Benchmarking various applications on ARM boards

Beeram Sandya CS21BTECH11006
Challa Akshay Santoshi CS21BTECH11012
Cheekatla Hema Sri CS21BTECH11013

Figure 1: Odroid-XU4

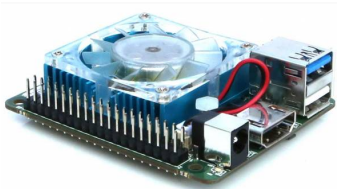

Figure 2: Odroid-N2L

1. Boards used:

- Odroid XU4 (32-bit processor)
- Odroid N2L (64-bit processor)

2. Benchmark Suites used:

Polybench https://github.com/MatthiasJReisinger/PolyBenchC-4.2.1
Polybench consists a total of 30 benchmarks.
Splash-4 https://github.com/OdnetninI/Splash-4

3. Odroid-XU4

Odroid XU4 is a new generation computing device with more powerful and energy efficient hardware. It also has a advantage of its size. Due to it's smaller size, it is convenient to carry and easy to use.
Odroid-XU4 supports 32 bit processor. It has eight core processor. The CPU is based on little-big architecture. L2 Cache is used.

- 4 ARM Cortex-A15 cores running at 2.0 Hz
- 4 ARM Cortex-A7 cores running at 1.4 Hz

The big cores(Cortex-A15) are generally used for high performance tasks, where as the little cores(Cortex-A7) are generally power-efficient and used for lighter tasks.

For detailed information : https://wiki.odroid.com/odroid-xu4/odroid-xu4

4. Odroid-N2L

Odroid-N2L is a single board computing device, which is small in size yet powerful performance. The main CPU is based on little-big architecture. It has a total of 6 cores. Two little cores and four big cores. L2 Cache is used.

- 4 ARM Cortex-A73 cores
- 2 ARM Cortex-A53 cores

For detailed information : https://wiki.odroid.com/odroid-n2l

5. Instructions for boards set up

1. Using Angry IP Scanner we found the IP address of the boards.
2. Using the IP address we login to the devices (boards) using ssh. "ssh odroid@IP-Address" command is used and the password for both the boards is "odroid".
3. Cross-compilation to get executable files for boards.
4. For polybench benchmark suites, execute the files XU4_gen_exec.sh, N2L_gen_exec.sh as instructed in the readme file in the given link: LINK
5. As for splash benchmark suites, to generate the executable binaries for board, navigate to the 'Makefile.config'
    - replace the word gcc in line 1 with aarch64-linux-gnu-gcc to generate executables for odroid N2L board
    - replace the word gcc in line 1 with arm-linux-gnueabihf-gcc to generate executables for odroid N2L board
6. After obtaining the executable file, they can be sent to the board via 'scp' command (to transfer files from PC to board).

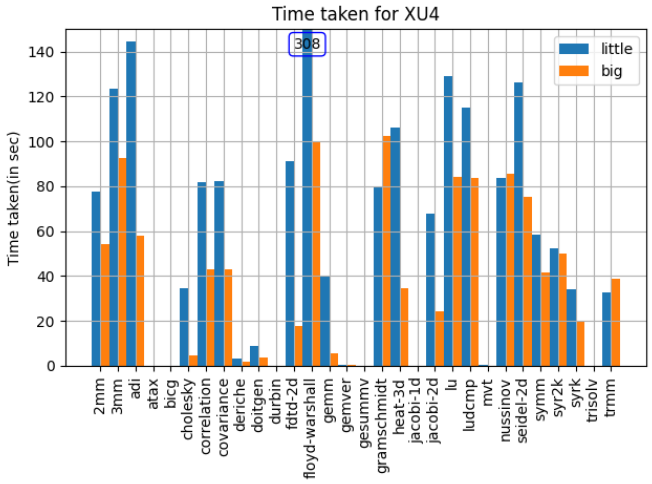

Figure 3: Time taken for execution on Odroid-XU4

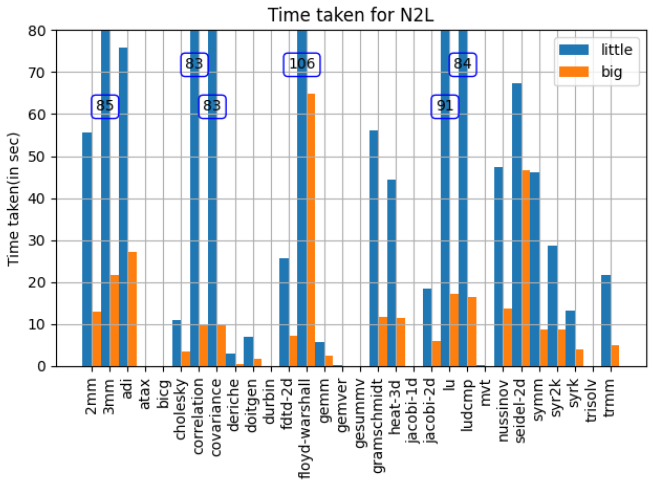

Figure 4: Time taken for execution on Odroid-N2L Time taken on the N2L board is less compared to the XU4 board.

7. The execution of binary files to obtain the data of time taken for execution can be automated via bash scripts. For example, the bash script to automate the execution to collect time for polybench benchmark suites is given in the following link: LINK (refer files XU4_execures.sh and N2L_execures.sh)
8. But for collecting data of power consumption by benchmark suites, it is preferable to run one at a time manually.
    - Power is measured using the device smartpower3. To extract power values measured by the device, use minicom (a terminal emulation program)
9. To run program on each core parallely, use '&'

6. Polybench

Polybench has 30 applications used for various mathematical purpose.

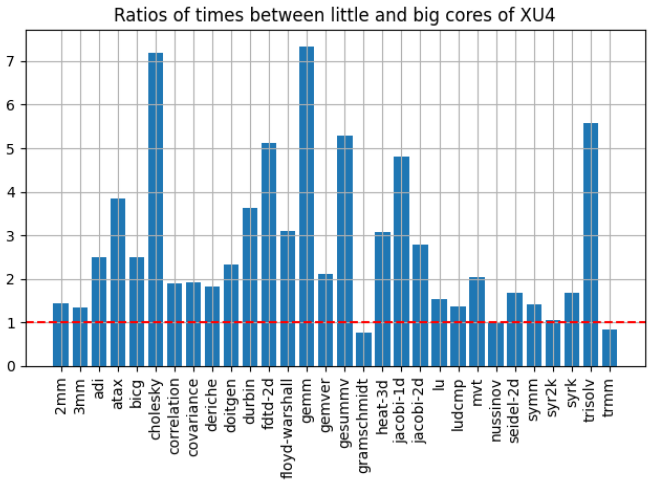

Figure 5: Ratio of times taken on little core vs big core.As expected, the time taken for execution on little core is more than the big core for most of the applications. The plot helps us to identify whether a task prefers big core or small core in terms of raw processing power (time taken). With the exception of gramschmidt and trmm, we can observe from the plot that little cores are taking more time than the big cores. The exceptions may arise due to lightweight computations present in them that benefit from power efficiency of the little cores or not being able to take full advantage of the capability present in big cores. Most of the tasks take less time on big cores because they have higher clock speeds for greater processing power, larger cache sizes which helps in improved data access and other complex micro architecture like pipeline depth etc.

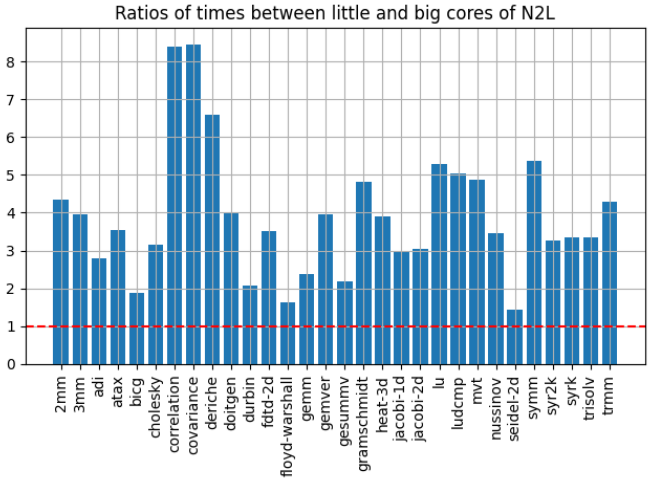

Figure 6: Ratio of times taken on little core vs big core.As expected, the time taken for execution on little core is more than the big core for most of the applications. Even exceptions which occurred in odroid XU4 (gramschmidt and trmm) which took more time in big cores than little cores are here showing the general trend, i.e big cores are outperforming little cores in all. A73 cores in N2L are more powerful than A15 cores in Xu4. N2L has larger L1 cache size per core compared to XU4. Gramschmidt and trmm have characteristics that favour the architecture of little cores in Odroid XU4 and big cores in Odroid N2L in terms of time.

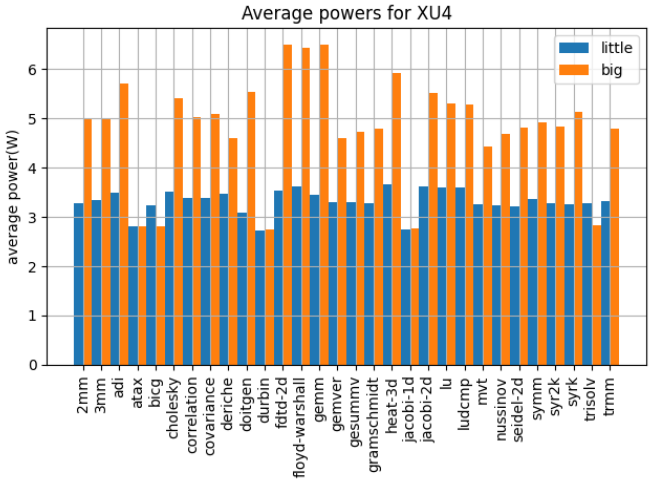

Figure 7: Power used by the little core for all the benchmarks on XU4 is around 3.32 W. Power used by the big core for all the benchmarks on XU4 is around 4.82 W.So the ratio of the power used by the big and the little core is 1.45.

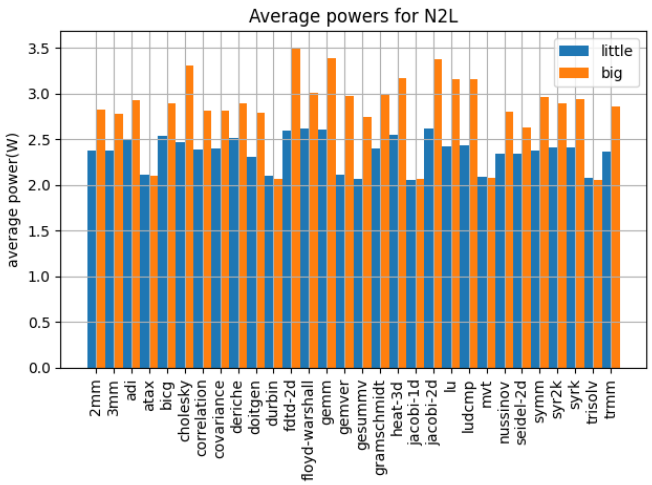

Figure 8: Power used by the little core for all the benchmarks on N2L is around 2.37W. Power used by the big core for all the benchmarks on N2L is around 2.83W. So the ratio of power used by the big and the little core is 1.19.

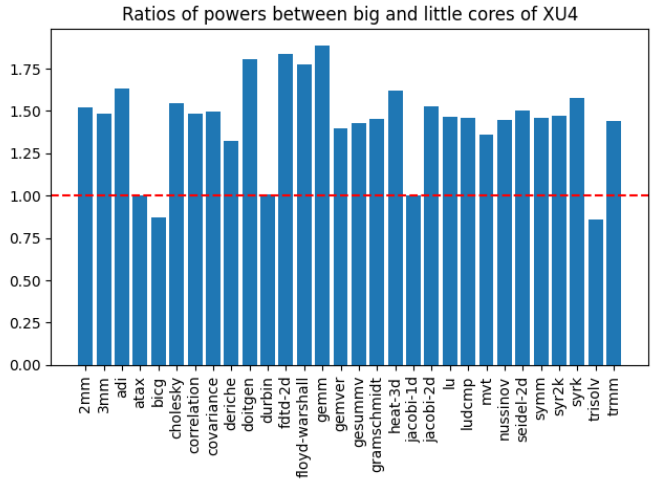

Figure 9: Ratio of power used on big vs little core. Power used by the big core is more compared to the little core. The plot helps us to identify which core is preferable for each benchmark suite in terms of power efficiency. With the exception of bicg and trisolv, we can observe that little cores consume less power than big cores. This is because little cores have simpler microarchitecture with smaller pipelines which helps in reducing power consumption. The exceptions may arise due to the benchmarks having computations that prefer big cores than little cores. For example, if the benchmarks exhibit specific memory access patterns that align better with the cache hierarchy or memory subsystem of bigcores or benefit significantly from higher clock speeds, then they may prefer big cores even if it results in higher power consumption.

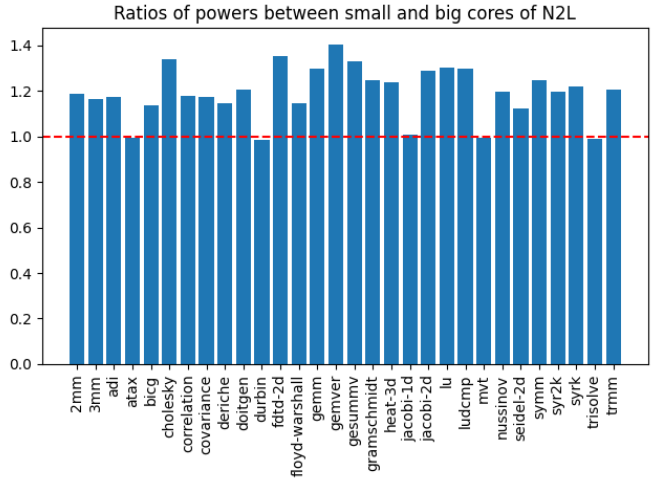

Figure 10: Ratio of power used on big vs little core. Power used by the big core is more compared to the little core. We can observe from the plot that power consumed is less in little cores than big cores. Even for benchmark suites for which big cores are showing less power consumption, the ratio is very close to 1. So we can generalize that little cores are outperforming big cores in terms of power consumption in N2L.

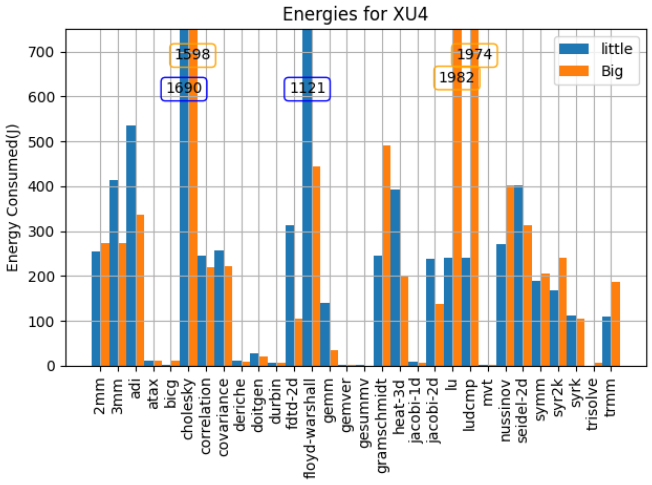

Figure 11: Energy usage by the applications on Odroid-XU4

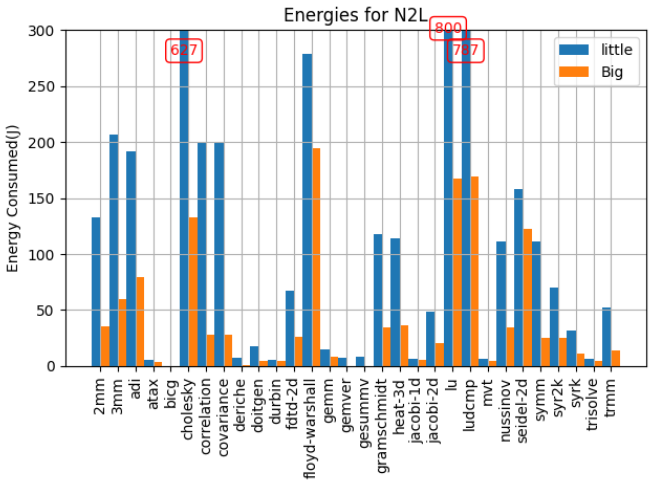

Figure 12: Energy usage by the applications on Odroid-N2L. Energy used on the N2L is less than the XU4.

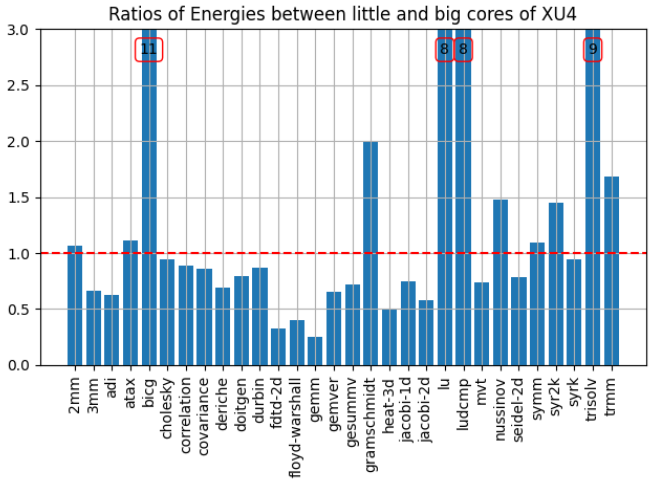

Figure 13: Ratio of energies on little vs big core

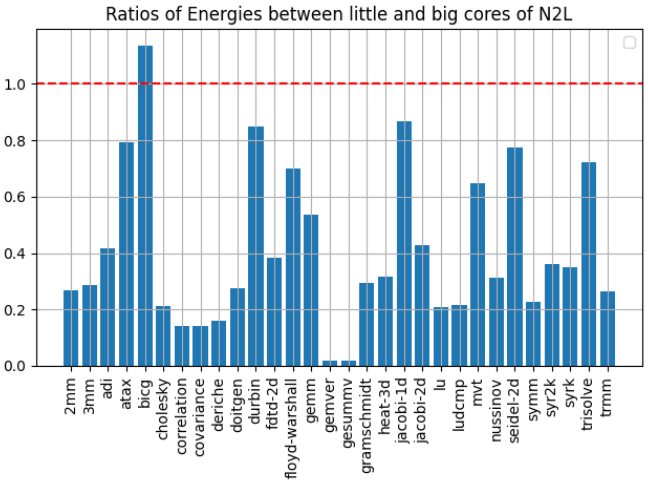

Figure 14: Ratio of energies on little vs big core

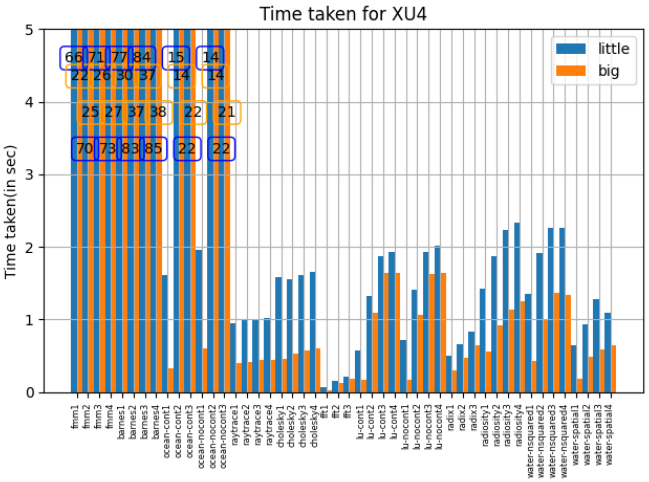

Figure 15: Time taken on the XU4 for each application. Time increases as the number of processes increase.

7. Splash-4

Splash-4 has 15 applications. We have worked on 13 applications leaving the volrand and related applications. We can change the size of the inputs and also the number of processes for most applications.

8. Difficulties faced

- Finding the reason for anomalous behaviour of some applications
- Understanding the code of the applications
- Extracting power was difficult, as we have to run each application manually and wait for its execution to be completed. So that we can stop extracting the power from the device.

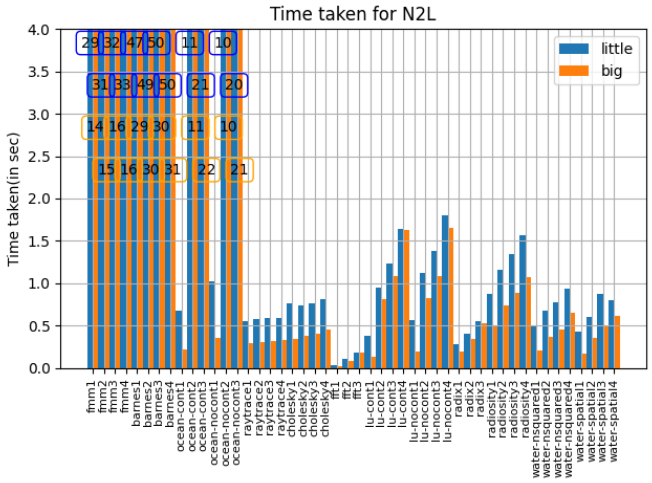

Figure 16: Time taken on the XU4 for each application. Time increases as the number of processes increase.

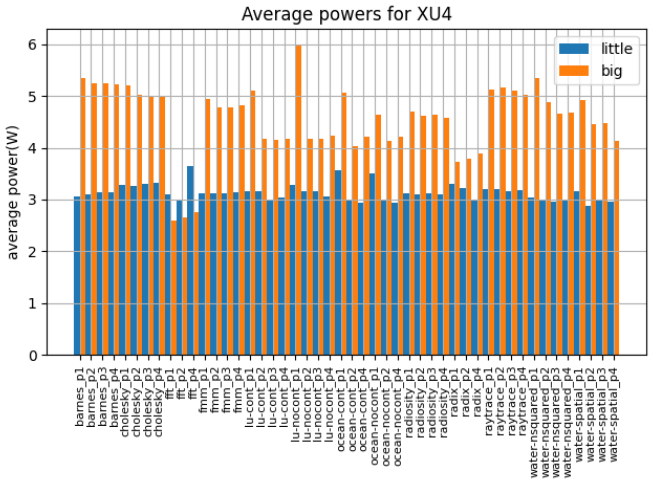

Figure 17:

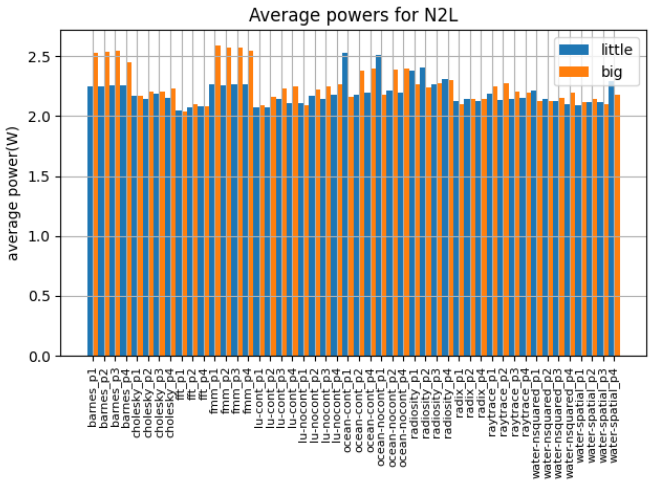

Figure 18:

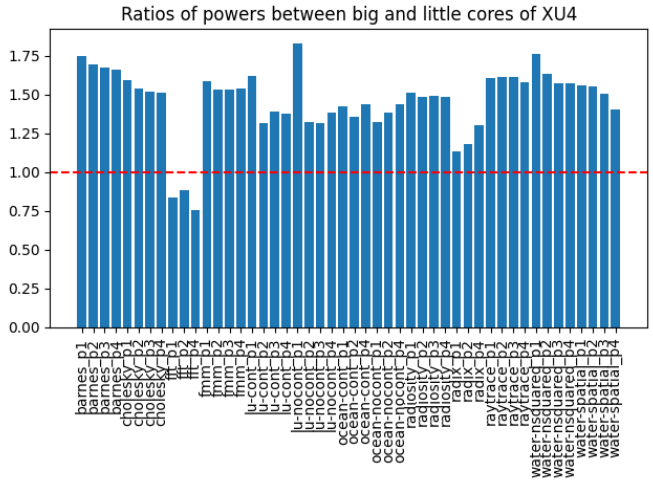

Figure 19:

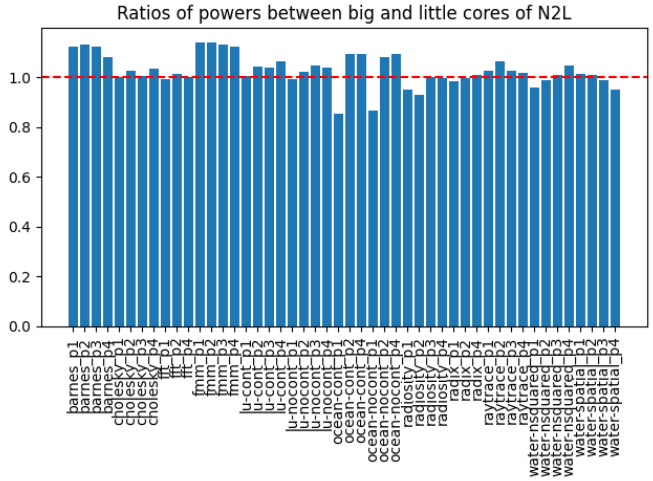

Figure 20:

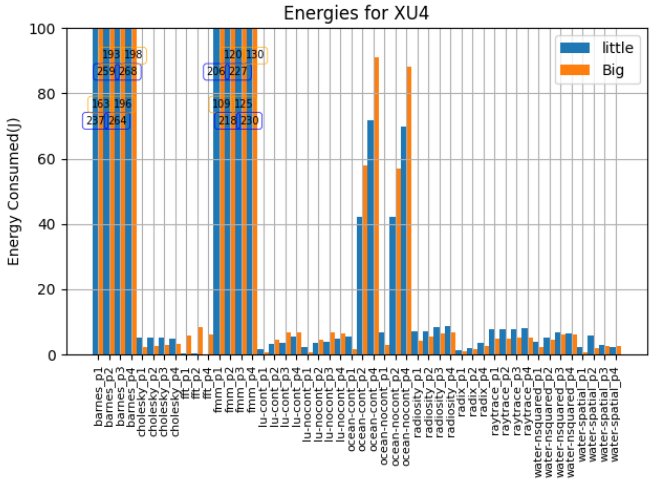

Figure 21:

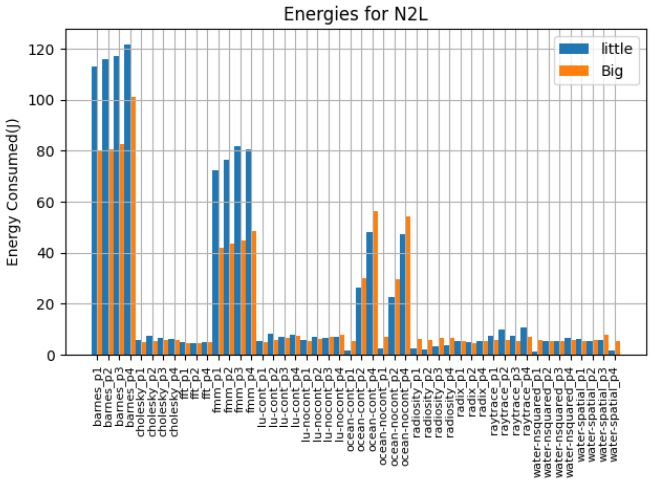

Figure 22:

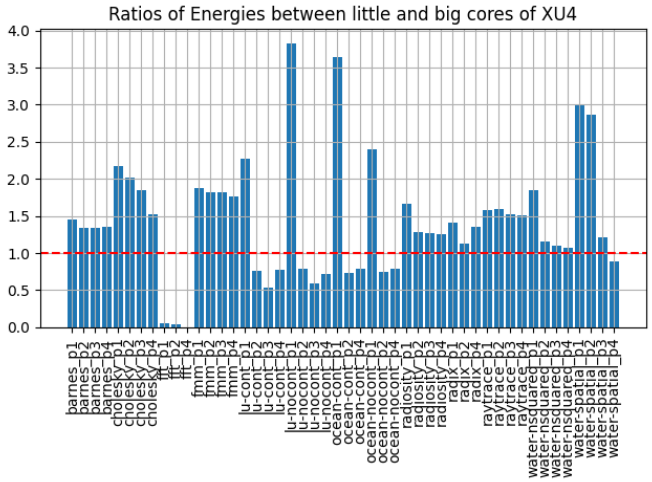

Figure 23:

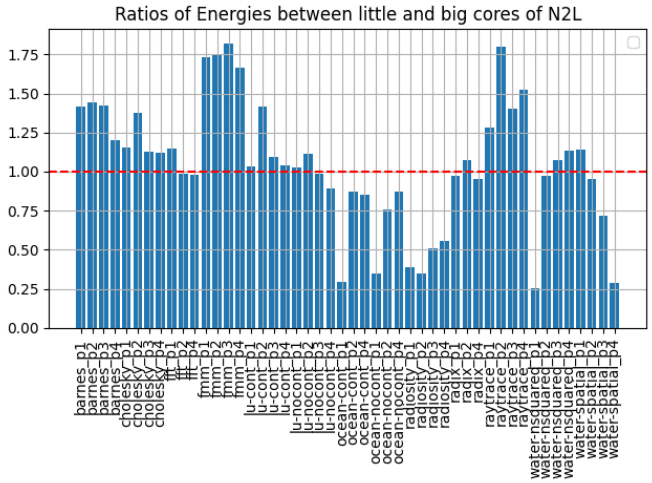

Figure 24:

9. Conclusion

- For both Polybench and Splash-4 benchmark applications , N2L is more energy efficient than XU4
- Power used by the N2L boards is less compared to the XU4 board, although we expect other way round.
- As the number of processes are increasing for Splash-4 applications, the performance of the big cores becomes better.
- Cores numbered 0,1,2,3 are little and 4,5,6,7 are big in Odroid-XU4.
- Cores numbered 0,1 are little and 2,3,4,5 are big in Odroid-N2L.

10. Feedback

- It would have been better if we had understand the boards architecture more.
- Working as a group was helpful.(Mainly for executing and analysing)
- Splash-4 has various input sizes and processes. Therefore, we have to consider various aspects while analyzing the data.