Supplementary Material: Analog In-Memory Computing Attention Mechanism for Fast and Energy-Efficient Large Language Models

1 CMOS layout

We design a custom CMOS layout of the proposed gain cell and charge-to-pulse circuits. In this study, the circuit simulations were done in TSMC 28 nm silicon CMOS technology. We used this conventional design style as a proof of concept to demonstrate the capacity of our gain cells-based architecture to perform the attention mechanism. However, CMOS gain cells lead to relatively large area footprint, primarily due to Metal-Oxide-Metal (MOM) capacitors which must be relatively large due to their high leakages. Our layout results show that each cell has a dimensions of $3.9\,\mu\text{m} \times 4.9\,\mu\text{m}$, resulting in an area of $0.08\,\text{mm}^2$ per 64×64 array, or $1.28\,\text{mm}^2$ for one entire attention head (16 sub-tiles). In comparison, the ReLU charge-to-pulse circuitry and its signed variant occupy an area of $0.01\,\text{mm}^2$ and $0.02\,\text{mm}^2$ per attention head, respectively. The Layout of the gain cells storing V values and computing $\phi(S) \cdot V$ is shown in shown in Fig. 1. Note that the Layout of the gain cells storing K has transposed World Lines (WL) and Bit Lines (BL).

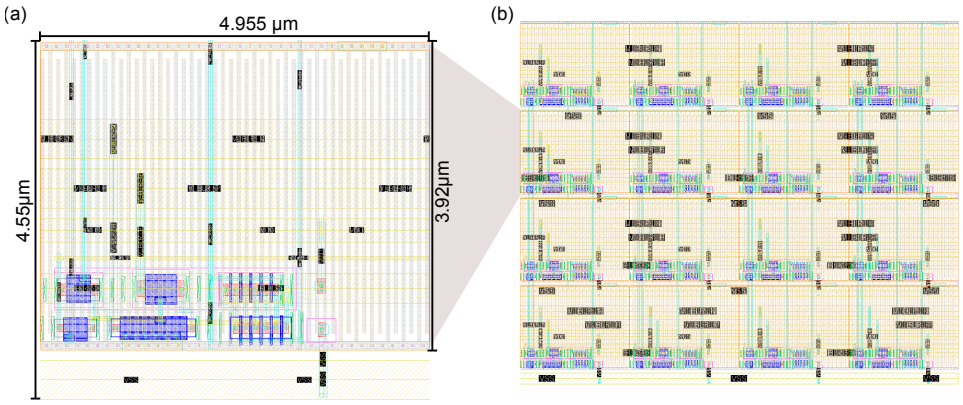

Fig. 1 (a) Single cell layout of a for storing V values and computing $\phi(S) \cdot V$. (b) 4×4 array Layout. Note that apart of the bottom row the hight scales with $3.92\,\mu\text{m}$

1.1 ReLU charge-to-pulse converter

In this section, we provide additional information on the working principle for the ReLU charge-to-pulse circuit block. This charge-to-pulse circuit operates in three distinct phases: sampling, discharge, and reset. During the sampling phase, input pulses are applied to the first gain cell array, and the currents generated by the cells are integrated by a capacitor (C_2) in the charge-to-pulse circuit. This capacitor also utilizes the wire capacitance of the word line. In the discharge phase, the voltage of the capacitor C_2 is discharged with a constant current controlled by the bias voltage V_b. However it is important to note that the system employs an energy saving scheme by checking the voltage on the integrating capacitor V_{cap} and only preforming the discharge in case the voltage is positive. An inverter acts as a simple comparator, triggering a pulse of variable width. Finally, in the reset phase, the bit line is reset to the initial bit line voltage to prepare for a new inference step.

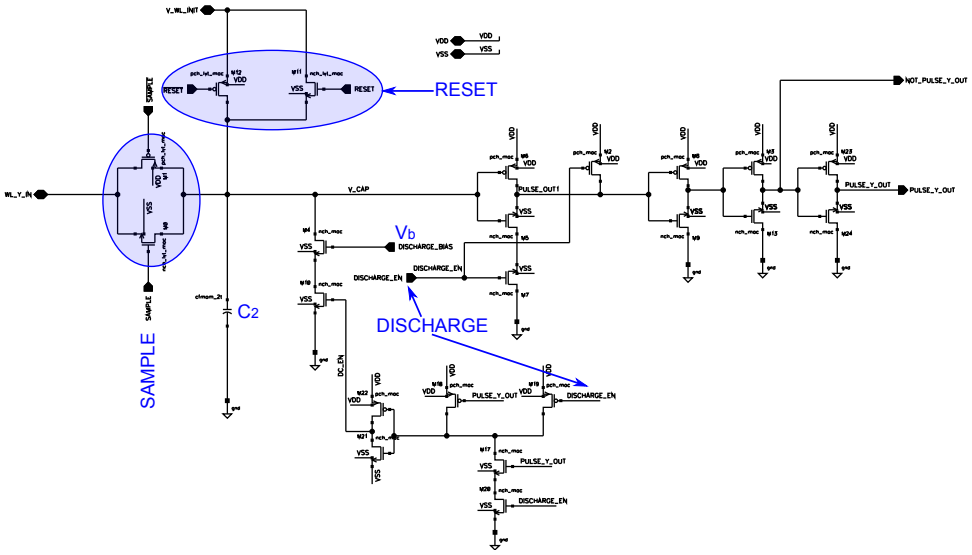

Fig. 2 CMOS schematic of the ReLU charge-to-pulse converter

2 Signed charge-to-pulse converter

To implement a signed charge to pulse circuit, the main difference from the circuit in 1.1 is the addition of a charge-up path and a D Flip-Flop. This serves the following purpose: at the end of the sampling stage, the D Flip-Flop captures the polarity of the voltage on the capacitor and stores it for subsequent operations. This stored sign determines whether a charge or discharge is applied to the capacitor voltage. The pulse-forming circuit is now slightly more complex to ensure consistent, high-active output pulses. Ultimately, the circuit outputs both the sign and the output pulses.

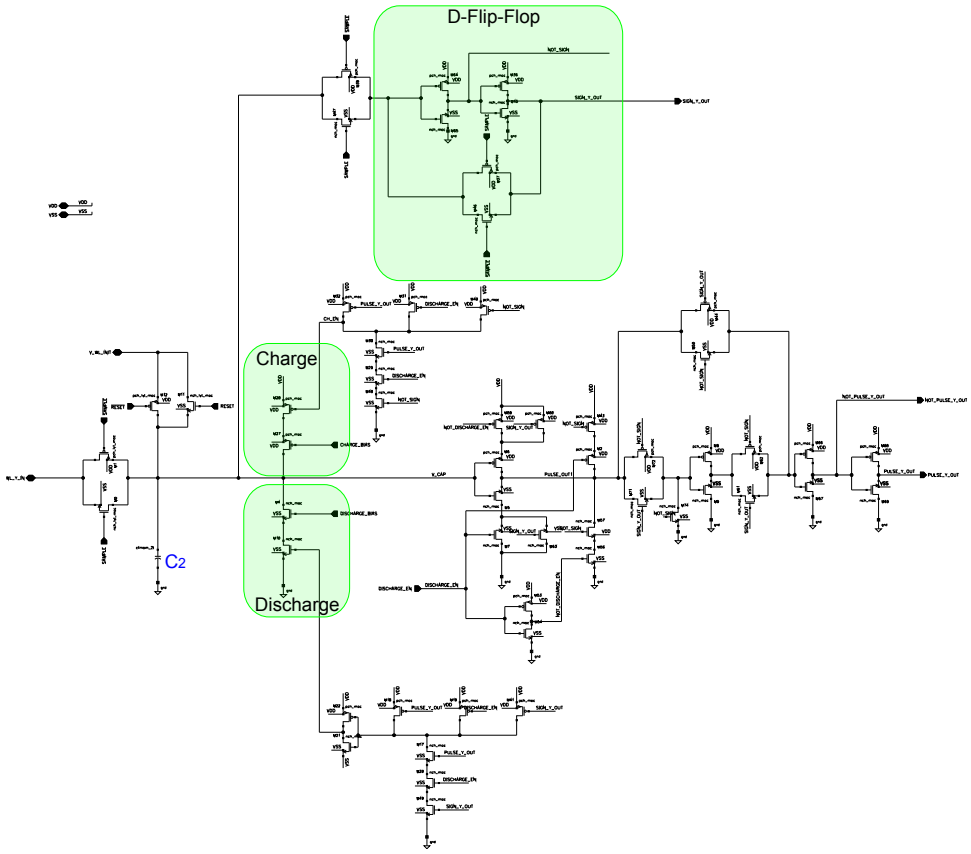

Fig. 3 CMOS schematic of the signed charge-to-pulse converter

Fig. 3 shows the circuit schematics. The two distinct charge up and charge down behaviours given a certain SIGN are displayed in Fig. 4.

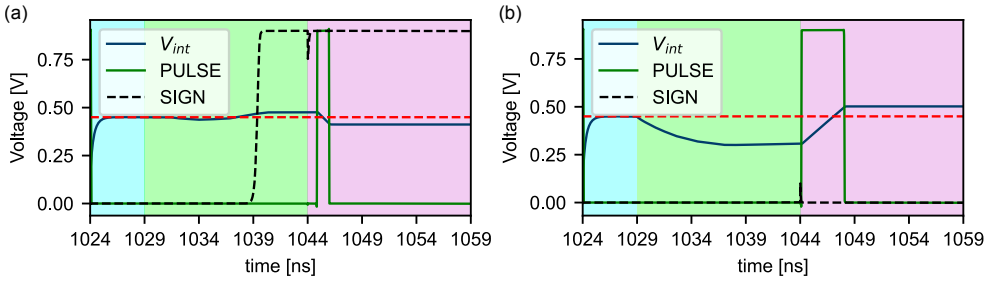

Fig. 4 Example for a MAC result with positive signed (a) and negative sign (b) featuring distinct charging behaviours.

3 Adaptation Algorithm

3.1 Pseudo-code

Algorithm 1 Pseudo-code for the adaptation algorithm used to map the nonlinear model to the linear model.

g^L ▷ Linear model's modules
g^{NL} ▷ Nonlinear model's modules
$INPUT_0 \leftarrow SAMPLE$ ▷ Get text from the dataset
$\epsilon \in]0, 1[$ ▷ Error threshold
$\gamma \in]0, 1]$ ▷ Measure rate
$\Gamma \in]0, 1]$ ▷ Adaptation rate
$ERROR \leftarrow 1$
for i in MODULES INDEXES **do** ▷ Inference on the linear model
    $INPUT_i \leftarrow g_i^L\left(INPUT_{i-1}\right)$
end for
for i in SCALING MODULES INDEXES **do** ▷ Measure the linear model statistics
    $x \leftarrow INPUT_i$
    $y \leftarrow a_i^L x + b_i^L$
    $\sigma_i^L \leftarrow \sqrt{\frac{1}{n}\sum_j^n \left(y_j - |y|\right)^2}$
    $\mu_i^L \leftarrow |y|$
end for
while $ERROR > 0$ **do** ▷ Adaptation loop
    $ERROR \leftarrow 0$
    for i in MODULES INDEXES **do** ▷ Inference on the nonlinear model
        $INPUT_i \leftarrow g_i^{NL}\left(INPUT_{i-1}\right)$
    end for
    for i in SCALING MODULES INDEXES **do** ▷ Measure statistics and adapt scaling
        $x \leftarrow INPUT_i$
        $y \leftarrow a_i^{NL} x + b_i^{NL}$
        $\sigma_i^{NL} \leftarrow \gamma\sqrt{\frac{1}{n}\sum_j^n \left(y_j - |y|\right)^2} + (1-\gamma)\,\sigma_i^{NL}$
        $\mu_i^{NL} \leftarrow \gamma|y| + (1-\gamma)\,\mu_i^{NL}$
        if $|\sigma_i^{NL} - \sigma_i^L| > \epsilon$ **then**
            $ERROR \leftarrow ERROR + 1$
            $a_i^{NL} \leftarrow \Gamma a_i^{NL}\frac{\sigma_i^L}{\sigma_i^{NL}} + (1-\Gamma)\,a_i^{NL}$
        end if
        if $|\mu_i^{NL} - \mu_i^L| > \epsilon$ **then**
            $ERROR \leftarrow ERROR + 1$
            $b_i^{NL} \leftarrow \Gamma\left(b_i^{NL} + \mu_i^L - \mu_i^{NL}\right) + (1-\Gamma)\,b_i^{NL}$
        end if
    end for
end while

3.2 Generalization to Different Nonlinear Functions

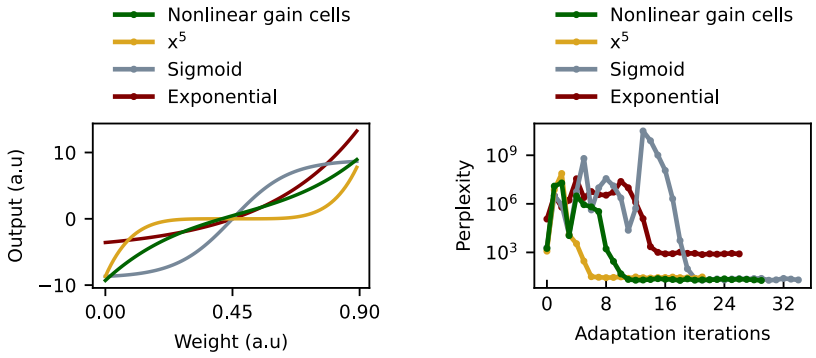

Fig. 5 (a) Different nonlinear functions tested in place of the gain cells nonlinearities. (b) Evolution of perplexity (lower the better) during the adaptation algorithm when our attention model is implemented with different nonlinearities applied on the stored keys and values.

In this experiment, we evaluate the capacity of our adaptation algorithm to generalize to other nonlinearities than the one modelling the gain cells (see Fig. 5). We perform the dot-products of the attention mechanism with different nonlinearities applied to the stored keys and values. The different functions tested are: $f(x) = \alpha(x-\beta)^5$, $f(x) = \alpha sigmoid(10(x-\beta))$, and $f(x) = \alpha e^{3(x-\beta)}$, with α and β chosen to yield to similar ranges for the different functions.

We see that our adaptation algorithm manage to reduce the perplexity drastically, except for the exponential function. The high asymmetry of the exponential function prevents the network to yield good accuracies. The adaptation algorithm manage to reduce the perplexity for functions which are anti-symmetrical even if they are highly nonlinear, such as x^5 (perplexity=29) or *sigmoid* (perplexity=21).