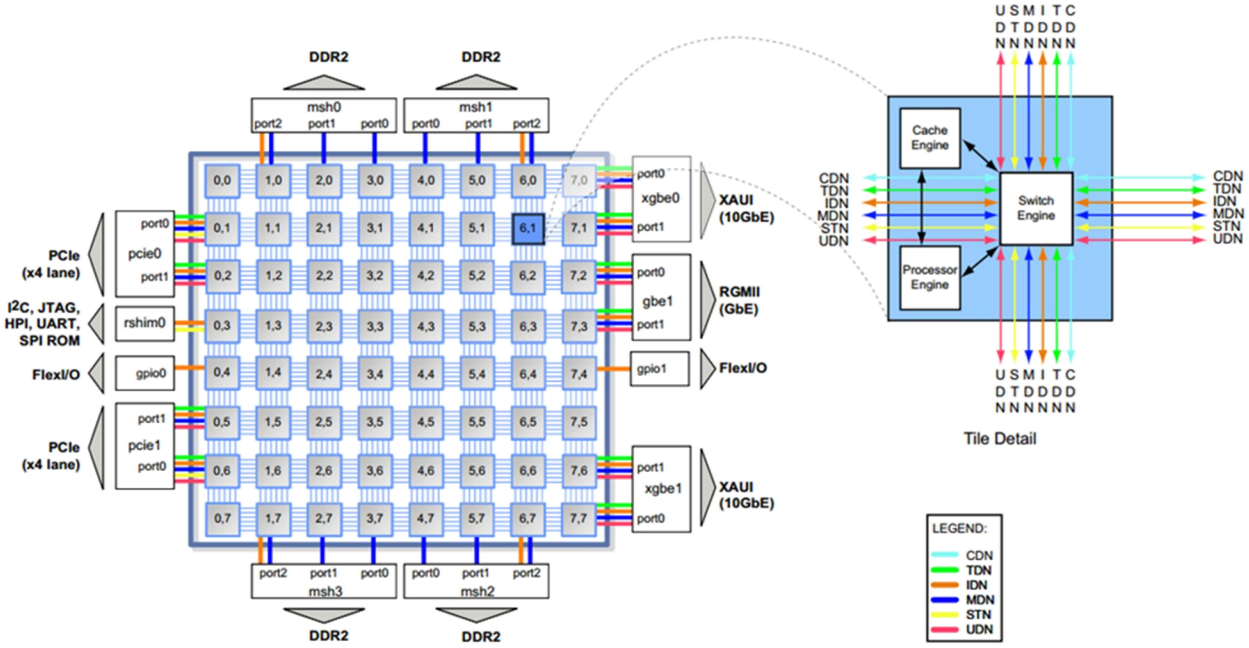
Memory access latency prediction in shared multiple memory controllers

1. Problem description

In safety-critical cyber-physical systems, the usage of multicore platforms has been hampered by problems due to interactions across cores through shared hardware. Shared resource access interference, particularly memory and system bus, is a big challenge in designing predictable real-time systems because its worst case behavior can significantly differ. In this project, I will get a study on the shared multiple memory controllers latency and propose a new virtual and physical address mapping algorithm help us to predict the memory access latency exactly.

In modern CPU, there are many cores and several memory controllers in one processor. For example, in the 2-way SMPs with AMD Opteron 6128, there are 16 cores and 2 memory controllers in 1 node. Another example is there are 4 memory controllers in the TILEPro64 and each of them is responsible for different page address. The architecture of TILEPro64 is as following.



As the graph above, when we want to access a page in memory, each quarter of the page is respectively located at one memory controller. There are totally 64 cores in the TILEPro64. The distances are different between different core access one of the memory controller. We suppose that the distance decides the latency of core accessing memory. So we can find that it is very difficult to predict the memory access latency exactly. If the task is assigned to different core, the memory access latency is different.

Our goal is to design a new virtual and physical address mapping algorithm help us to predict the memory access latency exactly.

2. Configure steps

Step1. Utilize brute-force test program to verify the shared multiple memory access latency existed.

Step2. Check out the current virtual and physical address mapping algorithm.

Step3. Modify the virtual and physical address mapping algorithm.

3. Reference

[1] TILEPROCESSOR ARCHITECTURE OVERVIEW FOR THE TILEPROSERIES

[2] Christopher Zimmer and Frank Mueller. Low Contention Mapping of Real-Time Tasks onto a TilePro 64 Core Processor.

[3] Balasubramanya Bhat and Frank Mueller. Making DRAM Refresh Predictable

[4] Zheng Pei Wu, Yogen Krish, and Rodolfo Pellizzoni. Worst Case Analysis of DRAM Latency in Multi-Requestor Systems

[5] Heechul Yun, Gang Yao, Rodolfo Pellizzoni, Marco Caccamo and Lui Sha. Memory Access Control in Multiprocessor for Real-time Systems with Mixed Criticality

[6] Wei Wang, Tanima Dey, Jack W. Davidson, and Mary Lou Soffa. DraMon: Predicting Memory Bandwidth Usage of Multi-threaded Programs with High Accuracy and Low Overhead

[7] Noriaki Suzuki, Hyoseung Kim, Dionisio de Niz. Coordinated Bank and Cache Coloring for Temporal Protection of Memory Accesses

[8] Hyoseung Kim, Dionisio de Niz, Bjorn Andersson. Bounding Memory Interference Delay in COTS-based Multi-Core Systems.

[9] Heechul Yun, Renato Mancuso, Zheng-Pei Wu, Rodolfo Pellizzoni . PALLOC: DRAM Bank-Aware Memory Allocator for Performance Isolation on Multicore Platforms,