ILP Project Report

Ahmed H. Mahmoud

	Processor 1	Processor 2	Processor 3	Processor 4
Performance	1.08	1.425	**1.52**	
Power (watts)	35.08	34.88	**37.12**	
Area (units)	354	352	424	
Cost ($)	**16.4**	16.15	28.23	
Performance/area(%)	0.305	0.404	0.358	
Performance/cost(%)	6.58	**8.82**	5.38	
CPI	**0.96**	0.73	0.68	

Figure 1: Metric table for the four processors

Processor One (lowest cost with CPI of 1 or less): To achieve CPI of one or less, we must issue more than one instruction every clock cycle. Thus, we would have to increase the fetch width and consider out-of-order execution. This is would deem useless if we do not increase the issue and decode width as well. With experimentation, we found that the best choice is to have fetch width, issue width and decode width of 4. Additionally, we should increase the reorder buffer accordingly. Since increase in the reorder buffer size has low increase in cost, we choose to have reorder buffer of size 16. In order to make use of the multiple issued instructions, we should increase also the ALUs. Looking at the benchmark programs upon which our CPI will be calculated (by taking the geometric mean), we would find that floating point instructions are minimal. Floating point instructions in the four programs are less than 1% expect for anagram program it is around 5% of the instructions. Thus, it is useless to add extra floating point ALUs and thus only additional integer ALUs will be considered. All better caches options are costly expect with adding the associativity (2-way) to the default 8K cache for both instruction and data caches and thus it is considered in out design. Another insight we get from looking at the benchmark programs' statistics is the amount of load/store instructions. They account for at least 30% of the instructions with exception for compress program in which they account for nearly 80% of the instructions. Thus, any increase in the load/store queue would be beneficial. Since increasing the load/store queue size is not that expensive, we considered a load/store queue of size 16. Other options are left as default to maintain the cost as low as possible. The processor CPI is 0.96 and costs. Table 1 shows the important metric of the processor. The insights we get from looking into the benchmark programs shall be considered in the other three processors.

Processor Two (maximize performance/cost): Since the cost increases proportional to the area^3, we should only consider adding/modifying hardware pieces such that one unit increase in area would give three folds increase in performance. Thus, we will not consider increasing the cache size here since the options given for caches are associated with high cost. We will take our base processor as Processor One and start modifying it. Since conditional branches accounts for about 15% of the instructions, we can further improve the performance (CPI) by considering improving the branch predication scheme. We can use the Branch Target Buffer that stores the address of conditional branches target in a buffer. When tested, the CPI was reduced by around 20% to 40% per program. Additionally, we can eliminate the pieces of greater area (cost) such as the decode width which can be reduced to 2 with very small performance degrading. The same thing goes for load/store queue which can be reduced to 8. The processor has performance of 1.425 with cost of 16.15 $.

Processor Three (maximize performance/power): In order to maximize the performance with focus on the power rather than the cost, we can now utilize the cache options since an increase in cache would increase the area substantially but that has low impact on the power (factor of 0.2 of the area increase). Given the options of L1 and L2 caches, the default L2 data and instruction caches are large enough and increasing this size will not have huge impact on the performance. This was verified by experimenting and found a very marginal improvement (less than 2% decrease in CPI) when doubling the L2 cache for both the data and instruction. For the L1 cache, since increasing the associativity would further decrease the miss rate, we to choose to keep the 2-way associative of both the data and instruction L1 cache. To our surprise, going for 4-way associative decreased the performance; higher CPI for the four benchmark programs. We increase the cache size to be 32K for both instruction and data cache. We utilized the BTB for to improve the performance even further (as seen in Processor Two). All other configurations are the same as Processor One. The processor power is 37.12 watts and has performance of 1.52 which is better Processor One (for the same increase in power).