CSE 431 | Final Project Report

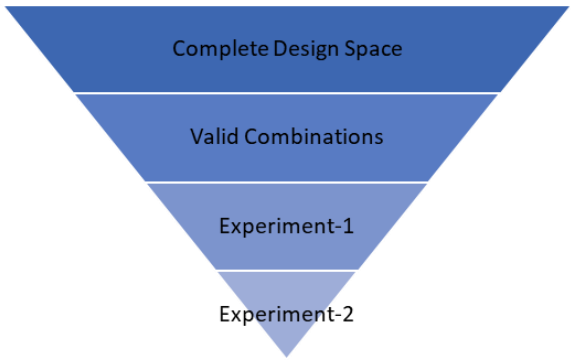
Rishabh Bhatt

AIM:

In this project we aim to use SimpleScalar as an evaluation engine to perform a design space exploration over 18 dimensions/parameters. Since an exhaustive exploration over a sample space of **1,481,421,312,000** would not be efficient our aim is to strategically explore the design space to select the best performing design under two different optimization functions: best ***performance-oriented design*** and most ***energy-efficient design.***

EXPLORATION STRATEGY:

As mentioned above an exhaustive search over the design space would not be efficient. Moreover, many combinations would not be valid under the given model constraints. To this end we create a **validation function** which checks for valid configurations before they can be simulated. This helps us reduce our design space to a smaller set of valid combinations only.



EXPERIMENT-1:

In our first experimental simulation we use our knowledge of performance and efficiency tradeoffs to heuristically narrow the search space. To do this we select the following subsets (one for each optimization function) from the 18 parameters to construct our search space. We explore the subspace created by these parameters while fixing the remaining parameters to the default state. This allows us to substantially reduce the search space which we randomly explore.

Performance Optimization		
Index	Parameter	Combinations
0	width	4
1	fetchspeed	2
2	Scheduling	2
3	RUU Size	6
4	LSQ Size	4
8	iL1 Sets	9
9	iL1 Ways	3
17	Branch Predictor	6
Search Space = 62,208		

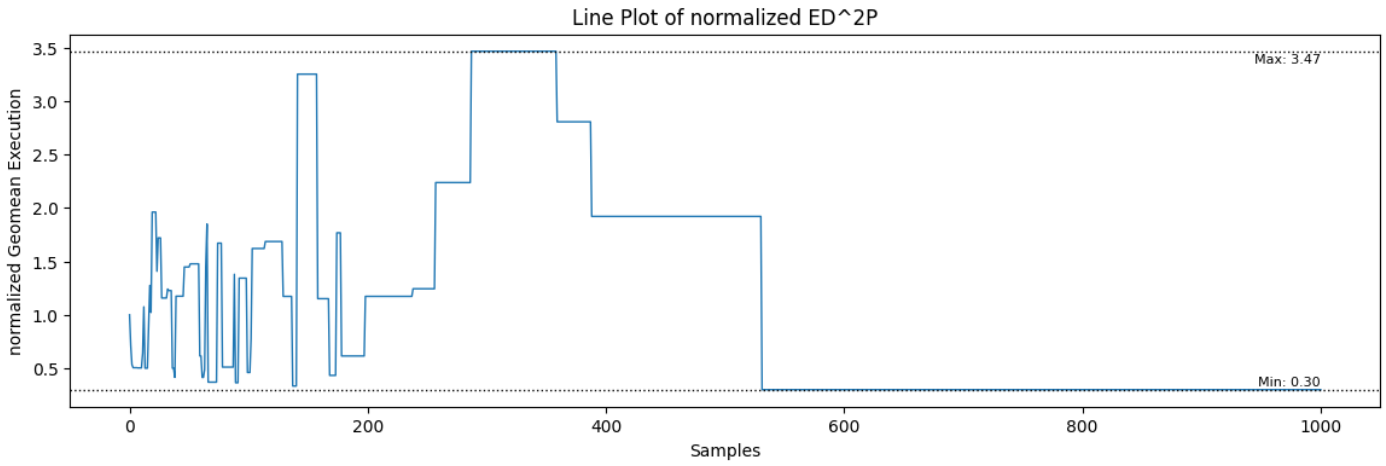
Energy Optimization		
Index	Parameter	Combinations
0	width	4
2	Scheduling	2
6	dL1 Sets	9
7	dL1 Ways	3
10	L2 Sets	10
11	L2 Blocksize	4
12	L2 Ways	5
17	Branch Predictor	6
Search Space = 259,200		

PARAMETERS SELECTION:

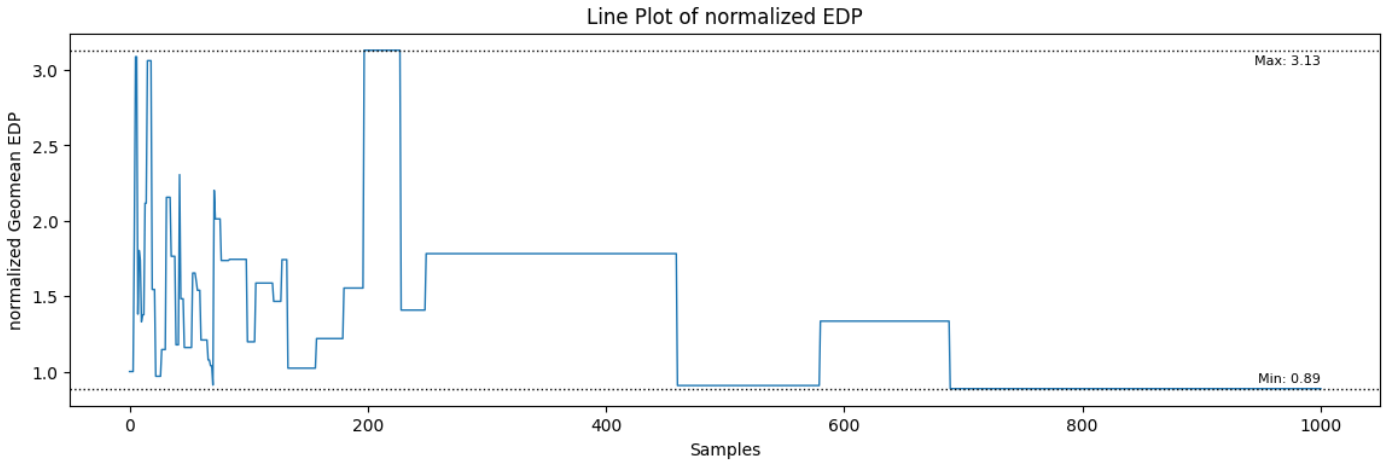
- **Width (BOTH):** This parameter determines the **width of the processor's datapath**, which determines the number of bits that can be processed simultaneously. Increasing the width can lead to higher performance by enabling the processor to perform more operations in parallel. However, this can also increase energy dissipated due to the larger circuitry required to handle the wider datapath. Moreover, in our framework width is also linked to L1 block size which affects both functions.
- **Scheduling (BOTH):** This parameter determines the **scheduling policy** used by the processor to issue instructions to the execution units. In-order scheduling can simplify the design of the processor and reduce energy dissipation. Out-of-order scheduling can increase performance but requires more complex circuitry and can dissipate more energy.
- **Branch Predictor (BOTH):** This parameter determines the branch prediction scheme used by the processor. Better branch prediction can lead to higher performance by reducing pipeline stalls caused by branch mispredictions. However, more complex branch prediction schemes can also dissipate more energy.
- **RUU Size (Performance):** This parameter refers to the **size of the register file** used by the processor. Increasing the register file size can increase performance by enabling more data to be stored locally and reducing the number of memory accesses required.
- **LSQ Size (Performance):** This parameter refers to the **size of the load-store queue**, which is used to buffer memory operations. Increasing the size of the load-store queue can increase performance by allowing the processor to better exploit memory-level parallelism.
- **Fetch Speed (Performance):** This parameter determines how quickly the processor can fetch instructions from memory. A higher fetch speed can increase performance by allowing the processor to access instructions more quickly.
- **Cache Hierarchy**: The **cache hierarchy** of the processor can have a significant impact on both performance and energy dissipation. Generally, increasing the size of the caches can increase performance by reducing the number of memory accesses required. However, this can also increase energy dissipated due to the additional circuitry required to handle the larger caches.
    - iL1 **(Performance)**
    - dL1 **(Energy)**
    - dL1**(Energy)**
- **Cache Latency -> Cache Size:** Since we have a direct mapping between the il1/dl1/ul2 sizes and their respective latency. Changing the cache size directly affects the cache latency as well.

Experiment-1 Results:

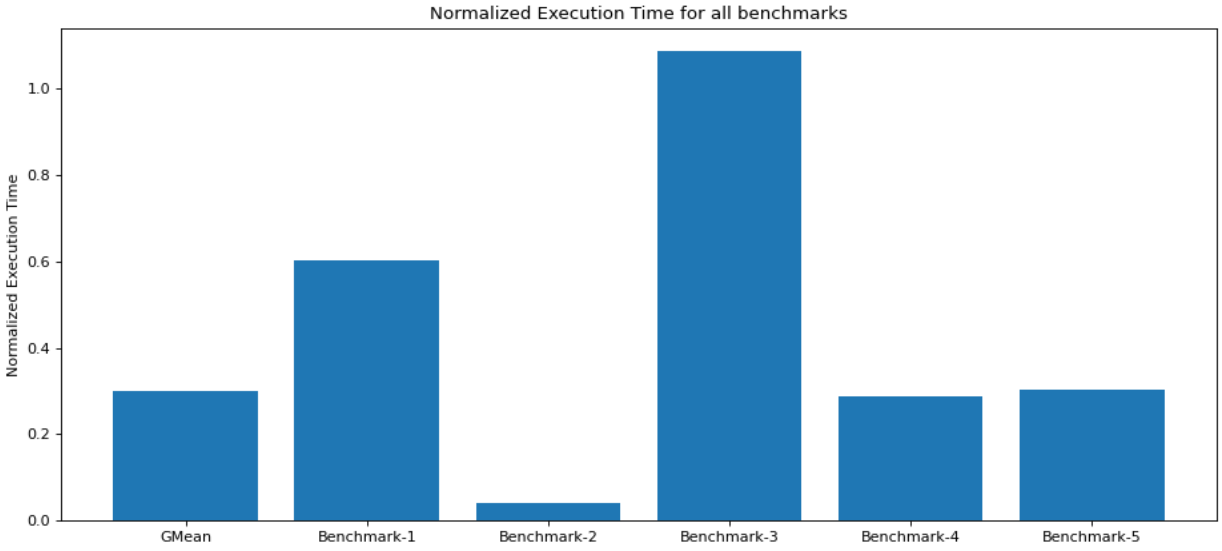
Plot 1:



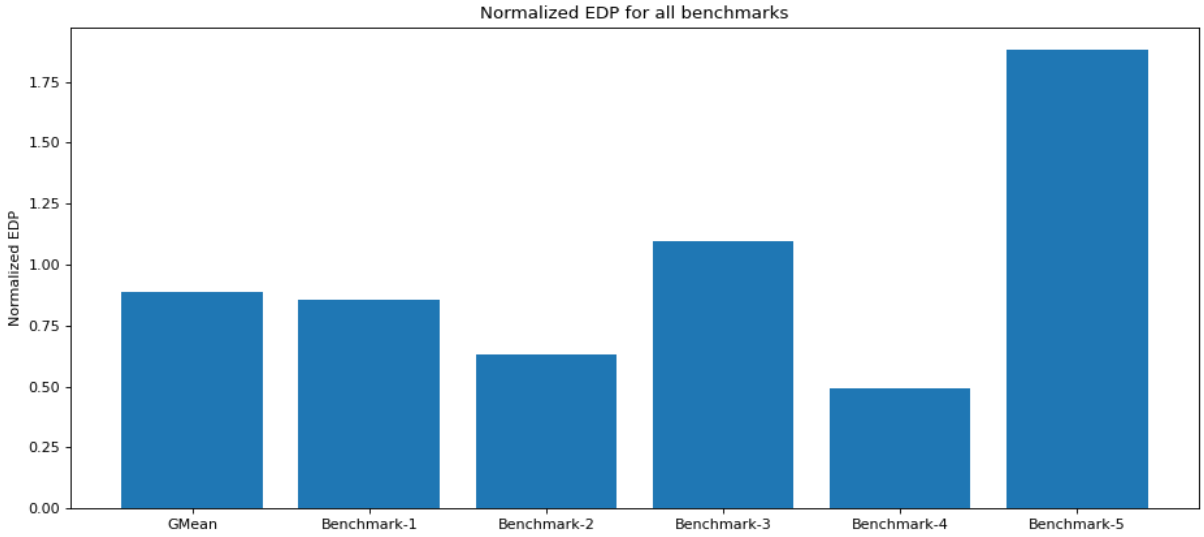
Plot 2:



Plot 3:



Plot 4:



Observations:

- We observe that the best configuration for both optimization functions were similar in terms of the following parameters/dimensions: ***Scheduling*** (out-of-order) and ***Branch Predictor*** (Combined)
- **Performance Optimization:**
  - normalized ED^2P had a range of [3.41, 0.30]
  - we see that while the search space was well suited for optimal performance
  - the performance for most benchmarks was much better than the baseline with an average improvement of ~70%,
- **Energy Optimization:**
  - normalized EDP had a range of [3.13, 0.89]
  - we observe that the normalized EDP (energy) improved to only 0.89,
  - Energy dissipation for most benchmarks wasn't impressive either. We only see a geometric mean improvement of ~20% from baseline.

We can therefore conclude that the ***search space needs to be expanded*** for energy optimization.

EXPERIMENT-2:

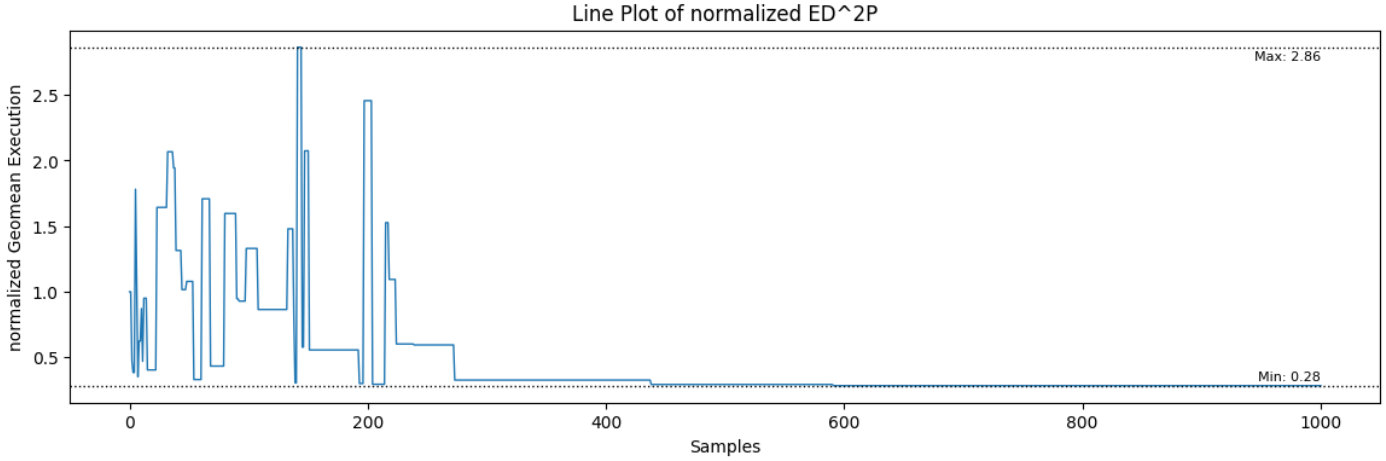
- Based on the observations from the previous experiment we need to improve the search space at least for energy optimization function
- **We therefore reconsider our parameter selection for Energy optimization:**
  - Added RUU Size (Energy) parameter: Increasing the register file size enables more data to be stored locally and therefore reducing the number of memory accesses required.
  - Added LSQ Size (Energy) parameter: Increasing the size of the load-store queue can increase performance by allowing the processor to better exploit memory-level parallelism.
- **We also try to improve results for Performance optimization:**
  - Added L2 Cache Hierarchy parameters into the search space to check if we can improve the results.
- **To keep reducing the search space, we fix the following parameters:**
  - Fixing scheduling (BOTH): Since the best configuration for both optimization function had *Out-of-order* scheduling I fixed the values for this parameter
  - Fixing Branch Prediction (BOTH): Since the best configuration for both optimization function had *Combined* prediction scheme, I fixed the values for this parament as well
  - Fixed Fetch Speed, RUU Size and LSQ Size (Performance)

Performance Optimization		
Index	Parameter	Combinations
0	width	4
1	Fetch speed	Fixed
2	Scheduling	Fixed
3	RUU Size	Fixed
4	LSQ Size	Fixed
8	iL1 Sets	9
9	iL1 Ways	3
10	L2 Sets	10
11	L2 Blocksize	4
12	L2 Ways	5
17	Branch Predictor	Fixed

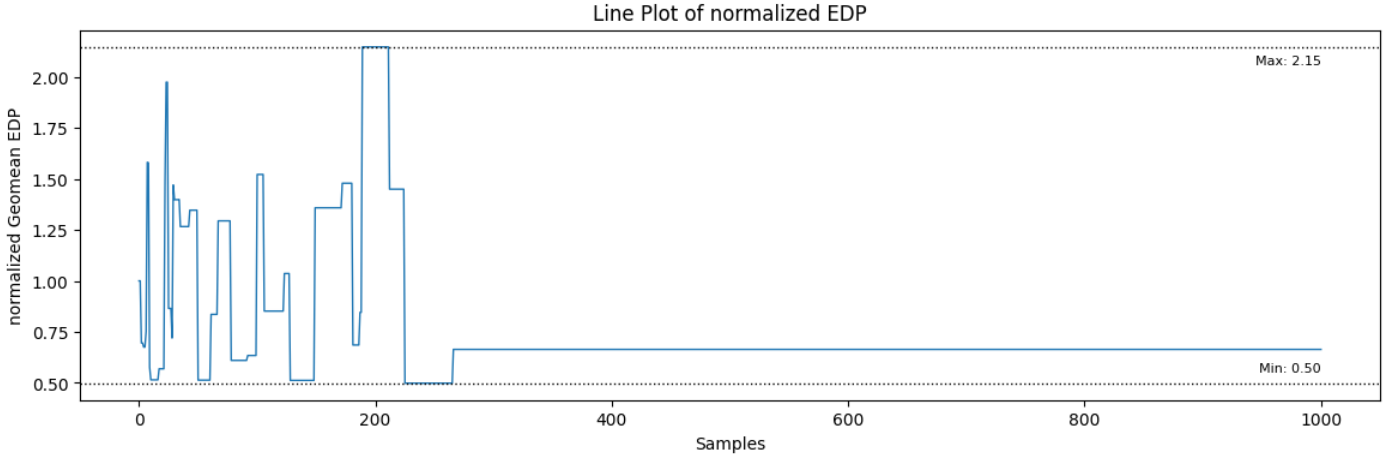
Energy Optimization		
Index	Parameter	Combinations
0	width	4
1	Fetch speed	2
2	Scheduling	Fixed
3	RUU Size	6
4	LSQ Size	4
6	dL1 Sets	9
7	dL1 Ways	3
10	L2 Sets	10
11	L2 Blocksize	4
12	L2 Ways	5
17	Branch Predictor	Fixed

Experiment-2 Results:

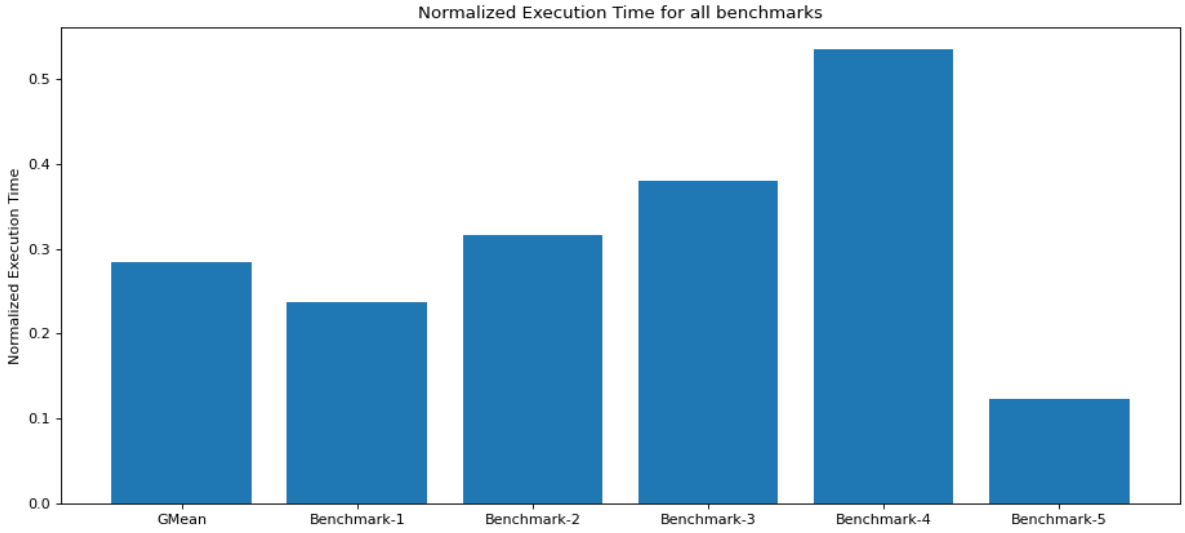
Plot-1:



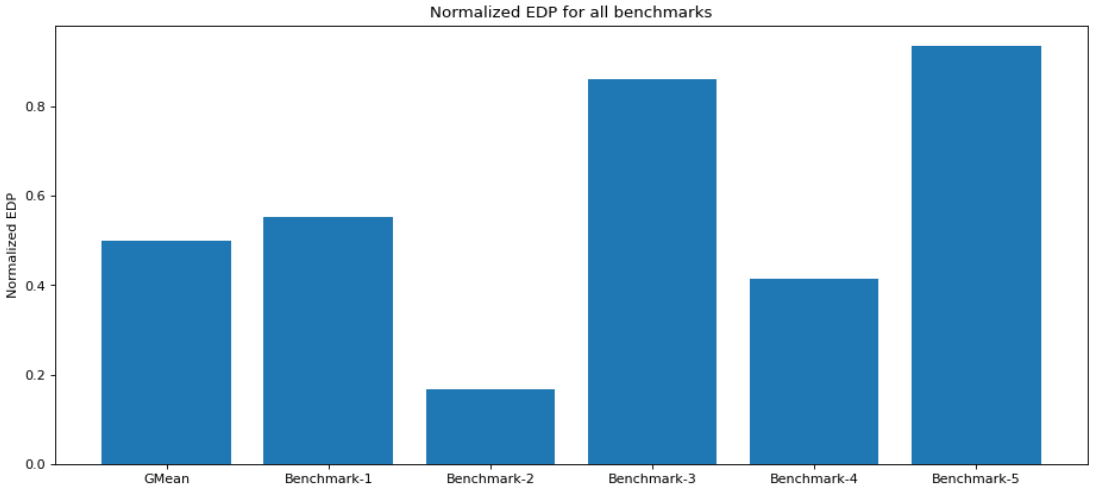
Plot-2:



Plot-3:



Plot-4:



Observations:

- **Energy Optimization:**
  - As expected, we see significantly better results, by adding RUU and LSQ size to the search space as both enable improvement in memory access/parallelism.
  - The improvement goes from 11% to 50% on the baseline EDP
  - We also see a significantly better GM of ~0.5 compared to ~0.8 in our previous experiment.
- **Performance Optimization**
  - We do not see a huge improvement in overall performance as evident by the min in the line graph and geometric average in the bar graph
  - We do however see an improvement in performance for branchmark-3

We can therefore conclude that adding RUU and LSQ size parameters to the search space dramatically improved energy efficiency. While we did not see any significant improvement in execution time on adding L2 Cache Hierarchy parameters to search space.

BEST CONFIGURATION FROM BOTH EXPERIMENTS:

		Best Config for Performance		Best Config for Energy	
Index	Parameter	Strategy - 1	Strategy - 2	Strategy - 1	Strategy - 2
0	Width	4	2	2	4
1	Fetch Speed	2	2	1	2
2	Scheduling	Out-of-order	Out-of-order	Out-of-order	Out-of-order
3	RUU size	128	128	4	128
4	LSQ size	32	32	4	32
5	Memport	1	1	1	1
6	DL1 Sets	1024	1024	64	1024
7	DL1 Ways	1	1	4	1
8	IL1 Sets	256	1024	1024	1024
9	IL1 Ways	2	1	1	1
10	UL2 Sets	1024	4096	512	1024
11	UL2 Block	64	32	128	128
12	UL2 Ways	4	1	1	2
13	TLB Sets	32	32	32	32
14	DL1 lat	4	2	3	3
15	IL1 lat	4	2	3	3
16	UL2 lat	13	10	9	12
17	Branch settings	Combined	Combined	Combined	Combined