Bringing the Multicore Revolution to Safety-Critical Cyber-Physical Systems

**Motivation**
- Shared hardware like caches & memory introduce timing unpredictability for real-time systems (RTS).
- Worst-case execution time (WCET) analysis for RTS with shared hardware resources is often too pessimistic that the extra processing capacity of multicore systems is negated.
- Different levels of assurance are required for different criticality tasks.

**Problem**
- Recent work has shown that, by combining hardware management and criticality-aware task provisioning, capacity loss can be significantly reduced when supporting real-time workloads on multicore platforms.
- Supporting real-world workloads has not been realized due to a lack of support for sharing among tasks.

**Solution**
- We considered two type of sharing among tasks: shared buffers and shared libraries.
- Controller-Aware Memory Coloring:
  - Colors the entire memory space of heap, static, stack, and instruction segments with locality affinity for controller and bank-awareness.
  - Avoid memory accesses to remote node.
  - Reduce conflicts among banks.
- Supporting Data Sharing in Mixed-Criticality, Multicore Systems:
  - Implemented two inter-process communication (IPC) mechanisms:
    - Produces/consumer buffers (PCBs) and wait-free buffers (WFBs)
  - Proposed three techniques to mitigate interference due to shared memory:
    - Selective LLC Back-off (SBP): Designate a buffer as an uncacheable and allocate it from the Level-C banks.
    - Concurrency Elimination (CE): If a buffer is shared by two tasks at Level A and/or B, assign both tasks to the same core and allocate the buffer from that core’s bank.
    - LLC Locking (CL): Permanently lock a buffer in the LLC.
- Allowing Shared Libraries while Supporting Hardware Isolation:
  - Introduced per-partition library replicas.
  - Implemented a system call to replicate shared libraries.

**Supporting Data Sharing in Mixed-Criticality – Solutions & Results**

**Controller-Aware Memory Coloring – Solutions & Results**

**Conclusions**
- Designed Controller-Aware Memory Coloring techniques to support memory coloring allocations for entire memory space.
- Avoid remote memory node access, and reduce bank-level contention.
- Evaluated on Intel and AMD 16 core platforms and LMX6 quad-core platform.
- Implemented two IPC mechanisms based on shared memory.
- Designed three levels to reduce interference due to sharing data among tasks.
- Proposed partitioning heuristics and a criticality-aware optimization technique for allocating shared buffers.
- Proposed per-partition replicates of shared libraries to allow sharing libraries.
- Conducted micro-benchmarks and large-scale overhead-aware schedulability studies.