Introduction

In this project, we will apply the knowledge you have acquired over the past weeks and put it to full use. It was a subgoal of this course to prepare you for this task. You now have the ability to design, implement, and test your own compute units (e.g., adders and their respective algorithms) in LT-Spice, model them with Python and test them in the end applications such as machine learning and image processing.
We remind you that you that these are cutting-edge concepts on the boundaries of research and development.
We highly encourage you to give your best effort in this task, as it is possible to create a research publication (paper) based on this work (as has been done in previously in our team) and publish them in top-notch conferences and journals. An important potential bonus on top of a deep understanding in this field.

Task

1. For your project, you need to choose an exact memristive PIA compute unit (adder or multiplier)[1] similar to what you have seen in the last two exercises and homeworks (but not the ones we have already presented to you!). You might also choose to form a bigger group (4 people) and work on both the adder **and** the multiplier.

2. After choosing an existing memristive PIA compute unit (this will be an exact PIA compute unit that you need to get from a paper), your task will be to come up with an own algorithm to convert this compute unit to an approximate version of it. Use all your wits. Note that we have been doing this in the homework and exercise group and talked about how to do it and there were some novel ideas by yourselves too. Your work will be to find a way to make it approximate! This could either mean to come up with a different circuit design, or it might be the same design with a different algorithm, etc.

3. Use LT-Spice for to

    - a) simulate the exact compute unit that should be completely ready to run upon simulation, also measure the power consumption, computational time (in seconds and number of steps) and the number of memristors used for this unit. This will be recorded and used later in the Python code that will model and simulate the effect of using this exact memristive PIA compute unit and track the overall computational costs/time/number of memristors in the end application.
    - b) simulate the approximate version of this unit you came up with yourself using LT-Spice. Just like before, we want to record the power consumption, computational time and number of memristors needed to perform the computation compared to the exact version. You are free in this task to create something new and let your creativity flow. The important part is that it should do the task (add or multiply). Notice that we leave you free to choose the size of the unit you want to approximate yourself (it must not be necessarily one bit, you can also consider a two/three bit adder as the base and make an approximate version of it). Like it is always in science, you will not know how good your solution may be. Ideally (and hopefully), no one has thought of your solution before and you might create something new that will add value to this field.

4. After having both, the exact and approximate compute unit (possibly for both adder and multiplier) with all the **power consumption, time and number of memristors** recorded for the entire logical truth table, you are ready to compare the effectiveness of your algorithm in in Python (as a model/simulator of your end application). For this you have also been prepared. Just like you saw in the exercise group, you will modify the (or even programm your own) 1-bit (or depending on your compute unit N-bit) adder/multiper function to represent the hardware you simulated above.

    - If you worked with an adder and made it approximate, now you need to emulate a multiplier using your adder circuit. See additional notes for more information in this regard.

[1] Your choice of compute units must be memrtistive IMC-based and ideally include stateful logic (e.g., those using IMPLY Logic, MAGIC, FELIX, TMSL, SIXOR, etc. or a combination of them).

You will be asked to have full control of how much you approximate (just like we saw in one of the homeworks, where you could choose the number of approximate units). However, you can start with the minimum level of approximation (e.g., 1 bit approximation). Having implemented the adding and multipying functions in Python (they should let you add integer values, and they will be built upon your 1 or N-bit compute units you simulated above) you can now perform computations to test the effectiveness.

5. For testing your exact/approximate adder/multiplier compute units, you should test it in **two** examples of machine learning **and two** examples of image processing. In the case of machine learning, you can test your algorithm on the already existing LeNet that you saw in one of the homeworks. Further, use a network/machine learning application that you have not already seen. You need to follow similar steps to be able to simulate both exact and approximate versions and compared them. In the case of image processing, you can test your compute units on the algorithm you already have (for example, low-pass or high-pass filter, to smooth or blurring an image, something you have already seen in the exercise group). In addition to that, test your algorithm on an image processing application that you have not already seen (e.g., edge detection or grayscale filter). You can use any of the open-source implementation of such image processing algorithms online as the base and follow the procedure you went through to be able to simulate and implement both exact and approximate version in a comparable fashion.

6. Record and report what the power consumption, delay (calculated using number of steps/cycles or actual time from the LT-Spice simulation, not the execution of the Python code on your computer!), number of memristors, etc. is (for each level of approximation). Compare your approximate algorithm with the exact version in terms of all respective parameters (what you learned in the course regarding performance and accuracy criteria, e.g., PSNR and similarity index for image processing and classification accuracy for machine learning). What is the performance gain and the quality sacrifice? Check if the quality of the output is acceptable or not.

7. Repeat Step 4 to 6, however, after step-by-step increase in the level of approximation. Continue the process until the quality level of output becomes unacceptable. That is, to find out how far you can push approximation before the sacrifice in quality is discernible and not anymore a reasonable compromise for the performance gain. E.g., depending on the level of approximation (how many bits of approximation) the similarity index for image falls below 90% compared to the precise version of the operation (90% being a rough figure), or visually the maximum level of approximation until the resulting image gets too blurry. Or the classification accuracy drops more than 1% or 5%.

Additional Notes

1. What we are asking you were previously done by former students in the group (one was a BSc student who did this for his BSc thesis, from scratch and alone). You can see these papers and how they went through the process and how they reported it in C51, J26 (the latter has an initial conference version too, which is C43) on the Eclectx website, publication list (https://eclectx.org/publications.html). They applied the technique only on image processing applications but you apply them to both image processing and machine learning to see the (potential) differences in the boundary for these two different applications (what may be good in one application is not necessarily good in the other and vice versa). For the base memristive compute units (exact ones), you can use the literature. The papers mentioned here cite many of such exact version of these compute units and you can find more using a literature search. We are happy to consult and help you if you have any question or concerns regarding selection of such a base exact compute unit. The task of coming up with an approximate version is still up to you.

    - Please remember that you cannot choose the exact compute units that were already made approximate in C51 and J26 as your base design. We already showed them to you in the class and you have done their simulation as a part of your exercise or homework.
    - You can also choose to not use the literature and an existing exact compute unit as your base and come up with a completely new exact compute unit. However, this would be significantly more challenging and we do not recommend it since you may get stuck. You should do that only if you are very confident of your skills in memeristive circuit design and don't mind the additional work, potential setbacks etc. Even though we do not recommend, we also do not stop you from doing so.

2. In the case that you choose to do both adder and multiplier compute unit, you should/can/better be 4 people in your group. Otherwise you should be 3 people. In the case that you choose not to implement e.g. the multiplier compute unit in LT-spice (a group of 3), you will still need to do approximate

multiplication in Python! For this you will have to search how one can do multiplication out of only addition! (e.g. see e.g. https://en.wikipedia.org/wiki/Binary_multiplier). You can use the same procedure that one needs to make (exact) multiplication out of many (exact) additions, to build an approximate multiplication our many approximate additions.

- Of course, you can also be creative and build in more approximation in the way you make multipliers out of adders. However, this additional level of approximation may be too much and detrimental to your end application. So like before, if you want to do this, to it step by step. Start with no or very little approximation and increase the approximation level.
- You should notice that you either have to do these tasks in LT-Spice (which is the interesting and possibly harder to do) or you must do it in Python (building upon your other compute unit which will be slightly easier but still not entirely trivial). This way, independently of what compute unit you choose, your python programm will be able to perform exact and approximate addition and multiplication (all four combination). Thus you can do the task of machine learning and image processing as well as record and compare the effectiveness of your algorithm in both exact and approximate version.

3. If you choose to only implement a multiplication compute unit in LT-Spice, you can decide whether you adders are going to be exact or approximate. Both options are acceptable. In this specific case (i.e., implementing only a multiplier in LTSpice and making it approximate, as a group of 3 people), you are allowed to use the approximate adders we have provided you already.

Evaluation

1. At the end of your project, you will be asked to hand in a report that explains your work (method, findings, comparisons) in detail. This report should be approximately 2-3 pages (IEEE double column format) per person involved in your project (i.e., 6-12 pages). You can take the aforementioned C51 paper as a guiding example of how this report should look like.
   - Once you register your team, we will provide you with a latex file (on Overleaf) such that you can directly write your report without worrying about the formatting of it.
2. You will need to hand in all the files required to reproduce and validate your work. That is, (i) LT-Spice files for exact and approximate simulations, one bit and multibit (at least 2 bits for multibit, ideally 3 or 4 bits) and other necessary files for it (like the CSV for control operations), (ii) All the Python files used to simulate the applications, (iii) All test data files used as input for testing the algorithms (e.g., images used for processing and classification), (iv) the original version of all the figures/plots appearing in the report.
3. In addition to the report and files, you will be asked to make a short video (at least 3 mins) to explain your work. This will be done instead of a final presentation. Hence, we will post it on Moodle for your other classmates to check out your work. You are also welcome to make two videos, i.e., one short (around 3 mins) and one long (around 10-15 mins) but you don't have to. One video is mandatory, two is optional.

We believe making a publication out of your work is a win-win situation for all parties involved (although not a requirement or part of your evaluation). We will try to help you do that the best we can. So feel free to ask your questions and get us involved early, to make sure the output of your work is not just a run of the mill project, but a high-quality research work that can be eventually published (ideally, with as little extra work as possible) as a new contribution to the field.

Due Date

The assignment is due on [29/02/24].
You can ask us question in the planned Q&A sessions in January or contact us via email to ask your questions (or potentially arrange an extra meeting, if possible).