ECE 511 Assignment 1 Part 2

Andrew Smith

February 5^{th}, 2019

1 Introduction

For this part if the assignment I created a pseudo-instruction in gem5 to enable me to evaluate the cache performance of three different systems using gem5's Full System (FS) mode. I evaluated three different L2 cache sizes on a dual CPU system. These systems are described in Section 2. The test program was a NxN matrix multiply with varying tile sizes to find a tile size that utilizes the cache hierarchies most effectively.

2 Systems

2.1 64kB L2 System

The first system that I configured was a benchmark system. This system was a dual CPU ARM processor with a $16kB$ L1 instruction cache, $64kB$ L1 data cache, and a single $64kB$ L2 cache. The topology is shown below in figure 2.1.

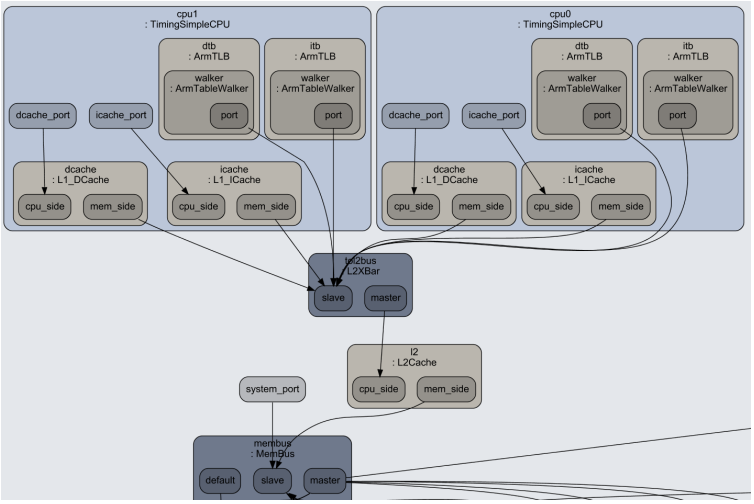

Figure 1: 64kB L2 Cache System

2.2 128kB L2 System

The second system had a dual CPU ARM processor with a $16kB$ L1 instruction cache, $64kB$ L1 data cache, and a single $128kB$ L2 cache (Figure 2.2).

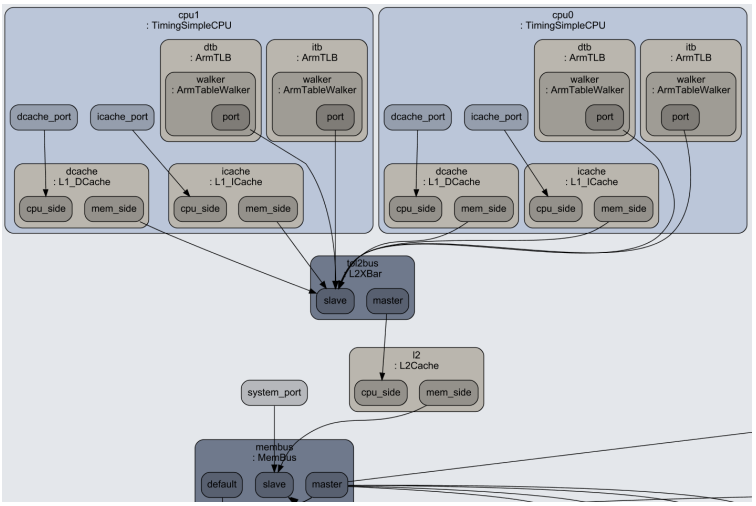

Figure 2: 128kB L2 Cache System

2.3 256kB L2 System

The third system had a dual CPU ARM processor with a $16kB$ L1 instruction cache, $64kB$ L1 data cache, and a single $128kB$ L2 cache (Figure 2.3).

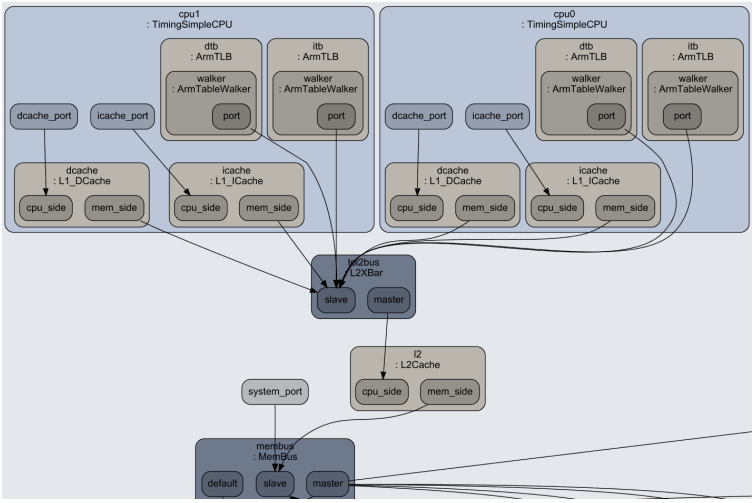

Figure 3: 256kB L2 Cache System

3 Results

64kB L2 Cache 128x128 Matrix Multiply					
Tile Size	Miss Rate	Hit Rate	Cache Writebacks	IPC (inst/tick)	Simulation Time
1	0.69754	0.30246	357624	0.00185998085	156153638500
2	0.69750	0.30250	357624	0.00186001357	155989259500
4	0.69753	0.30247	357653	0.00186003405	155931567500
8	0.69753	0.30247	357720	0.00186005004	155908915500
16	0.69730	0.30270	357820	0.00186005928	155900073000
32	0.69696	0.30304	358091	0.00186005600	155895158500
64	0.69826	0.30174	358208	0.00186005383	155895009000

Figure 4: Simulation Statistics 64kB L2 System

128kB L2 Cache 128x128 Matrix Multiply					
Tile Size	Miss Rate	Hit Rate	Cache Writebacks	IPC (inst/tick)	Simulation Time
1	0.58781	0.41219	334687	0.00185997814	156153638500
2	0.58777	0.41223	334686	0.00186001473	155989259500
4	0.58779	0.41221	334718	0.00186004333	155931567500
8	0.58779	0.41221	334813	0.00186005084	155908915500
16	0.58779	0.41221	335357	0.00186005734	155900073000
32	0.58804	0.41196	335552	0.00186005865	155895158500
64	0.58869	0.41131	335780	0.00186005585	155895009000

Figure 5: Simulation Statistics 128kB L2 System

256kB L2 Cache 128x128 Matrix Multiply					
Tile Size	Miss Rate	Hit Rate	Cache Writebacks	IPC (inst/tick)	Simulation Time
1	0.43786	0.56214	289594	0.00185998452	156153638500
2	0.43784	0.56216	289600	0.00181938906	155989259500
4	0.43780	0.56220	289627	0.00186003711	155931567500
8	0.43783	0.56217	289722	0.00186005419	155908915500
16	0.43893	0.56107	290906	0.00186005777	155900073000
32	0.43784	0.56216	290477	0.00186006252	155895158500
64	0.43827	0.56173	290853	0.00186006107	155895009000

Figure 6: Simulation Statistics 256kB L2 System

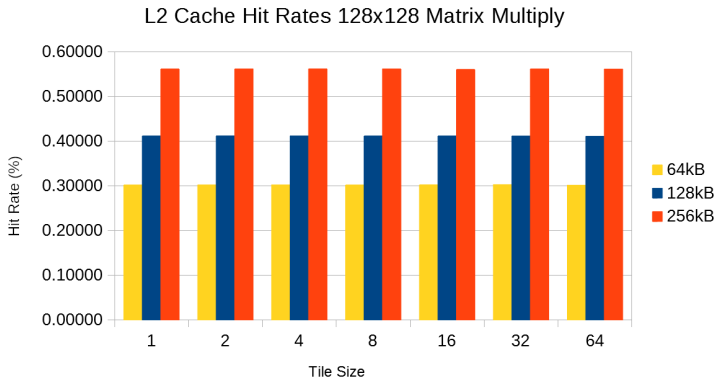

Figure 7: L2 cache hit rates across all systems

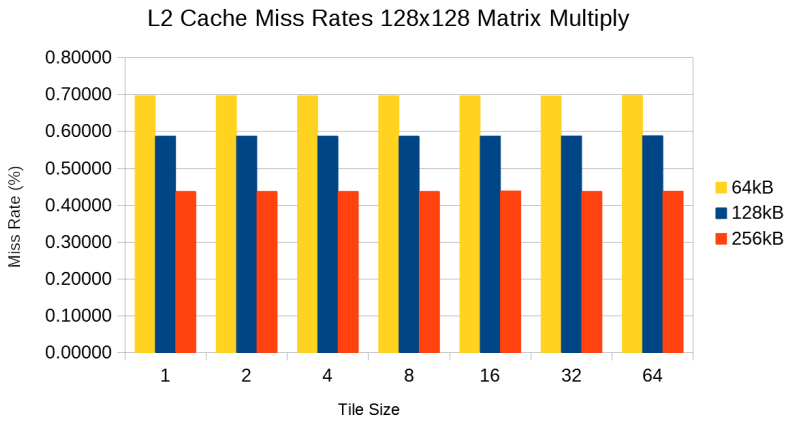

Figure 8: L2 cache miss rates across all systems

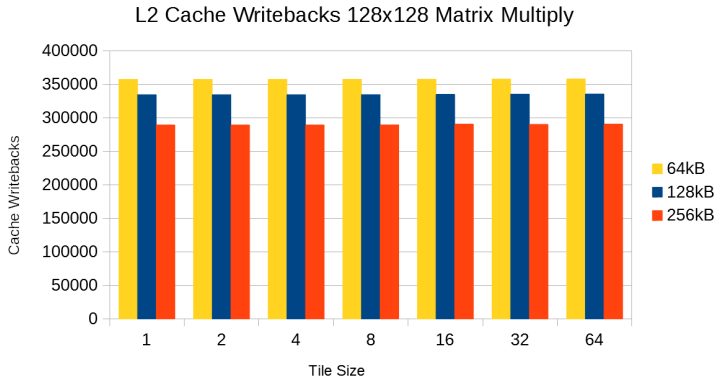

Figure 9: L2 cache writebacks across all systems

4 Conclusion

As expected, the system with the largest 256kB L2 Cache was able to perform better than the systems with the smaller L2 caches. The larger cache meant the system could fit more rows and columns from the matrices in the cache so it had to make fewer demand requests to the main memory (Figures 3, 3, 3. What I found interesting was that there was very little variance among the miss rates among the different tile sizes of the matrix. I'm not sure whether this is due to user error or not testing enough matrix size or tile size combinations. Generally tile sizes between 4 and 32 performed the best, although the change in miss rate of a specific cache size could be considered statistically insignificant.