Milestone Report for CS740: Computer Architecture

Configuration Swapping and Caching

Group Members: Suman Kumar Nath (sknath@cs)
Suraj Sudhir (ssudhir@ece)
Project home page: http://www.cs.cmu.edu/~sknath/740

1. Major Changes:

We have slightly shifted one of our goals. Initially our plan was to explore the possibilities of caching configuration on the fabric, find good replacement algorithms for this, and compare performance improvement over various algorithms. We further planned, if time allowed, we would extend our exploration to a multitasking environment. While we still are stick to our primary goal, we now plan to skip looking at the multitasking case, unless time permits. Rather, we now want to explore different configuration caching models for reconfigurable computing and compare their performance.

The rationale for this is that there has been only one study on this fledgling area and practically no work has been done to find out how well different algorithms can be implemented in hardware. In future as reconfigurable computing overcomes its current technological & integration hurdles and enters the mainstream the area of configuration caching will gain much more significance. Therefore we plan to propose some realistic online and offline caching models and evaluate their performance.

If time permits, we also plan to explore the multitasking environment.

2. What we have done so far:

a. We have modified the following tools.
   i. **SUIF (find_custom):** We have hacked the code to get actual size of the RFU's. The original find_fustom did not produce this information as it had bugs that we had to fix.
   ii. **Impact compiler:** We have hacked it to get the sequence of the rfuOp calls (calls to the RFU) that occur in different benchmark programs we are evaluating. Further the total number of times each rfuOp is called is also tabulated.

b. We have implemented some replacement online and offline RFU replacement algorithms for a two-level model (main memory and reconfig fabric) and are implementing a three-level model (main memory, configuration cache, fabric). We assumed two models: *loading with relocation* and *loading with relocation+defragmentation*.. Further we are working on a partial loading model where only part of a configuration is loaded at a time if there isnt space to load all of it.

3. Meeting milestone:

In fact, we have done something beyond our original milestone. Initially we planned to create a 2-level cache model and develop replacement algorithms for it. We have completed this phase. Further, we have developed a theoretical model of the problem, and already started implementing a 3-level caching model and a partial configuration loading model. We have found some interesting performance results and finding the way to implement the algorithm in actual hardware.

4. Surprises

a. The *find_custom* (SUIF) tool part of the SUIF/Impact test suite had some bug :). It was unexpected and we had to invest a significant portion of time to fix it.

b. Offline replacement algorithm for variable size page size is NP Complete, although the same problem for fixed size is polynomial.

c. Compaction (which makes better utilization of chip area) does not ALWAYS mean improvement of performance when page size is variable.

5. Revised Schedule (from now)

Week 1 : Complete implementation of two different modes: relocation+defragmentation mode, partial load/eviction mode. Extend to 3 level configuration with and without inclusion.

Week 2 : Complete all algorithms and data collection start writing the report summarizing the statistics collected. If possible try for prefetching study. Possibly describe a multitasking architecture and study how much it costs to swap configurations in and out, both in terms of time and space.

Week 3 : Complete and submit report. Create presentation for poster session.

6. Resource Needed:

We don't think we need any other new resources or tools.