Supplementary Material

1 Selection Unit circuit details for positive and negative numbers

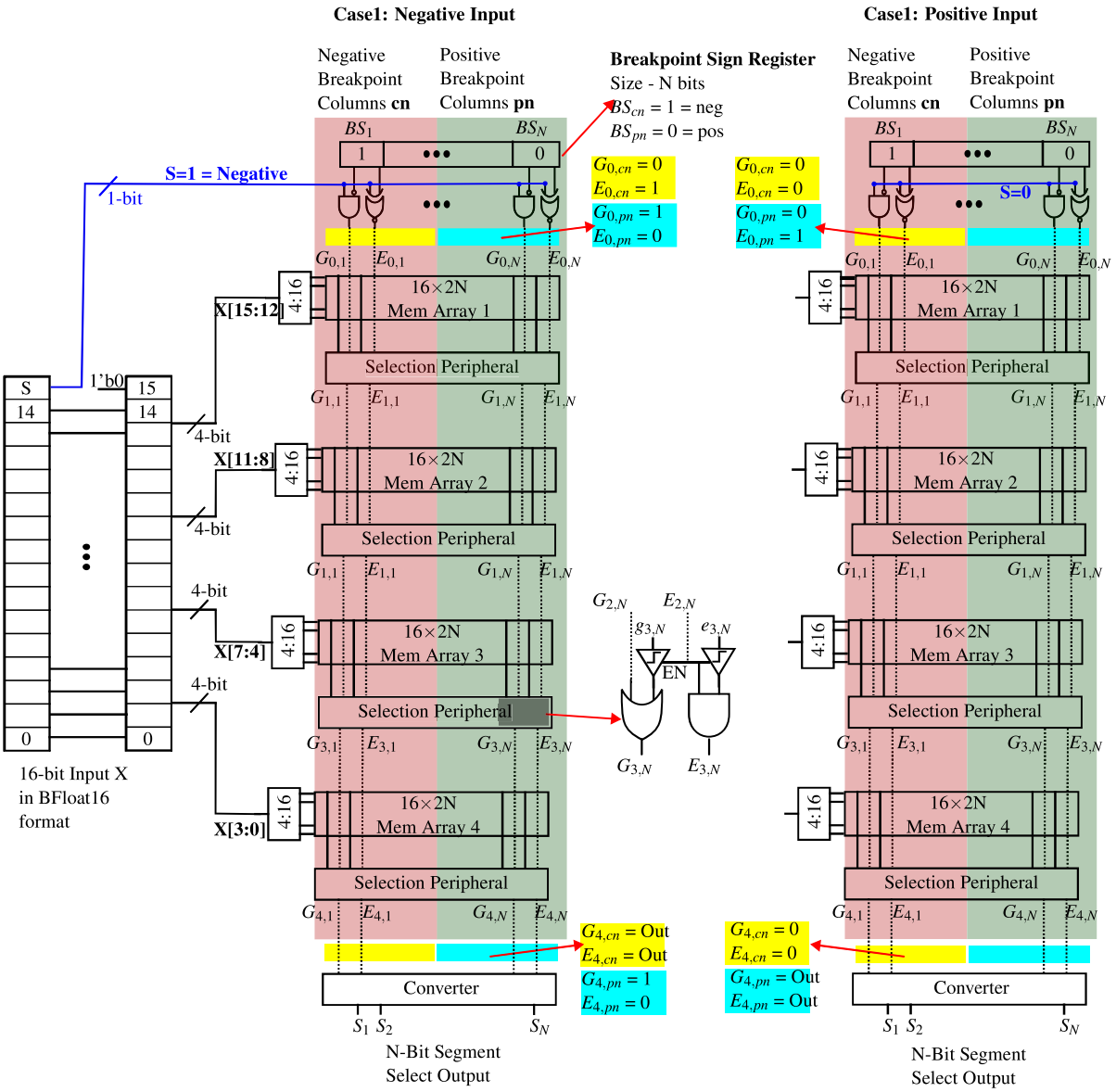


Figure 8. CIM-based Segment Selection Unit's operation for positive and negative numbers

The CIM-based Segment Selection Unit is capable of handling negative and positive breakpoints. The memory array is divided into two sections: one for negative breakpoints (*cn*) and one for positive breakpoints (*cn*). The partitioning is determined by the 'Breakpoints Sign Register', where each register cell controls the sign of two memory columns of the memory arrays. Columns with a register value set to logic-1 are designated for negative breakpoints, and those with logic-0 are designated for positive breakpoints. The breakpoints are stored in the memory array in two formats: one-hot encoding and thermometer code. During this process, the sign bit of the breakpoint is ignored. Positive breakpoints are stored in ascending order, while negative breakpoints are stored in descending order, as the ascending order of negative numbers (ignoring the sign bit) is equivalent to the descending order of their absolute values.

If the input is negative, only the columns (*cn*) that have negative breakpoints are enabled by setting the $E_{0,cn}$=1 and $G_{0,cn}$=0. The columns (*pn*) with positive breakpoints are disabled by setting $E_{0,pn}$=0, which intern results in $E_{4,pn}$=0. At the same time,

set $G_{0,pn}$=1 to make $G_{4,pn}$=1, as the positive breakpoints in *pn* columns should be greater than the negative input. The signals $G_{4,cn}$ and $E_{4,cn}$ output corresponds to the actual selection output. In case the input is positive, only the *pn* columns are enabled by setting the $E_{0,pn}$=1 and $G_{0,pn}$=0. The *cn* columns are disabled by setting $E_{0,cn}$=0, which intern results in $E_{4,cn}$=0. Setting $G_{0,cn}$=0 along with $E_{0,cn}$=0 will make $G_{4,cn}$=0. This indicates that the *cn* columns are less than the negative input. The signals $G_{4,pn}$ and $E_{4,pn}$ output corresponds to the actual selection output.

2 Non-Idealities and Array Sensing Latency

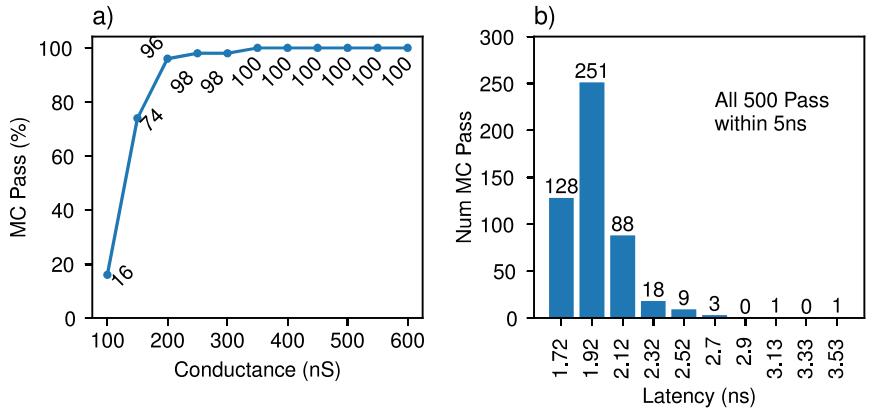


Figure 9. Array Monte Carlo simulations and latency results. a)Impact of CMOS process variation on the 1-bit current sensor (I_{ref}=100nA) and selection peripheral unit accuracy for various LRS conductances with HRS fixed at 20 nS – array size = 16×64. Here, the Monte-Carlo (MC) pass percentage depicts the number of times the 1-bit current sensor and selection peripheral can correctly sense the array data and select the appropriate Segment. b) Impact of CMOS process variation on the latency of 1-bit current sensor (I_{ref}=100nA) and memory array with LRS = 400 nS, HRS=20 nS, and array size = 16×64.

Typically, the conventional analog CIM accelerators are prone to output errors due to various memory array non-idealities, such as bit-line IR drops, device-to-device variability, memristor cycle-to-cycle variability, read noise, and conductance drift over time. To address this issue, the KA-CIM tile architecture is designed to ensure all arrays, including the CIM-based Segment Selection unit, operate like conventional memory elements that access data from only one row at any given instance. Additionally, the stored data in each cell is restricted to a 1-bit binary value, minimizing the effects of device variability and drift. So the main source of error in our architecture will be due to the CMOS process variation within the array sensing and peripheral circuit. Figure 9 presents the experimental results related to the impact of CMOS process variation on the KA-CIM tile and its output error. For this experiment, we considered a low power, low area, current mirror-based, 1-bit current sensing ADC (similar to[49]) with 100 nA reference current as the array data sensing circuit. Furthermore, in this experiment the HRS is fixed at 20 nS, and the LRS is varied from 100 nS to 600 nS. 50 Monte Carlo simulations are conducted for each memristor LRS conductance value. The goal of this experiment is to find the lowest LRS value with which we can have zero sensing error. When the memristor LRS value is in the range of 100 to 200 nS the current on the bit line is very close to the reference current and the CMOS process variation within the current mirror-based sensing circuit dominates and results in incorrect data read. The memristor cell variation will further add up to this error. As the conductance of LRS is increased, the resulting current on the bit line moves farther to the reference current to clearly distinguish the cell value despite worst-case CMOS and memristor variations. From 350 nS onwards the Monte Carlo simulation is 100%, i.e., all the data is sensed correctly. we consider 400 nS as the LRS (to have an additional gap) and 20 nS as the HRS for our next 500 Monte Carlo simulations All the stored data in each of the 500 Monte Carlo simulations is sensed correctly with the maximum sensing latency of 3.53 ns. Considering the worst-case conditions, the sensing latency of the memory array is set to 5 ns for all the other experiments. These experiments prove that the CMOS and Memristor array variations can be addressed without incurring errors. The CMOS process variation has no impact on the other KA-CIM tile circuits like selection peripheral logic and MAC unit as these are all based on digital logic gated that the standard technology library would have taken care. Overall, KA-CIM architectural decisions ensure that the impact of non-idealities is the lowest and affects minimally the output value.

3 Tile Energy Breakdown

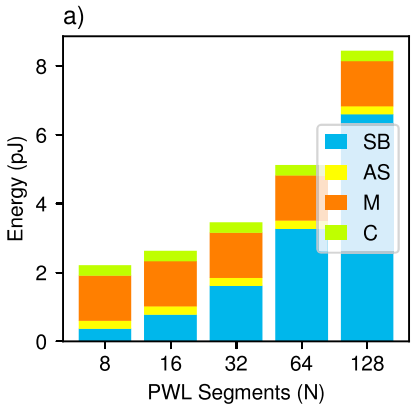


Figure 10. a) Energy breakdown of a single KAN-CIM Macro, where SB = Energy of Segment Selection Unit, AS = Data Access Energy from Slope and Intercept Buffer, M = MAC unit energy, and C = Communication Energy (tile related only)

The energy increase in the KA-CIM tile corresponds to the sensing overhead from larger N. The larger share is for Segment Selection Unit, as it increases the number of columns to $2 \cdot N$ in four arrays, thereby also increasing the number of parallel sensed data. On the other hand, the M_N and Y_N buffers have to increase the number of rows to N, but the the number of columns remain same. Notably at $N = 32$, the energy consumed by the Selection Unit (for selecting the PWL segment) and the energy required for F(X) computation (i.e. Buffer+MAC+Communication) are equal.

4 Four main opcodes of KA-CIM

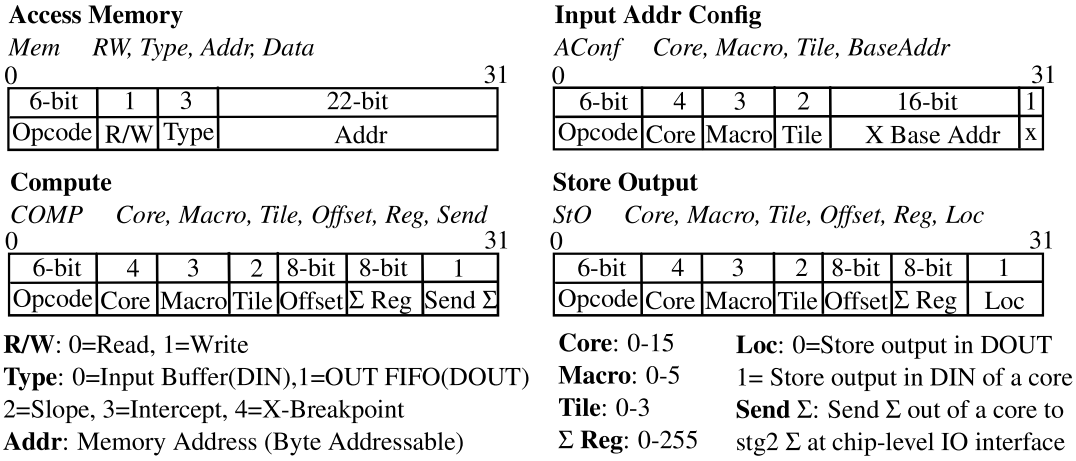


Figure 11. KA-CIM four major instructions.

Figure 11 shows the four major instructions of KA-CIM details and its opcode details. In addition to this KA-CIM also supports other standard instructions like Move, ADD, and MUL.

5 Detailed pipeline operation with sample control flow program

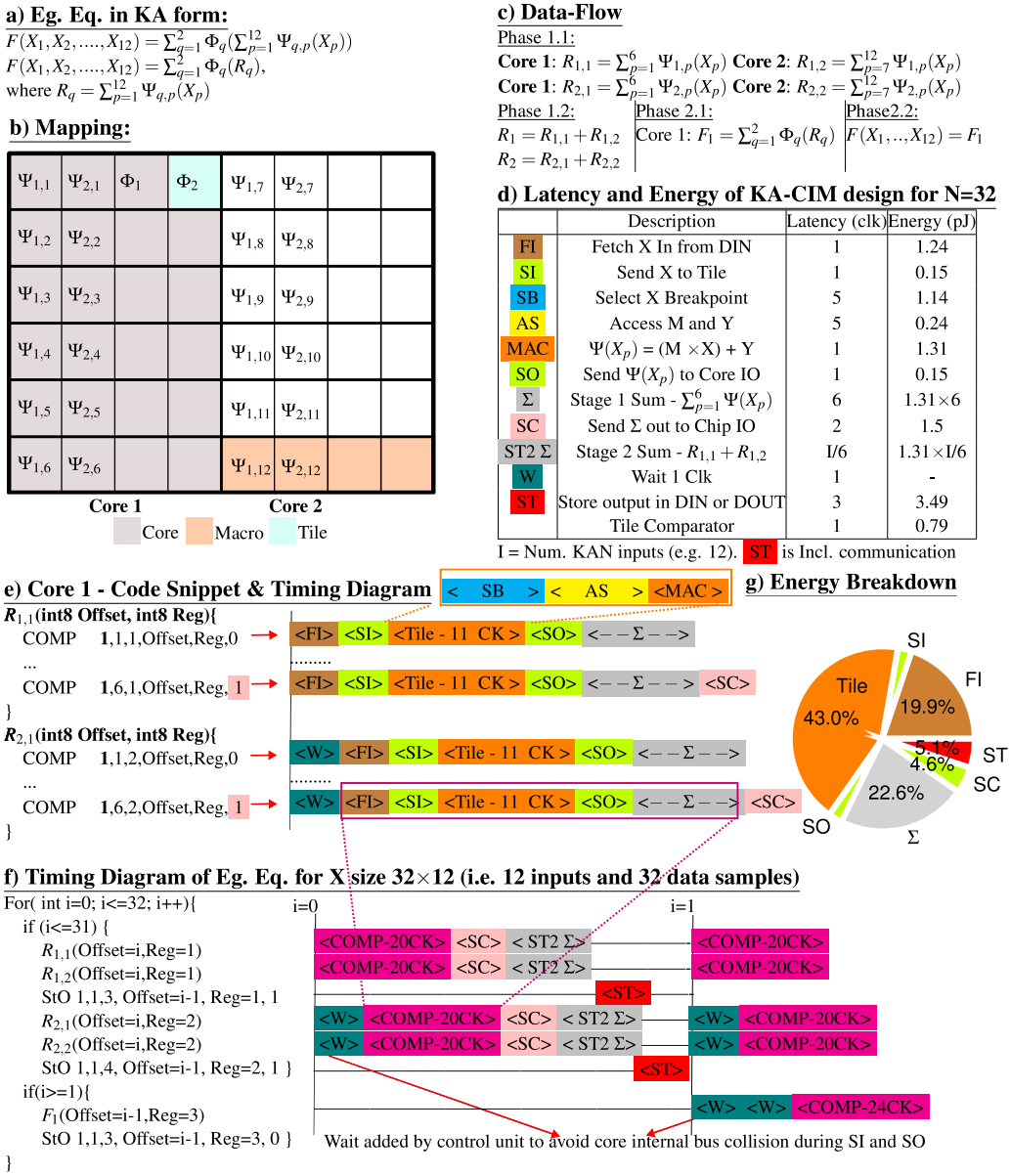
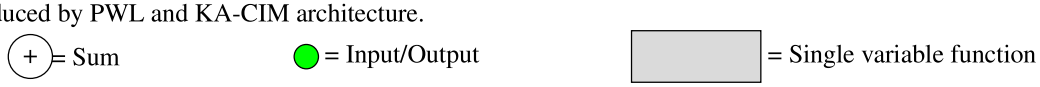


Figure 12. KA-CIM mapping, data-flow, latency, energy, and timing diagram. **The total energy per data sample = 0.16 nJ, total latency per data sample = 57 clk cycles, and throughput = 38.46 × 10^6 Output-Samples/s** for the given equation $F(X_1,X_2,....,X_{12}) = \sum_{q=1}^{2} \Phi_q(\sum_{p=1}^{12} \Psi_{q,p}(X_p))$. The number of PWL segments (N) per function (e.g. Φ & Ψ) is set to 32 and the tile partitioning is not considered here.

6 Conversion of all example multi-variable functions to KAN

In this section, we present the manual conversion of the example multi-variable functions to the KAN format. This approach was chosen to eliminate conversion errors induced by the limitations of automated methods and focus solely on the errors induced by PWL and KA-CIM architecture.



Hodgkin Huxley:

As a demonstration, we only evaluate the equation of three 'gating' variables m, n, h equations on KA-CIM.

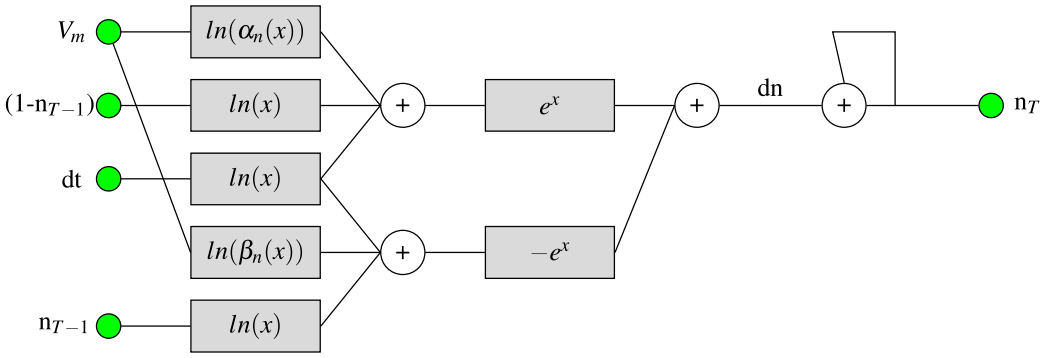
$$\frac{dn}{dt} = \alpha_n \cdot (1-n) - \beta_n \cdot n \qquad \alpha_n = n(\frac{0.01(10-V)}{exp(\frac{10-V}{10}) - 1})$$

$$dn = \alpha_n \cdot (1-n) \cdot dt - \beta_n \cdot n \cdot dt \qquad \beta_n = 0.125exp(\frac{-V}{80})$$

$$dn = exp^{\left(ln(\alpha_n \cdot (1-n) \cdot dt)\right)} - exp^{\left(ln(\beta_n \cdot n \cdot dt)\right)}$$

$$dn = exp^{\left(ln(\alpha_n) + ln(1-n) + ln(dt)\right)} - exp^{\left(ln(\beta_n) + ln(n) + ln(dt)\right)}$$

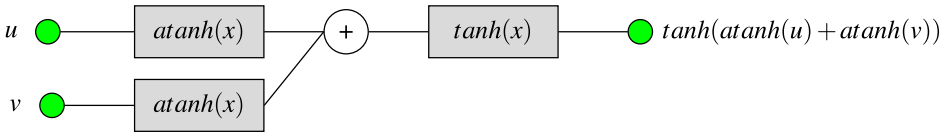
$$n_T = n_{T-1} + dt$$



Similarly, m_T and h_T are also converted to KAN.

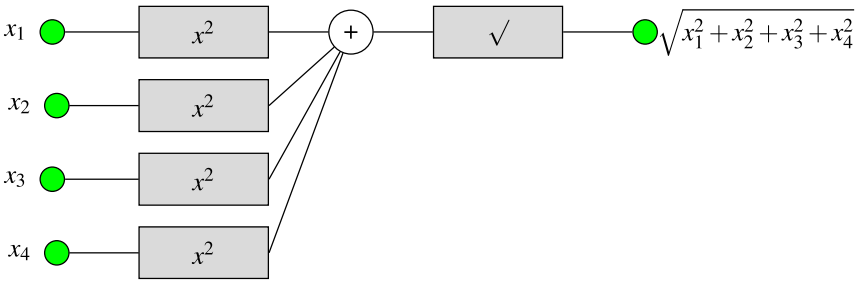
Relativistic Addition

$$\frac{u+v}{1+uv} = tanh(atanh(u) + atanh(v))$$



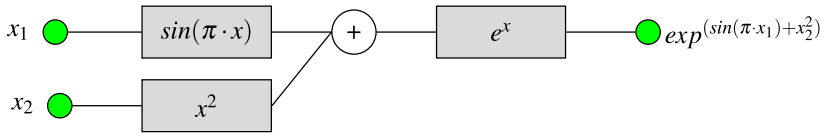
Root-Sum-Square

$$\sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2}$$



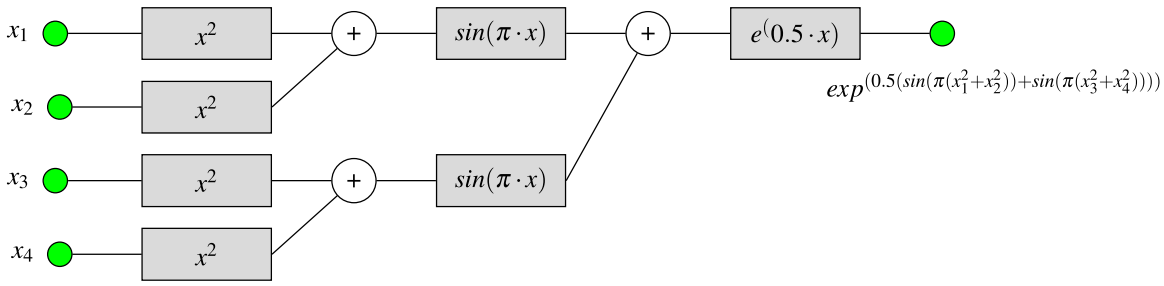
Trigonometry Equation - 1

$exp^{(sin(\pi \cdot x_1)+x_2^2)}$



Trigonometry Equation - 2

$exp^{(0.5(sin(\pi(x_1^2+x_2^2))+sin(\pi(x_3^2+x_4^2))))}$



7 Reduction in the total number of KAN layers with multiplication node

Hodgkin Huxley: KAN without multiplication node

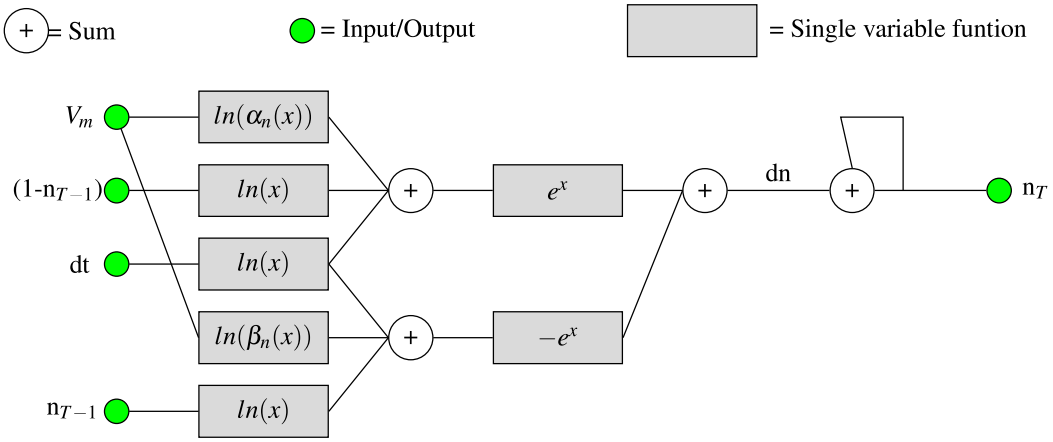
$$\frac{dn}{dt} = \alpha_n \cdot (1-n) - \beta_n \cdot n \qquad \alpha_n = n(\frac{0.01(10-V)}{exp(\frac{10-V}{10})-1})$$

$$dn = \alpha_n \cdot (1-n) \cdot dt - \beta_n \cdot n \cdot dt \qquad \beta_n = 0.125exp(\frac{-V}{80})$$

$$dn = exp^{\left(ln(\alpha_n \cdot (1-n) \cdot dt)\right)} - exp^{\left(ln(\beta_n \cdot n \cdot dt)\right)}$$

$$dn = exp^{\left(ln(\alpha_n)+ln(1-n)+ln(dt)\right)} - exp^{\left(ln(\beta_n)+ln(n)+ln(dt)\right)}$$

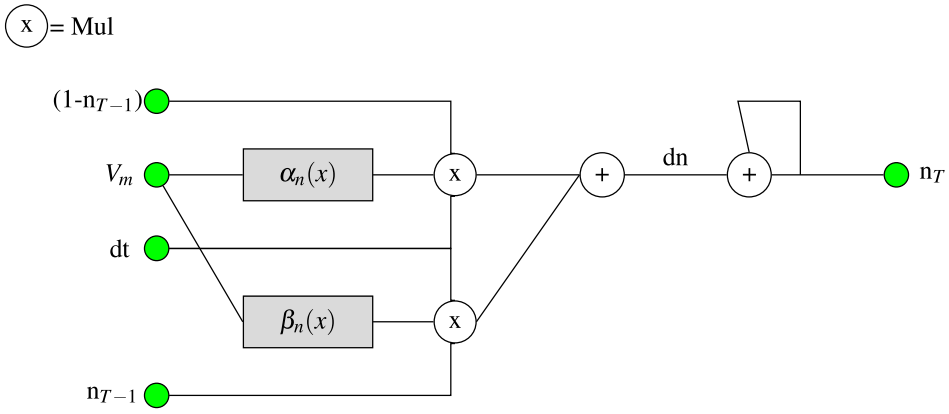
$$n_T = n_{T-1} + dt$$



Hodgkin Huxley: KAN with multiplication node

$$\frac{dn}{dt} = \alpha_n \cdot (1-n) - \beta_n \cdot n$$

$$dn = \alpha_n \cdot (1-n) \cdot dt - \beta_n \cdot n \cdot dt n_T \quad = n_{T-1} + dt$$



8 Extended comparison of KA-CIM vs 1000 TOPS/W CIM

Table 4. Comparison of KAN-CIM vs Conventional-CIM for scientific applications.

Application 1: Predicting the signature of a knot[12] - Inference			
Network Total Num Operations	KAN = [17,1,14] 31 Functions + Sum	MLP = [17,300,300,300,14] 189300 MACs	
Architecture	**KA-CIM**	**CIM-1 (100 TOPS/W)**	**CIM-2 (1000 TOPS/W)**
Energy per output sample (pJ)	182.68	3786	378.6
Latency (ns)	60	224	224
Output Samples/s	466.66 M	250 M	250 M
Norm. Energy Delay Product	1	77.37	7.73
Total Array Size	19.375 KB	184.86 KB	

Application 2: Trigonometry Equation: $F(x_1,x2) = exp(-x_1) \cdot sinh(x_2)$			
Network Total Num Operations	KAN = [2,1] 2 Functions + Mul	MLP = [2,64,64,64,1] 8384 MACs	
Architecture	**KA-CIM**	**Conv-CIM-1 (100 TOPS/W)**	**Conv-CIM-2 (1000 TOPS/W)**
Energy per output sample (pJ)	16.25	167.68	16.768
Latency (ns)	21	224	224
Output Samples/s	47.61 M †	17.85 M	17.85 M
Norm. Energy Delay Product	1	110.06	11.006
Total Array Size	1.25 KB	8.1875 KB	

* For CIM (i.e. Convention-CIM), we consider the following assumptions:
1) Eight-bit data width.
2) All the MACs of a single MLP layer are computed in parallel and require eight iterations to output the final MAC result.
3) Latency for a single MLP layer = $8 \times 1\_Iteration\_Lat$ = 8×7ns =56ns.
4) $1\_Iteration\_Lat = CrossbarRead + ADC + Accumulation$ = 5ns + 1ns + 1ns = 7ns (best-case).
† Using only 2 tiles