Implication: Safety and security analyses must assume there is no checksum feature
 - Result: Inefficiency due to mitigations (e.g., don't use L2 or flush/invalidate L2 often to reduce vulnerability)

Integrity Barrier



» Integrity barriers

- Integrity properties are typically a combination of design and environment
 - Analysis assumes design errors are detected through verification activities, so the focus is on transient failures (e.g., radiation induced Single Event Upsets)
 - Analysis further focuses on failures that are **not detected** or the **tenure of a failure before it is detected**
- Primary integrity issues surrounding multi-application integration are in shared resources
 - Registers
 - Logic
 - Busses
 - Memories (both on-chip caches as well as external memories)
- Registers, Logic, and Busses have very low tenures and small area, so Memories get the most attention

Integrity Barrier



» Memory integrity

- Focus is on the ability to detect an error and tenure of errors before detection
- Memories with no error detection are biggest eyebrow raiser
 - Examples: L1 caches, TLBs, BTICs
 - Key questions become “are there indirect ways to detect errors”

Vulnerability Inefficiency Barrier



» Conclusion – The inability to use resources because of vulnerabilities, or the adaptations for vulnerabilities can result in inefficiency barriers

Solutions



- » The key to overcoming the barriers is to understand them
- » Barriers
 - Performance loss due to processor architecture choices
 - Primary issue is cache effects spilling from one application to another
 - Note that multi-core may exacerbate the problem due to cache effects from other cores as well as other applications
 - Inefficiencies resulting from processor vulnerabilities
 - Inability to verify functionality
 - Integrity issues due to lack of error detection

Solutions



» Possible solutions to prime the pump

» Cache effects solutions

- Partition caches for each application
 - Similar to memory mapping
 - Supported on some MIPS processors
 - Smaller available cache may offset some of the benefit

» Vulnerability solutions

- Provide the ability to verify features through test (e.g., Built-in-Test)
 - Need the ability to inject errors and verify detection works properly
- Add integrity mechanisms (e.g., checksums or CRCs) to
 - Caches
 - “Cache-like” features – BTIC, TLB, etc.

Questions and Answers



» Contact information

Tim Skutt

Software Architect

GE Aviation

timothy.skutt@smiths-aerospace.com

(616) 241-8645

Power Architecture™
DEVELOPER CONFERENCE

'07

Power.ORG ™