Design-Space-Exploration

Introduction:

We used Simple Scalar as the evaluation engine in this project to explore an 18-dimensional processor pipeline and memory hierarchy design space in order to select the best performing design under two different optimization functions - the best performance-oriented design and the most energy-efficient design. The design space was explored using a workload of five benchmarks and a design point restriction of 1000. The framework given enabled quick exploration of the design space with little simulation time. Instead of a brute force exhaustive search, the framework enabled us to create efficient heuristics. Furthermore, designating numerous design possibilities as incorrect aided in speeding up the research process.

Heuristic:

I developed a heuristic way to explore the design space relatively efficiently. My way of heuristic involved changing some dimensions that is stable and changing some dimension with the most effect, with all other dimensions fixed. This approach allowed us to explore more design points and find a better design point. Additionally, we kept calling our simple heuristic function and used the output from the last call as the baseline for the next call. This allowed us to improve the current best design points and find better baselines. We found that our heuristic was effective and allowed us to explore the design space efficiently within the given limitation.

- The Best Performance
  - bestEXEConfig {0,2,1,5,3,1,2,0,6,0,3,2,3,4,0,3,5,5}
  - bestEDPConfig {0,2,1,5,3,1,2,0,6,0,3,4,3,4,0,3,5,5}
- The most energy efficient
  - bestEXEConfig {0,2,1,4,3,1,2,0,4,2,3,0,3,4,1,3,6,5}
  - bestEDPConfig {0,2,1,4,3,1,2,0,4,1,3,3,3,4,1,3,6,5}

Reasoning:

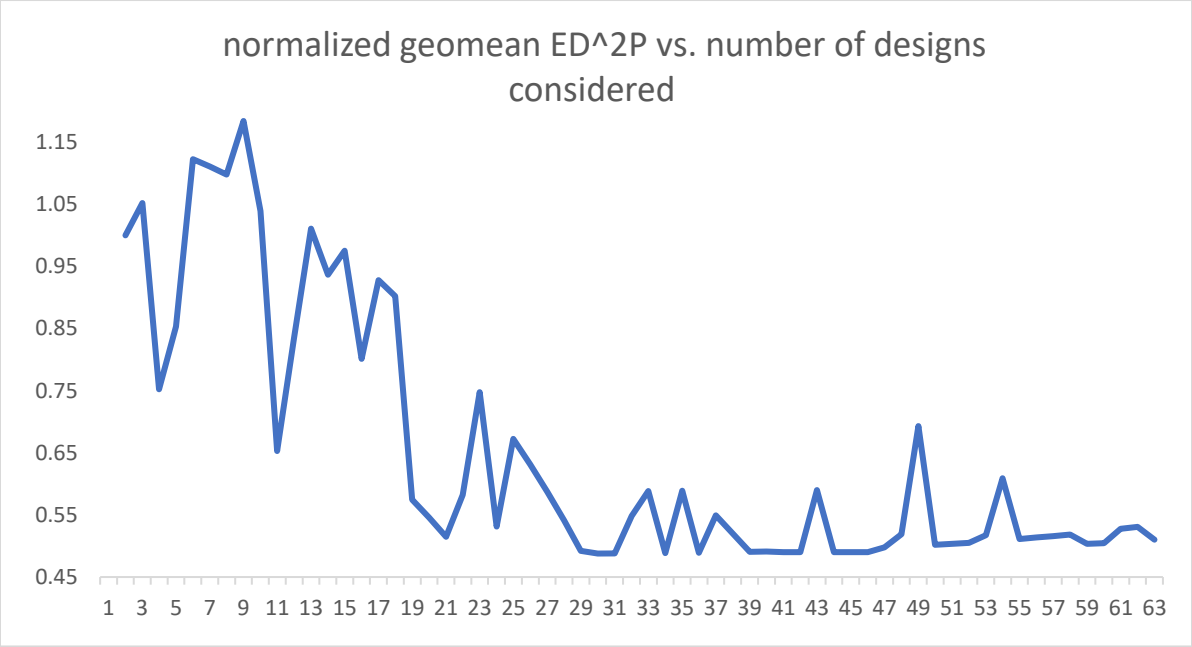
Parameter	**Performance**	**Energy(EDP)**
Width	Value:1 Reason: Too much width will result in more data hazard to occur which will end up in both reduced time and power.	Value:1 Reason: Too much width will result in more data hazard to occur which will end up in both reduced time and power.
		Value:2

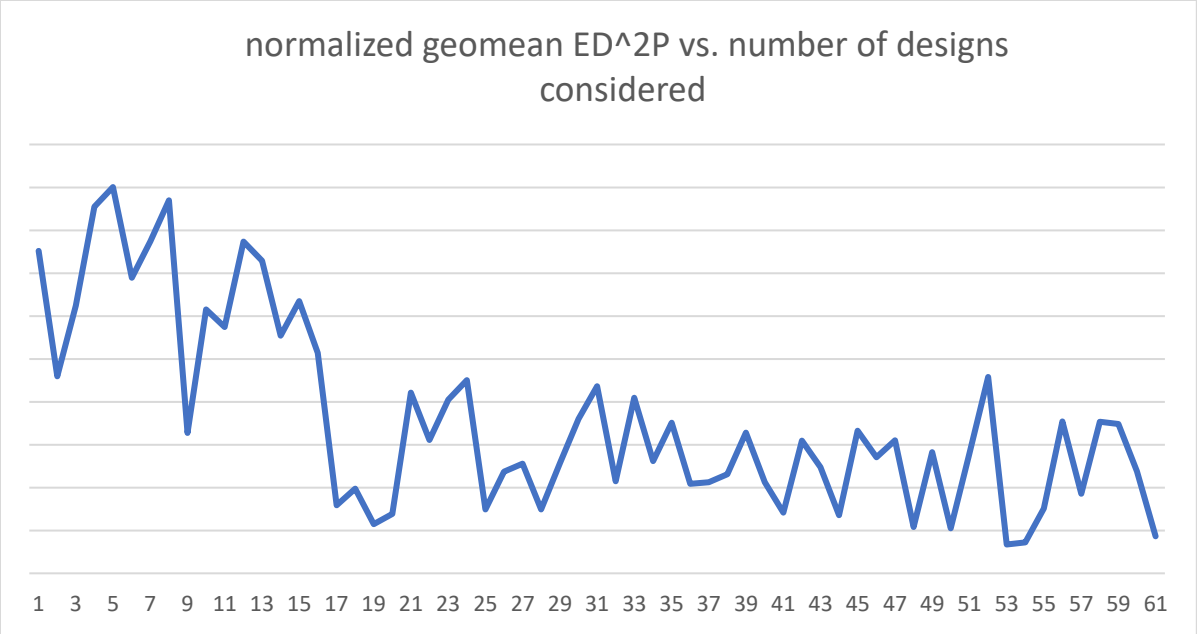
Parameter	Performance	Energy(EDP)
Fetch speed	Value:2 Reason: The speed will affect the entire program running speed, which if the fetch is slow not even performance goes down the energy also wasted	Reason: The speed will affect the entire program running speed, which if the fetch is slow not even performance goes down the energy also wasted.
Scheduling	Value: Out of order Reason: Out of order scheduling will almost certainly faster than in order	Value: Out of order Reason: By using out of order schedule we will be able to avoid a lot of the to deal with stall and revokes.
Ruusize	Value: 128 Reason: The larger the Ruu size is, the better the speed since no more replacing	Value: 64 Reason: The larger the Ruu size it will also take a lot of memory to store them
Lsqsize	Value: 32 Reason: The larger the LSQ size is, the better the speed since no more replacing	Value: 32 Reason: The larger the LSQ size is, the better the speed since no more replacing
Memport	Value: 2 Reason: The more the memport are there will provide more ways for the process to be running	Value: 2 Reason: The more the memport are there will provide more ways for the process to be running
Dl1sets	Value: 128 Reason: The larger the cache set is the less the miss rate will get thus better speed	Value: 128 Reason: The larger the cache set is the less the miss rate will get thus better speed
Dl1assoc	Value: 1 Reason: The higher association will reduce the amount of time for huge penalty	Value: 1 Reason: Save energy since no more look up needed.
Il1sets	Value: 8192	Value: 8192

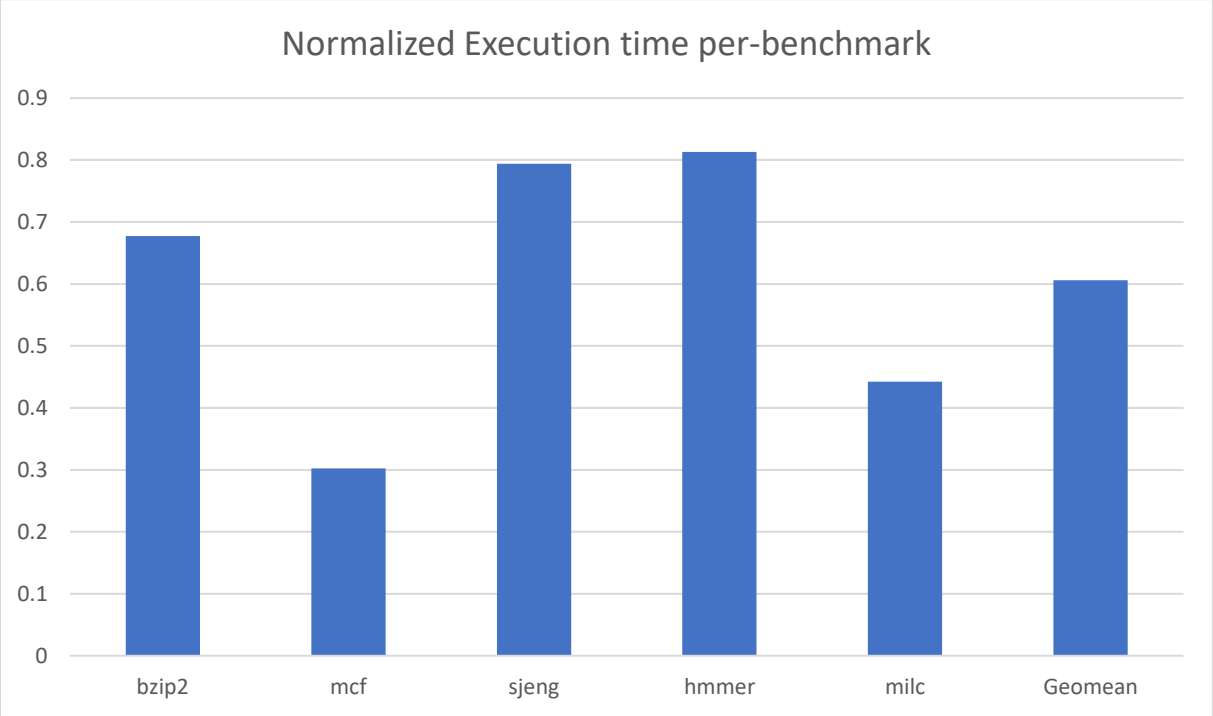
Parameter	Performance	Energy(EDP)
	Reason: As the sets increases the there are more instruction at a time	Reason: As the sets increases the there are more instruction at a time
Il1assoc	Value: 1 Reason: As the association of the instructions getting choosing the right number of sets	Value: 1 Reason: As the association of the instructions getting choosing the right number of sets
Ul2sets	Value: 2048 Reason: L2 should be certainly large enough to take possible penalty from miss on l1.	Value: 1024 Reason: L2 should be certainly large enough and not as huge to take possible penalty from miss on l1, and save some power if possible.
Ul2block	Value: 64 Reason: Block size that reduce the global AMAT	Value: 128 Reason: Block size that reduce the global AMAT, Higher since greatly reduce energy cost by saving amat
Ul2assoc	Value: 8 Reason: Low miss rate	Value: 8 Reason: Low miss rate
Tlbsets	Value: 64 Reason: The size of TLB can greatly reduce capacity miss	Value: 16 Reason: Smaller Tlb for saving energy but not lowest
Dl1lat	Value: 1 Reason: This is calculated by Dl1 block, set and associations.	Value: 1 Reason: This is calculated by Dl1 block, set and associations.
Il1lat	Value: 6 Reason: This is calculated by Il1 block, set and associations.	Value: 6 Reason: This is calculated by Il1 block, set and associations.
Ul2lat	Value: 11 Reason: This is calculated by Ul2 block, set and associations.	Value: 10 Reason: This is calculated by Ul2 block, set and associations.

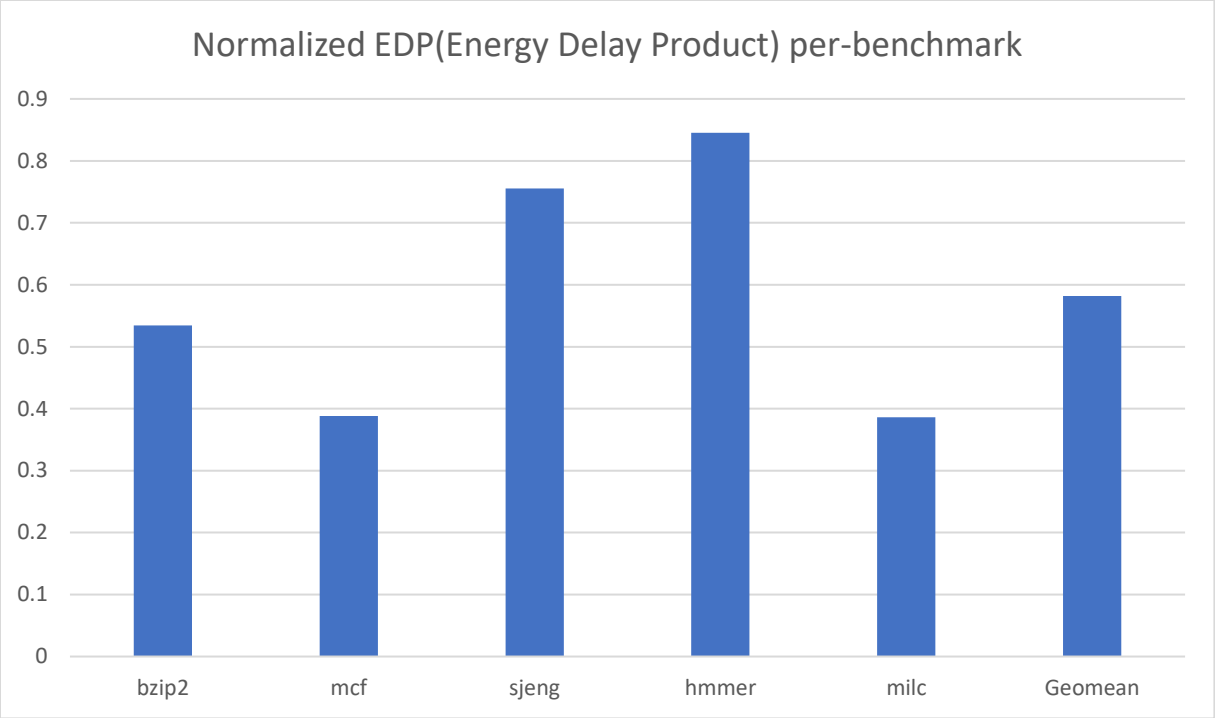
Parameter	Performance	Energy(EDP)
Branchsettings	Value: -bpred: comb 1024 Reason: Comb is doing really well in term of accuracy comparing with other Branch predictors which is why it is the best for both speed and energy.	Value: -bpred: comb 1024 Reason: Comb is doing really well in term of accuracy comparing with other Branch predictors which is why it is the best for both speed and energy.

Charts for optimizing Performance:

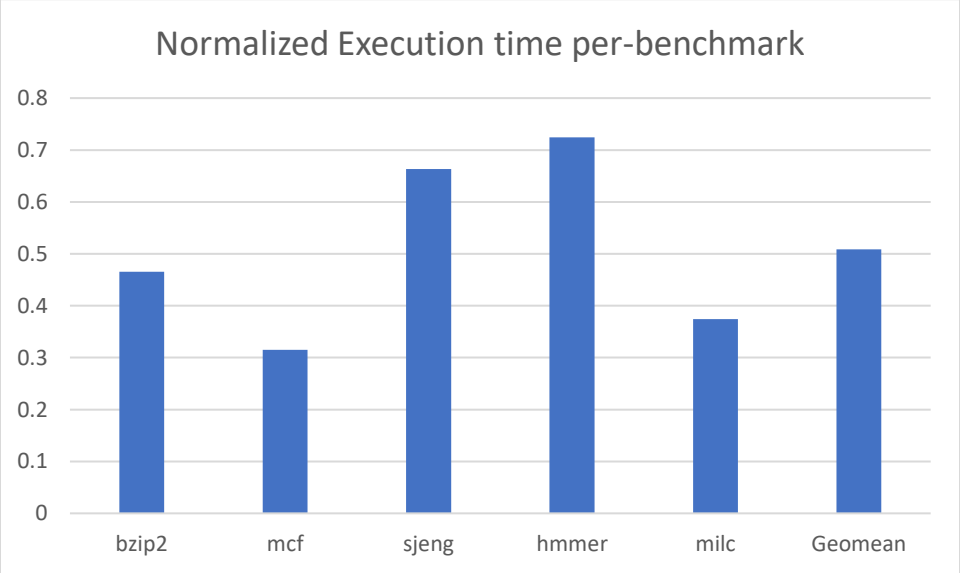


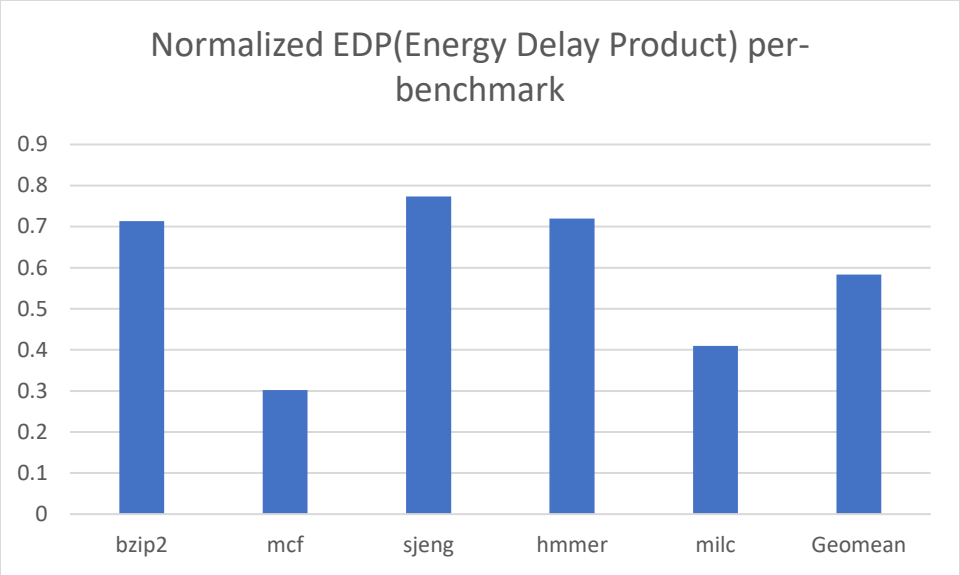

normalized geomean ED^2P vs. number of designs considered
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61


Normalized Execution time per-benchmark
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
bzip2
mcf
sjeng
hmmer
milc
Geomean



Charts for optimizing Energy:





Conclusion:

We will learn a lot about design space exploration and the use of heuristics to explore huge design spaces through this assignment. I recognized that complete exploration of huge design spaces is not possible and that heuristics must be used to reduce the design space and prioritize consideration of more reasonable design ideas first. We also learnt about how design aspects like width and scheduling affect clock cycle time, power, and energy. In-order processors with a smaller fetch width have a shorter clock cycle time, less core leakage power, and use less energy per committed instruction. We also learnt about cache size and how it influences access energy and leakage, with lower cache sizes resulting in less access energy and leakage.

Lastly, utilizing a heuristic technique, I effectively explored the 18-dimensional CPU pipeline and memory hierarchy design space and discovered optimal performance-oriented and energy-efficient solutions. There is a lot written about the effects of design factors on performance, power, and energy, as well as the use of heuristics to efficiently explore design space. My findings validate several intuitions established in class and prescribed readings while also providing new insights into the implications of design factors on performance, power, and energy.