Rithvik Shetty
12/15/23
CMPEN 432 001L
John Sampson

CMPEN 431 Design Space Exploration Final Report

Overview:

For this project, I began with implementing my validation function. This function is designed to narrow down the over 1.4 billion unique combinations that are possible within this project to a subset that is plausible to accomplish. The specifications this project required were the following: matching il1 and dl1 block sizes with the ifq size, keeping ul2 size and block size to be twice the size of il1. There were also a large set of size and latency links that had to be implemented for il1, dl1 and ul2.

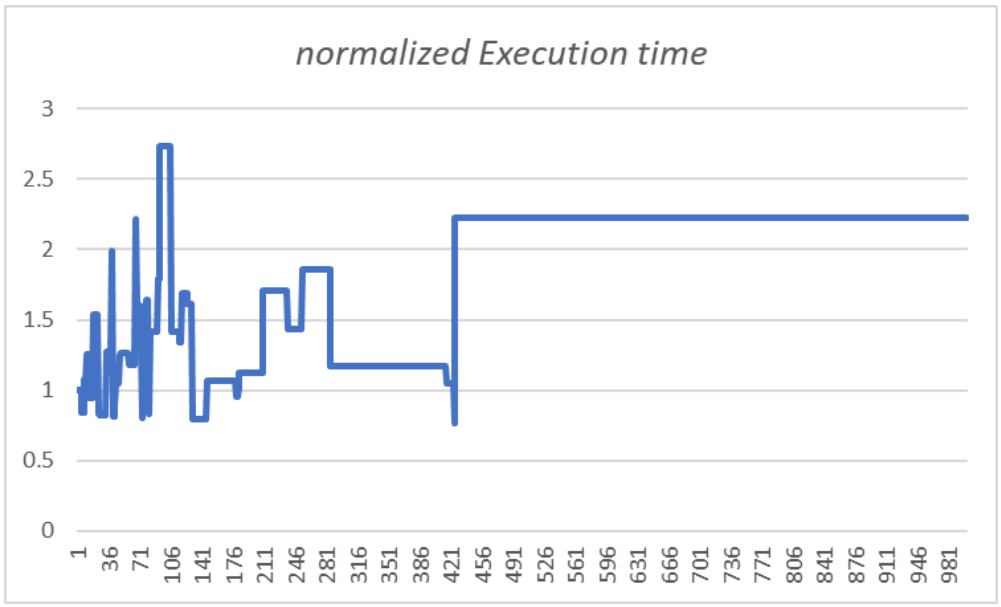
After the implementation of this validation function, the original 1.4 billion combinations have now been cut down to a far smaller set of possible combinations. However, it is still far too large to test every one of these, so we must create a proposal function which will test a small sample of specific 1000 cases. For my proposal function, I had to orient it towards two goals. The first was to ensure the best possible performance, while the second was to try to achieve the least energy consumption.

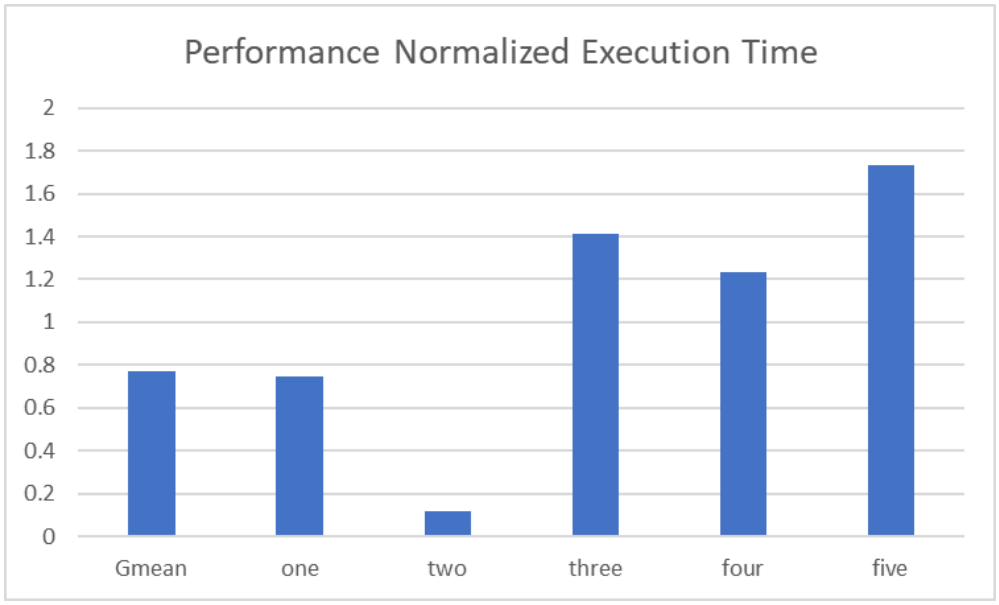
For my performance oriented proposal function, I decided to focus entirely on the cache and different combinations of the cache implementation in each simulation. In order to accomplish this, I set the following dimensions constant: width at 1, fetch-speed at 1, scheduling at Out-of-order, LSQ size at 4, memports at 1, and the branch predictor at perfect. I kept the branch predictor constant at “perfect” in order to test the best possible performance situation for each simulation. I also kept the others at the given base case in order to most accurately test the impact of the modification of the cache-related dimensions. I then took the following dimensions at had them be modified for each simulation run: RUU Size, L1 D$ Sets, L1 D$ Ways, L1 I$ Sets, L1 I$ Ways, Unified L2 Set, Unified L2 Blocksize, Unified L2 Ways, TLB Sets, L1 D$ Latency, L1 I$ Latency, Unified L2 Latency. The reason I focused on the cache in order to try to optimize the performance is that I knew from class lectures that the orientation of different caches and how they improve temporal and spatial locality was important towards optimizing performance, as caches can allow computers to bypass wasting time reaching into the main memory for commonly used data.

For my efficiency oriented proposal function, I decided to take a different approach and focus on the other aspects of the dimensions and how they would influence the energy consumption output of the simulations. I set the following dimensions constant: L1 I$ Sets at 32, L1 I$ Ways at 1, Unified L2 Sets at 1024, Unified L2 Block size at 64, Unified L2 Ways at 4, L1 D$ Latency at 1, L1 I$ Latency at 1, Unified L2 Latency at 8, and Branch Predictor at Combined (Tournament predictor). The reason I kept the Branch Predictor constant at Combined is because I wanted to test the efficiency with the most rigorous possible branch prediction in order to determine the most energy efficient model. I decided to have the following dimensions to be modified during each simulation run: width, fetch-speed, scheduling, RUU size, LSQ size,

memports, L1 D$ Sets, L1 D$ Ways, and TLB Sets. For energy efficiency, I decided to keep the cache the same baseline in order to investigate how changing the scheduling, size of different queues and number of sets and ways would influence the energy consumption. I also knew from our lecture slides that the spatial locality impacts the energy usage, which is something I took into account when designing my efficiency proposal function.

Performance Analysis:

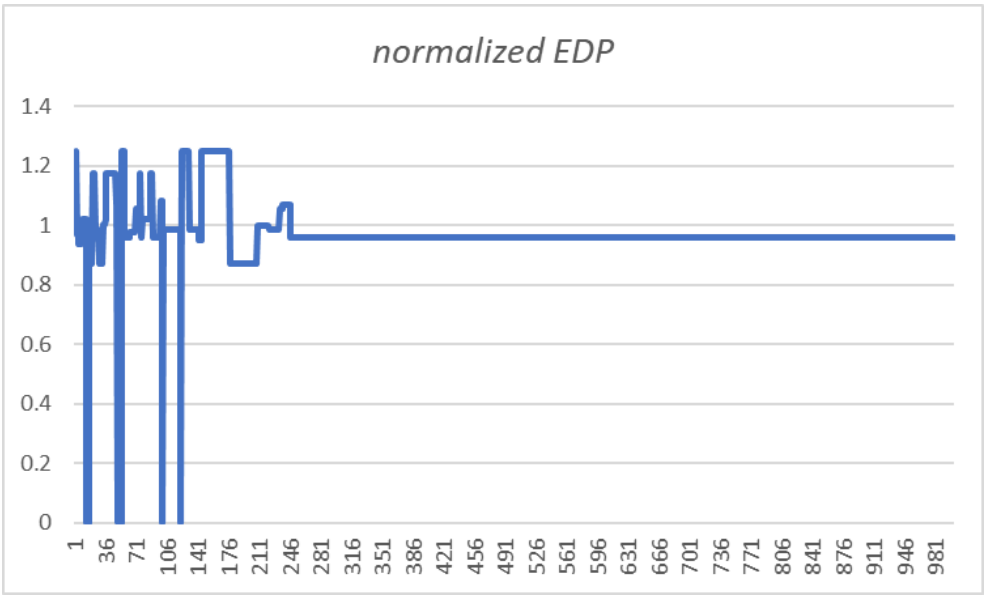


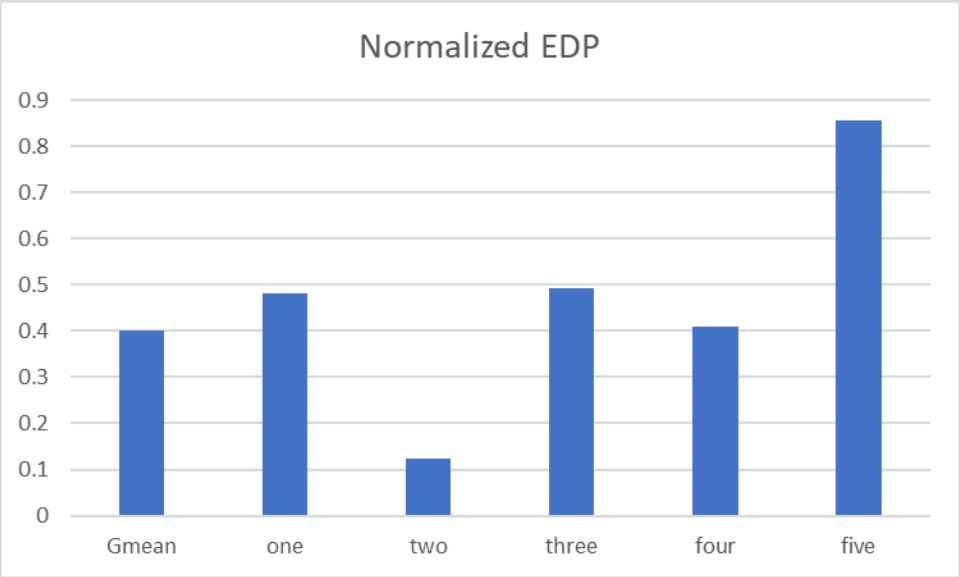


DIMENSION	Index [0]	Index [1]	Index [2]	Index [3]	Index [4]	Index [5]	Index [6]	Index [7]	Index [8]	Index [9]
0: Width	1	2	4	8						
1: Fetchspeed	1	2								
2: Scheduling	In-order	Out-of-order								
3: RUU Size	4	8	16	32	64	128				
4: LSQ Size	4	8	16	32						
5: Memports	1	2								
6: L1 D$ Sets	32	64	128	256	512	1024	2048	4096	8192	
7: L1 D$ Ways	1	2	4							
8: L1 I$ Sets	32	64	128	256	512	1024	2048	4096	8192	
9: L1 I$ Ways	1	2	4							
10: Unified L2 Sets	256	512	1024	2048	4096	8192	16384	32768	65536	131072
11: Unified L2 Blocksize	16	32	64	128						
12: Unified L2 Ways	1	2	4	8	16					
13: TLB Sets	4	8	16	32	64					
14: L1 D$ Latency	1	2	3	4	5	6	7			
15: L1 I$ Latency	1	2	3	4	5	6	7			
16: Unified L2 Latency	5	6	7	8	9	10	11	12	13	
17: Branch Predictor	Perfect	NotTaken	Bimodal, 2K entry	2 Level GAp: 1 8-entry history, four 256 entry bimodal tables	2 Level PAg: 4 8-entry histories, one 256 entry bimodal table	Combined (Tournament predictor)				

The design that had the best performance was the one circled above. Some takeaways I had from this was that the maximum possible RUU size helped performance the most. While this may not be logistically efficient energy-wise, it definitely would have a major impact since the Out-of-order scheduling was also selected so being able to perform as many instructions out of order as possible would be extremely valuable to maximizing the performance. Another takeaway I had from this design is that block size for the various levels of caches were maximized while the ways was minimized. This is another design that wouldn't be efficient in a functional design, however, it makes sense that this will maximize the performance, as more information is able to be passed with each block.

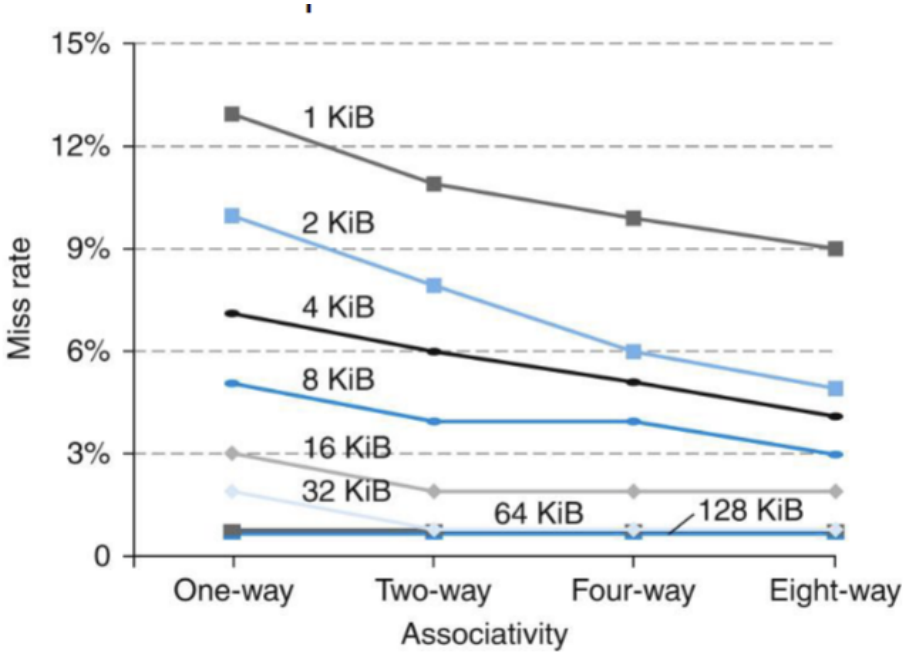
Efficiency Analysis:





DIMENSION	Index [0]	Index [1]	Index [2]	Index [3]	Index [4]	Index [5]	Index [6]	Index [7]	Index [8]	Index [9]
0: Width	1	2	4	8						
1: Fetchspeed	1	2								
2: Scheduling	In-order	Out-of-order								
3: RUU Size	4	8	16	32	64	128				
4: LSQ Size	4	8	16	32						
5: Memports	1	2								
6: L1 D$ Sets	32	64	128	256	512	1024	2048	4096	8192	
7: L1 D$ Ways	1	2	4							
8: L1 I$ Sets	32	64	128	256	512	1024	2048	4096	8192	
9: L1 I$ Ways	1	2	4							
10: Unified L2 Sets	256	512	1024	2048	4096	8192	16384	32768	65536	131072
11: Unified L2 Blocksize	16	32	64	128						
12: Unified L2 Ways	1	2	4	8	16					
13: TLB Sets	4	8	16	32	64					
14: L1 D$ Latency	1	2	3	4	5	6	7			
15: L1 I$ Latency	1	2	3	4	5	6	7			
16: Unified L2 Latency	5	6	7	8	9	10	11	12	13	
17: Branch Predictor	Perfect	NotTaken	Bimodal, 2K entry	2 Level GAp: 1 8-entry history, four 256 entry bimodal tables	2 Level PAg: 4 8-entry histories, one 256 entry bimodal table	Combined (Tournament predictor)				

For the energy efficiency design, the dimensional combination that had the best performance is the one above. The first thing I noticed was how compared to the Out-of-order design from the performance maximized run, this kept the RUU size to 32, something that would definitely help improve the energy consumption by not having a huge RUU, but it would also help improve locality, which in turn improves the energy consumption. Another difference from the performance run is that the ways are set much higher. In the performance design space, the ways was minimized to be as low as possible, but here we can see it is bigger, which would allow for the miss rate to improve greatly.



As we learnt in this class, and the figure above, as the associativity-ways approaches four-way and eight-way associativity, the miss rate is greatly improved and evens out, which would diminish energy consumption by the CPU.

Conclusion:
Overall, after analyzing the two approaches, the more realistic processor seemed to be the one designed by the energy efficient approach. While the energy efficient approach prioritized energy, it seems that this coincides with performance increases in a lot of areas, which is the most useful for designing computer processors. The performance simulations gave a lot of processors that gave amazing performance, however they are too unrealistic to implement in a real world scenario.