A Methodology for Evaluating the Energy Efficiency of Post-Moore Architectures

Pablo Josue Rojas Yepes, Carlos Jaime Barrios Hernandez, Luiz Angelo Steffenel

A Methodology for Evaluating the Energy Efficiency of Post-Moore Architectures

Pablo Josue Rojas Yepes[1][0000−0003−1823−7922], Carlos Jaime Barrios Hernandez[1][0000−0002−3227−8651] and Luiz Angelo Steffenel[2][0000-0003-3670-4088].

[1]Universidad Industrial de Santander, Cl. 9 # Cra 27, Bucaramanga, Colombia
[2]Université de Reims Champagne Ardenne, LICIIS – LRC CEA DIGIT Laboratory, Reims, France

pablo2198162@correo.uis.edu.co
cbarrios@uis.edu.co
angelo.steffenel@univ-reims.fr

Abstract. The improvement of computational systems has been based on Moore's law and Dennard's scale, but for more than a decade it has started to fall to a standstill. To maintain the improvement new technologies are proposed, this has established a Post-Moore era. Currently there are different methodologies to evaluate emerging devices and technologies, but the results of these methodologies end up being particular solutions. This paper presents a methodology that can evaluate Post-Moore devices or technologies in an agile way, characterizing an application, choosing a device that meets its needs, selecting tests and parameters that are executed to meet the objectives set by the methodology. The results show how to evaluate the energy efficiency of these new technologies, but the scope of the methodology can cover other needs.

Keywords: Evaluation Methodology, Embedded System, Benchmark, Low-cost Computing, Edge Computing, Many-core and Heterogeneous Computing.

Introduction

Measuring the performance of computer systems was an art in its beginnings, with the passage of time certain agreements were reached. Among the approaches that were adopted to measure the performance of a computational architecture are the time it took to complete a specific task or the number of operations performed, one of the main metrics that emerged due to this was FLOPS. One of the main tools to perform these measurements was HPL [1]. Due to its constant use it became the basis of measurement of the main HPC machines [2] until it became a standard for HPC hardware.

At the same time, Moore's law was established as the main driver of computational growth, but its progress started to stagnate in the last decade [3, 4]. To overcome this barrier, different approaches were presented to address the problem, which triggered a "Cambrian explosion" [5] of hardware and dubbed the current era as the Post-Moore

era [6]. One of the most dominant approaches states that applications determine the right hardware to run them [6]. Based on this proposition, the premise is formulated: Applications Choose Devices Carefully.

This paper assumes the premise just proposed and presents a methodology to evaluate the efficiency of selected Post-Moore architectures. **This methodology is called "Applications Choose Devices Carefully" or ACDC for short**, It should be clear that the methodology is aimed at single and low-cost hardware.

The first section presents some of the state of the art of evaluation methodologies, the second section shows the functioning scheme of the ACDC methodology, the third section describes each of the steps of the scheme, the fourth section presents a use case of the ACDC methodology and finally the conclusions of the work are presented.

1 State of the Art

After reviewing the extensive state of the art, three contributions stand out in the last three years in reference to the evaluation of Post-Moore architectures.

The first contribution [7] studies the applicability of FPGAs to accelerate the core of the Spectral Element Method (SEM) in many computational fluid dynamics (CFD) applications. This is evaluated using Stratix 10 GX series FPGAs versus systems such as ARM ThunderX2, NVIDIA Tesla/Pascal/Volta/Ampere series boards and general purpose many-core CPUs. The methodology consists of configuring the SEM on the FPGA and using languages such as OpenCL [8, 9] or OpenACC [10] to configure the SEM on the GPUs and CPUs, the development of the test methodology is implicit in the development of the work.

The second contribution [11] proposes a comprehensive open source methodology for evaluating emerging technologies. To demonstrate its effectiveness, the methodology is used to perform end-to-end simulation and analysis of heterogeneous architectures using CNFETs, TFETs and NCFETs, along with multiple hardware designs. The methodology consists of four main levels of modeling: device models, logic gates, logic and memory blocks, and architecture. This methodology evaluates CNFET, TFET and NCFET devices using hardware designs of varying complexity and provides the means to assess the high-level impact of emerging technologies.

Finally, the third contribution [12] presents a testbed [13, 14, 15, 16] covering networking, identity management, resource scheduling and tools along with techniques they have developed to manage these Post-Moore devices. More than a methodology, it presents several methods to evaluate different Post-Moore architectures with different tests. This generates particular evaluation methodologies for different devices.

With this brief presentation, the ACDC methodology will be outlined.

2 ACDC methodology outline

As discussed above, one of the main focusses of the Post-Moore era is that the applications determine the hardware on which they will be deployed. Furthermore, with the

"Cambrian explosion" [5], it is possible to adopt a methodology to evaluate the applications and the possible hardware that can execute them.

Another approach of the Post-Moore era is that the hardware integrates multiple specific purpose chips, tailored to the needs of the application [6]. It should be noted that the hardware to be used for the development of the work is based on traditional manufacturing. Hardware for neuromorphic computing, quantum computing [17] or 3D-Chips will not be taken into account for this methodology. Also note that the methodology will not use FLOPS as a metric, even though it is the most reliable metric to measure performance, the methodology intends to evaluate different types of operations, not only floating operations. The evaluation will take into account the number of operations completed in a defined time, their energy consumption, among others.

Fig. 1. is a outline of how the implementation of the ACDC methodology would be carried out. It consists of nine sequential steps but the penultimate step presents a loop that may point to step three or four depending on the situation.

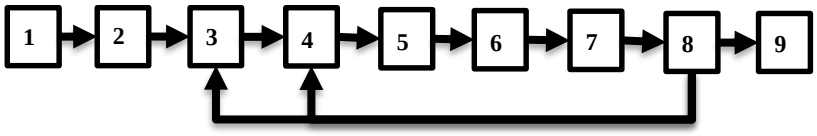

Fig. 1. ACDC Methodology Outline

The steps of the ACDC methodology are the following:

1. Selection of the application and definition of its primary needs.
2. Post-Moore device selection.
3. Test settings optimization.
4. Parameterization of the test.
5. Preparation of miscellaneous needs.
6. Running the test and data capture.
7. Data labeling and storage.
8. Repeat the test, select a new test or end the test.
9. Analysis of results.

It is necessary to understand the steps to follow in the application of the ACDC methodology. The following section presents this topic.

3 Description of the ACDC Methodology

3.1 Selection of the Application and Definition of its Primary Needs

As it is the first step, it will be the main guide for the development of the evaluation. Therefore, the backbone of the methodology is that the applications choose the devices that can be deployed carefully or the **Applications Choose the Devices Carefully (ACDC)**. In order to implement this methodology, all hardware that brings together chips of a specific use and that is manufactured in a traditional way or with traditional

technologies such as Post-Moore hardware is considered. When selecting an application (the application can be a methodology, a benchmark, a code, etc), a complete characterization of the selection must be made. This characterization defines the main needs for the implementation, which in most cases are as follows:

- Multiplatform or cross-compiled.
- Define the hardware the application needs (CPU, GPU, NPU, FPGA, etc).
- Identification of dependencies.
- Types of data that the application works.
- Classification of instructions (RISC or CISC).

3.2 Post-Moore Device Selection

In the previous section the application has been chosen, characterized and its main needs have been defined. To meet these needs, it is necessary that the devices contain the necessary hardware for the implementation of the application. By characterizing the application, its behavior is sketched and the hardware needs narrow down the search for devices, simplifying the selection process. Using the right hardware saves time in the implementation, improves the performance and efficiency of the application.

This step can reverse its order with the first one, sometimes there is a certain device. For these cases, the device must be characterized to know its composition. The selected application must have its main needs oriented to the available hardware.

3.3 Test Settings Optimization

With the application and the device chosen, two possibilities arise to carry out the evaluation. The first is to perform **synthetic tests**, the second is to **run the application**. With the application study, profiling can be carried out to see what the main bottlenecks are.

If choose a set of synthetic tests that simulate bottlenecks. The metrics for these tests are primarily the number of iterations of the test over a specific time or the amount of time it takes to perform a certain number of operations. The results of the tests are taken as metrics to present the conclusions of the evaluation.

When run the application, it is possible to use general metrics to measure the behavior of the device. In this case, metrics such as execution time acquire greater relevance and will be the focus of the results and conclusions.

The test configuration can be done in various ways, multiple tests can be performed with different resource allocations, the configuration that generates the highest performance can be chosen, or depending on what the evaluation is looking for, parameters are defined in the configuration. In this step, data captures are made to measure behavior but are not taken into account in the final results. Preliminary tests should be carried out to get an idea of the behavior of the test to be performed on the chosen devices.

3.4 Parameterization of the Test

The previous step generates a list of parameters that fit the needs of the evaluation and influence the test results. It is up to whoever performs the evaluation to choose which results are important for the test on the device.

3.5 Preparation of Miscellaneous Needs

With the parameters established, if the evaluation objectives require it, additional measurement elements should be established for the tests. Items such as consumption monitors, resource monitors or flags are assigned to the tests to facilitate their execution and data capture.

3.6 Running the Test and Data Capture

With the device, test and monitors synchronized, the test is run. It is advisable to carry out preliminary tests to verify the operation together. It is not advisable to carry out tests after turning on the device since the OS may be executing processes in the background. Monitors must be tracking the device before the test runs, this way a base behavior of the device can be set when not in use.

Before the test starts, data capture must begin. In this way, evidence of the base behavior and during the test is left. The data must be taken according to the metrics that are set for the evaluation, this work is humdrum and can generate headaches. The best option is to formulate and follow a roadmap, this simplifies and mechanizes the work.

3.7 Data Labeling and Storage

The captured data must be preserved for later analysis. It is necessary to generate a list of labels, to segment the data and facilitate its analysis. This process is important for generating conclusions and meeting objectives.

In the labeling process, ideas can arise that add more value to the research results, it is recommended to be open to new ideas that reveal new approaches. It is recommended that the data be stored in the cloud or on a medium that has a backup, the loss may delay the investigation or canceled. The stored data should use the suggested labels to facilitate its handling, using tables, matrixes or dictionaries facilitates their handling.

3.8 Repeat the Test, Select a New Test or End of the Test

This step is a trifurcation in which one must choose between the following paths:

- **Repeat the Test:** happens when there is insufficient amount of data to identify a pattern of behavior. Tests can be repeated as many times as deemed necessary, the number of repetitions may depend on factors such as standard variation or error reduction, but it is up to the tester to define this.

- **Select a New Test:** happens when it is considered that it is not necessary to repeat a test. Proceed to step three and start the process up to step eight, the test is repeated as above. In this way it is possible to mechanize the process of the methodology and speed up the acquisition of data in the tests.
- **End of the Tests:** at the moment when it is considered that there is an acceptable amount of data and all the tests were performed. The data are analyzed, thus generating conclusions that will cover the objectives set for the evaluation.

3.9 Analysis of Results

This is the last step, all the data collected must be processed to generate the results that will evaluate the methodology, the results must be clear for the generation of information, comparative tables, etc. They will be used in the conclusions and the fulfillment of the objectives that are proposed to be covered with this methodology .

The next section proposes the implementation of the ACDC methodology.

4 Execution of the ACDC Methodology

In order to see in a practical way how the ACDC methodology would be implemented, a hypothetical application is selected, the result is presented below:

4.1 Step 1. Selection of the Application and Definition of its Primary Needs

To simplify the selection, a hypothetical application with the following characteristics is proposed: the application is compiled for x86 and ARM architectures, it is written to use the maximum amount of available resources, it uses CPU parallelization, its installation can be done through package managers and the type of data it handles is floating.

The workload of the application depends on the resources available to the machine, but the main bottlenecks are:

- The application generates multiple pthreads (POSIX threads) at runtime.
- During application execution allocate, reallocate and free memory with malloc, calloc, realloc and free calls dynamically.
- The computation operations it performs are floating number ones.

To evaluate the methodology, the following objectives are established:

- Measure the performance of the selected devices.
- Measure the energy efficiency of the devices.

4.2 Step 2. Post-Moore Device Selection

The application uses only the CPU and is compiled for ARM and x86-64, this opens the possibility to use any device at hand. In this case, an Nvidia Jetson Nano [18] and

a desktop PC are selected. The desktop PC's dimensions (H x W x D) are 497 x 250 x 511 mm, while for the Jetson Nano are ~30 x 100 x 80 mm. These devices are considered Post-Moore because they have many cores and heterogeneous. Both have an Nvidia GPU but for the use case it is not necessary and is not taken into account. With the chosen devices the optimization of the test setup can be performed.

The **Device** column presents the name to be given to each device, the Jetson Nano is labeled Nano and the desktop PC is named PC. The **CPU** column describes the CPU of each device, the **GPU** column explains the GPU of each device, the GPU of the PC is a GTX 1050ti with 4GB of independent memory, while the Nano shares its memory between the CPU and GPU. The **RAM** column details the memory in both devices.

Table 1. Devices selected

Device	CPU	GPU	RAM
Nano	ARM Cortex-A57 Quad-Core @~1.5GHz	Nvidia Maxwell 128 CUDA-Cores ~921 MHz	4 GB LPDDR4 @1600MHz
PC	AMD Ryzen 5 3600 Hexa-Core @3.6~4.2GHz	Nvidia Pascal 768 CUDA-Cores 1350~1800 Mhz	16 GB DDR4 @3200MHz

Both devices meet the needs of the hypothetical application. It should be emphasized that the tests will only focus on the CPU due to what was stated in Step 1.

4.3 Step 3. Test Settings Optimization

For this case, the option of synthetic tests is taken, so that focused tests can be performed. The three bottlenecks are: Generation of multiple Pthreads, memory manipulation and operations with floating numbers. To deal with them, the following synthetic tests are proposed [19]:

- **pthread:** start N workers that iteratively creates and terminates multiple pthreads (the default is 1024 pthreads per worker). In each iteration, each newly created pthread waits until the worker has created all the pthreads and then they all terminate together.
- **cfloat:** 1000 iterations of a mix of floating point complex operations
- **malloc**: start N workers continuously calling malloc, calloc, realloc and free. By default, up to 65536 allocations can be active at any point. Allocation, reallocation and freeing are chosen at random; 50% of the time memory is allocation (via malloc, calloc or realloc) and 50% of the time allocations are free'd. Allocation sizes are default size being 64K. The worker is restarted if it is killed by the out of mememory (OOM) killer.

The way to measure the results in these tests are as follows Operations or Ops for simplicity. Another facility provided by these tests is that they can be run for a set amount of time. Preliminary tests show that the Ops increase with a greater amount of time and

with a greater availability of cores, but the amount of Ops per second or Ops/s remains independent of the allotted time. As a result of this behavior, Ops/s is defined as a metric and the number of cores is taken into account as a parameter to measure the scalability of the application. The pretests also show that Pthreads and Malloc consume all the resources available to the device. The Cfloat test allows the variation of the number of cores. One of the objectives of the evaluation, to measure the performance of the chosen devices, is fulfilled.

4.4 Step 4. Parameterization of the Test

The Ops/s are maintained in spite of the time variation, for this reason it is not relevant to mention the time allotted.

The pthread test is executed with 4 jobs and 1024 threads per job, in the preliminary tests, a higher number negatively impacts the test results on the Nano due to its scarcity of resources.

The malloc test runs 4 jobs with the default parameters, 64K in size and 65536 allocations, as in pthreads there is a negative impact on the Nano due to the lack of resources.

The cfloat test is run with 1, 2, 3 and 4 jobs, unlike the previous two, this test only takes the amount of CPU allocated and not the total resources available. In this way is possible to see how the application scales when computational resources are allocated on an individual device.

4.5 Step 5. Preparation of Miscellaneous Needs

The second objective of the evaluation and the main goal of the methodology is determining the energy efficiency of the devices. To meet this objective, energy efficiency is defined as the amount of energy consumed by the operations performed during a given time. Ops/s provides the number of operations in a given time. The energy consumed is determined by a consumption monitor [20], this monitor captures the data during the time the test is performed.

During the preliminary tests a consumption before and after the test is evidenced, this consumption is taken as the base consumption. The total consumption during the test is subtracted from the base consumption. The energy consumed during the test is averaged, thus obtaining the energy consumed during the test or W/s. The energy efficiency for the evaluation is determined by the number of operations over the consumption or Ops/W.

$$\frac{Ops/s}{W/s} = Ops/W \tag{1}$$

The consumption monitor provides the consumption data and the tests present a summary with the relevant data at the end of each test. This is how the two evaluation metrics are defined, Ops/s and Ops/W.

4.6 Step 6. Running the Test and Data Capture

To run the tests, the consumption monitoring should be started before the tests, it is recommended that the device has been on for some time to avoid consumption peaks. The test is executed with the parameters assigned in step 4, once the test is completed the data obtained must be stored.

4.7 Step 7. Data Labeling and Storage

The data collected during step 6 are organized as shown in **Table 2**.

Table 2. Data Labeling and Storage

Device	Test	Core	Ops	Ops/s	W/s	Ops/W
Nano	Pthread	4	1159500	19325	5,8	349286
Nano	Cfloat	1	268145	4469	1,1	235215
Nano	Cfloat	2	535617	8927	2,1	252650
Nano	Cfloat	3	796733	13279	3,1	257011
Nano	Cfloat	4	793350	13223	3,8	206602
Nano	Malloc	4	241772479	4029541	5,2	71109553
PC	Pthread	4	6987682	116461	41,2	170089
PC	Cfloat	1	296587	4943	14,8	19995
PC	Cfloat	2	590779	9846	19,8	29787
PC	Cfloat	3	884458	14741	24,9	35664
PC	Cfloat	4	1171197	19520	29,0	40340
PC	Malloc	4	1595115864	26585264	39,2	40739188

Table 2 shows the following information: The **Device** column presents two labels, Nano for the Jetson Nano device and PC for the desktop PC. The **Test** column shows the different tests performed on the devices. The **Core** column lists the number of cores used in the test. The **OPS** column stores the number of operations performed during the test execution. The **OPS/s** column represents the number of operations performed per second. The **W/s** column shows the average power consumption per second during the test run, obtained from the difference between the power consumption during the test and the base power consumption of the card. Finally, the **OPS/W** column is the relation between OPS/s and W/s, this result does not show the energy efficiency of each device. The data obtained is stored in a spreadsheet. It should be clarified that the values shown by Table 2. are averages to save space.

4.8 Step 8. Repeat the Test, Select a New Test or End of the Test

Each test was performed three times in a preliminary way, in order to evaluate how the different parameters affected the tests. The cfloat test is the only one to which resources can be allocated, after three iterations a pattern of behavior is seen and it is run a total of five times per core to have a good database. The pthreads and malloc tests use all available resources, run fifteen iterations per test to have a database similar to cfloat.

Once the assigned number of iterations is completed, a new test is conducted and configured with the parameters set by the preliminary tests. When all the tests are completed, the data is preprocessed.

Table 2 shows the averaged results of all tests performed with its respective labels. The standard deviation in the pthread test is 0.752% for the Nano and 0.253% for the PC. For the malloc test it is 0.246% for the Nano and 0.202% for the PC. Finally, the standard deviation in the test cfloat has four results for each device, for the Nano it is 0.032% with one core, 0.015% for two cores, 0.020% for three cores and 0.241% for four cores. For the PC case they are 0.339% with one core, 0.140% for two cores, 0.006% for three cores and 0.124% for four cores. A total of approximately 100 preliminary and final tests were realized.

4.9 Step 9. Analysis of Results

When analyzing the data, the objectives that were set for the development of the methodology, in this case the following objectives should be remembered: measure the performance of the selected devices and measure the energy efficiency of the devices.

With the data collected in **Table 2**, graphs are generated to give a better understanding of the conclusions of the work. The graphs are presented pointing to two topics: the performance measurement of the devices is carried out by a comparison between the Ops/s of the Nano and the PC in the different tests, and the measurement of energy efficiency is shown by means of a graph that compares the Ops/W of the two devices. **Fig. 2** show the performance of the devices in the Pthread.

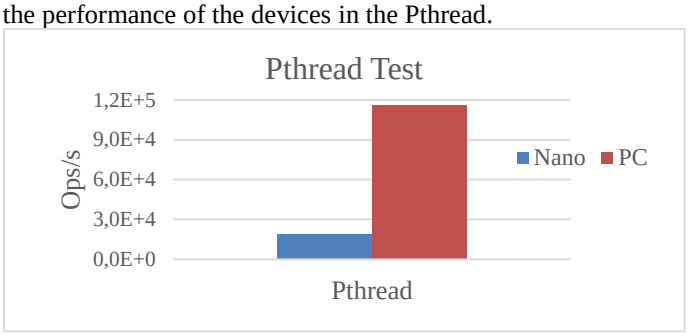

Fig. 2. Devices Performance in the Pthread Test

The results in **Fig. 2** show a performance of about six times higher for the PC. It should be mentioned that the ARM CPU has 4 cores, maximum frequency of 1.5 Ghz, 16 nm lithography and manages to execute 4838 Ops/s while the x86 CPU has 6 cores, maximum frequency of 4.2 Ghz, 7 nm lithography and manages to execute 19410 Ops/s. It can be assumed that an ARM CPU with the same characteristics as an x86 would have similar performance. **Fig. 3** displays the performance in malloc test.

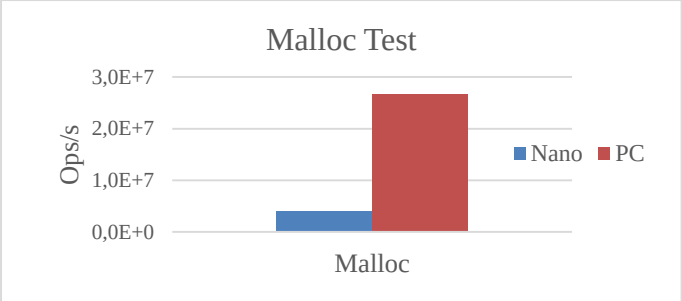

Fig. 3. Devices Performance in the Malloc Test

The malloc test (**Fig. 3**) exercises the memory of the devices, the PC achieves almost 6.6 times the performance of the Nano. As in the previous case, the memory resources on the Nano are much lower, causing this result. It is emphasized that if the Nano had similar capabilities to the PC the results would be very similar. **Fig. 4** illustrates the performance of the devices in the cfloat test.

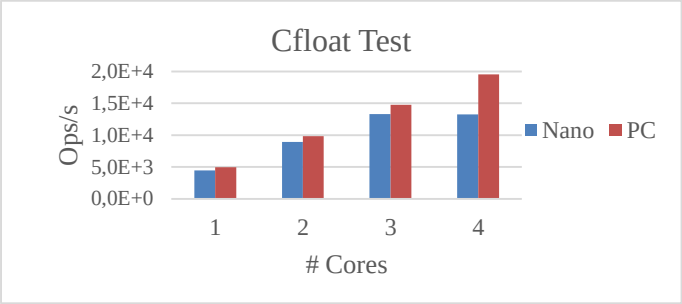

Fig. 4. Devices Performance in the Cfloat Test

The results in **Fig. 4** show the performance of the CPUs when working with float operations. The difference between the ARM and x86 CPU is 9.6% on cores 1, 2 and 3. The core 4 of the ARM CPU suffers a bigger difference. The drop in core 4 is due to the resource competition between the test and the Operating System (OS) on the Nano.

It is evident that the computational capacity of ARM CPUs are similar to x86 CPUs even though they have inferior characteristics. It is possible that the Nano OS does not have a good management in the pthreads and malloc tests generating the exposed results, it should also be taken into account that the ARM CPUs have a reduced set of operations and this negatively impacts the results of the tests. With the exposed in the three figures (**Fig. 2**, **Fig. 3** and **Fig. 4**) the objective of measuring the performance of the devices is covered. **Fig. 5** presents the energy efficiency of the devices.

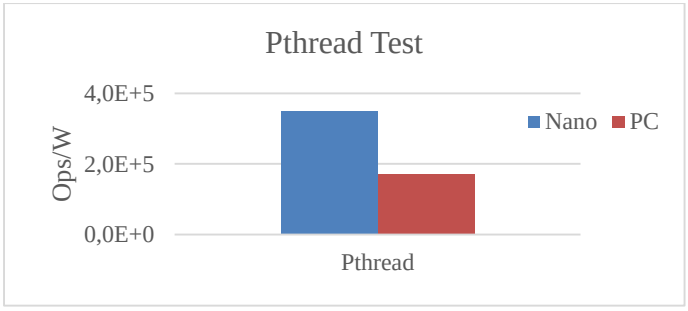

Fig. 5. Devices Energy Efficiency in the Pthread Test

In terms of energy efficiency, the behavior changes, the ARM CPU demonstrates twice the efficiency of the x86 CPU. This is because the Cortex A57 is focused on giving the best performance for the lowest power consumption. **Fig. 6** illustrates the energy efficiency of the memory of the devices.

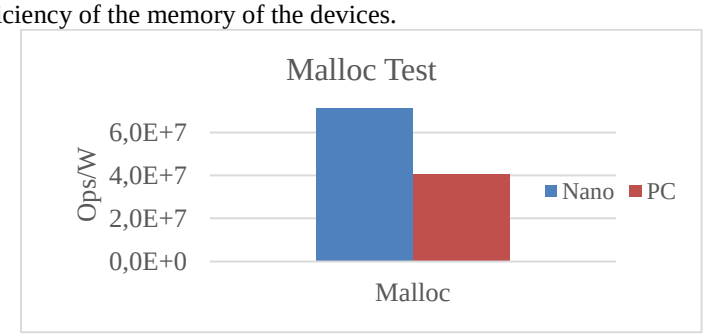

Fig. 6. Devices Energy Efficiency in the Malloc Test

In the malloc test, the efficiency of the Nano is 1.7 times that of the PC, but the performance of the PC is 6 times higher. The same happens in the pthreads test. This opens the discussion between performance or energy efficiency when deploying an application, opting for one or the other depends entirely on the deployment context. For this reason, a decision is not made in this paper as it is not the objective of this work. **Fig. 7** presents the energy efficiency of the cfloat test.

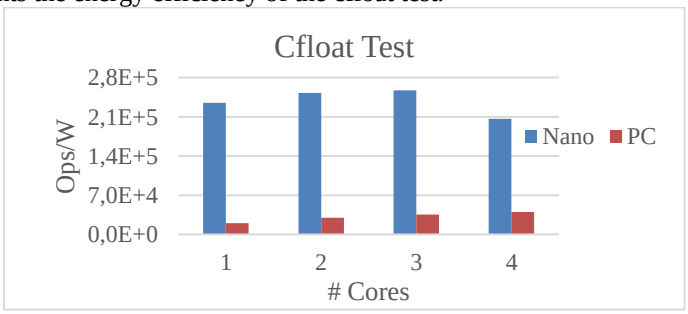

Fig. 7. Devices Energy Efficiency in the Cfloat Test

The difference in energy efficiency of floating point computing of ARM CPUs is abysmal, approximately 11 times higher than that of x86 CPUs. This tips the balance in favor of embedded devices in the efficiency area. Reinforcing the point made in **Fig. 4**, ARM CPUs and embedded devices would be a good choice for implementing applications that have a strong computational component and, if budget or power consumption is a constraint, makes them excellent candidates. But in general, the PC best implements the application. Still the main objective of this paper is not to decide which is better but to demonstrate the benefits of the ACDC methodology.

Once all the steps of the methodology have been completed, the conclusions of the work are presented, but in this case the conclusions generated by the methodology are presented.

Conclusions

The exercise presented in section 4 shows how the methodology allows a detailed observation of the application and the selected devices, equally it is possible to tailor the development of the methodology to meet the needs of the application, the devices that meet the needs of the application and the objectives that are set to evaluate the application or the devices.

The ACDC methodology seeks to generalize the steps to follow to evaluate a Post-Moore architecture, in the first step an application, benchmark, methodology or testbed is selected, it is characterized to know its needs and the proposed steps are followed.

If the ACDC methodology is compared to others, it's much easier to implement different measurement mechanisms. Each step in the methodology generates results that contribute to improve the evaluation results

Further works

The work focused only on CPU, in future studies it's intended to extend the synthetic testbed to other types of hardware such as GPU or FPGA. The methodology was tested on single devices, but it's interesting to see its implementation on agglomerated devices. If the catalog of tested embedded devices is increased, implementation guidelines can be developed that simplify the deployment of the application on Post-Moore devices.

References

[1] J. Dongarra and P. Luszczek, "LINPACK Benchmark," *Encyclopedia of Parallel Computing,* 2011.

[2] E. Strohmaier, J. Dongarra, H. Simon and M. Meuer, "Top 500 The List.," Prometeus GmbH, [Online]. Available: http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html. [Accessed 20 5 2021].

[3] M. M. Waldrop, "The chips are down for Moore's law," *Nature*, no. 530, p. 144–147, 2016.

[4] T. N. Theis and H.-S. P. Wong, "The End of Moore's Law: A New Beginning for Information Technology," *Computing in Science & Engineering*, vol. 19, no. 2, p. 41–50, 2017.

[5] S. Matsuoka, "Cambrian Explosion of Computing and Big Data in the Post-Moore Era," *In Proceedings of the 27th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '18)*, 2018.

[6] S. Matsuoka et al, "From FLOPS to BYTES: disruptive change in high-performance computing towards the post-moore era," *CF '16: Proceedings of the ACM International Conference on Computing Frontiers*, p. 274–281, 2016.

[7] M. Karp, A. Podobas, N. Jansson and T. Kenter, "High-Performance Spectral Element Methods on," *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 1077-1086, 2021.

[8] I. Kuon, R. Tessier and J. Rose, "FPGA Architecture: Survey and Challenges," *Now Publishers Inc*, 2008.

[9] T. S. Czajkowski et. al, "From OpenCL to high-performance hardware on FPGAs," *22nd international conference on field programmable logic*, p. 531–534, 2012.

[10] S. Lee, J. Kim and J. S. Vetter, "OpenACC to FPGA: A Framework for Directive-Based High-Performance Reconfigurable Computing," *2016 IEEE International Parallel and Distributed Processing Symposium*, p. 544–554, 2016.

[11] D. Vasudevan, G. Michelogiannakis, D. Donofrio and J. Shalf, "PARADISE - Post-Moore Architecture and Accelerator Design Space Exploration," *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 139-140, 2019.

[12] J. S. Young et al, "Experimental Insights from the Rogues Gallery," *IEEE International Conference on Rebooting Computing (ICRC)*, 2019.

[13] J. S. Young et al, "A microbenchmark characterization of the emu chick," *CoRR*, 2018.

[14] S. Lloyd and M. Gokhale, "In-memory data rearrangement for irregular data-intensive computing," *Computer*, vol. 48, no. 8, p. 18–25, 2015.

[15] M. Shantharam et al, "Performance evaluation of scale-free graph algorithms in low latency non-volatile memory," *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, p. 1021–1028, 2017.

[16] A. D. Bader et al, "Graph500 Benchmark 1 (search) Version 1.2," *Graph500 Steering Committee, Tech*, 211.

[17] J. S. Vetter, E. P. DeBenedictis and T. M. Conte, "Architectures for the Post-Moore Era," *IEEE Micro*, vol. 37, no. 4, p. 6–8, 2017.

[18] Nvidia, "Nvidia Jetson Nano," [Online]. Available: https://www.nvidia.com/es-la/autonomous-machines/embedded-systems/jetson-nano/. [Accessed 6 junio 2021].

[19] C. King and A. Waterland, "Ubuntu Manuals, Stress-ng," Canonical Ltd, 22 February 2018. [Online]. Available: http://manpages.ubuntu.com/manpages/bionic/man1/stress-ng.1.html. [Accessed 10 February 2021].

[20] VTA, "Manual del Usuario VTA-84630," VTA, [Online]. Available: https://www.vta.co/wp-content/uploads/2019/09/VTA-84630_manual_bn.pdf. [Accessed 15 Febrero 2021].