**Neurotorch Manual**

**Jonghyun Ko**

Please contact [kofficial@naver.com](mailto:kofficial@naver.com) for further information on Neurotorch.

**Table of Content**

1. **Introduction**
2. **Options**
3. **CIM Architecture, Schemes**
4. **Calculation Model**
5. **Introduction**

NeuroTorch is a simulator tool for system-level evaluation of compute-in-memory (CIM). NeuroTorch references NeuroSim but do not use external DRAM and weight gradient computation (WGC) units in CIM chip. Instead, NeuroTorch adopts Parallel Outer Product Update (POPU) Method for the on-chip training. Compared to the conventional on-chip training using the external DRAM and WGC units, the required hardware resources, such as area overhead, latency, and energy consumption, are much more reduced.

In NeuroTorch. The files for the evaluation of the accuracy, area, latency, and energy consumption of CIM systems are divided into ‘Accuracy.py’, ‘Area.py’, ‘Latency.py’, and ‘Energy.py’, respectively. By executing ‘main.py’ the CIM system evaluation is performed. Execute ‘main.py’ after setting options in ‘Parameter.py’ for the simulation.

The environment of the NeuroTorch development is specified as below:

Python 3.8.8

torch 2.0.1+cu117

torchvision 0.15.2+cu117

CUDA: Cuda compilation tools, release 11.3, V11.3.109

1. **Options (Parameter.py)**

#Chip Specifications  
GlobalBusWidth = 8192 # Global Bus width  
NumTilePerChip = 16 # Number of Tiles per Chip  
NumPEperTile = 4 # Number of PEs per Tile  
NumSubperPE = 8 # Number of SubArrays per PE  
SubRow = 256 # Row Size of SubArray  
SubCol = 256 # Column SIze of Sub Array  
PEBuffer = 'SRAM' # The type of Cache for PE (There are 'SRAM' , 'DFF')  
TileBuffer = 'SRAM' # The type of Cache for Tile (There are 'SRAM' , 'DFF')  
RowsPerADC = 8  
ColumnsPerADC = 8 # ADC sharing using MUX (For Area saving)  
ColumnsPerWeight = 1 # For Onchip Training, Column per weight should be 1 (For Parallel update)  
SRAMRow = 128 # Row Size of SRAM Cache  
SRAMRowF = 8  
SRAMCol = 128 # Column Size of SRAM Cache  
SRAMColF = 20  
BusToTileT = 'HTree' # Type of InterConnect to Tile ('HTree') (We now only coded only about HTree)  
BusToPET = 'HTree' # Type of InterConnect to PE ('HTree' , 'Bus')  
BusToSubT = 'Bus' # Type of InterConnect to SubArray ('HTree' , 'Bus')  
PEAccParr = 32 # Copy of AdderTree for PE to Process Accumulation Parallely. (If 1, only 1 Adder Tree exists)  
TileAccParr = 64 # Copy of AdderTree for Tile to Process Accumulation Parallely. (If 1, only 1 Adder Tree exists)  
GlobalAccParr = 64 # Copy of AdderTree for Chip to Process Accumulation Parallely. (If 1, only 1 Adder Tree exists)  
  
# Elements which is determined by Network  
IMGSIZE = 32  
MaxLayerInput = 32 \* 32 \* 64  
BATCH = 1  
LearnRate = 0.0005  
UpdateMax = 0.0005  
EPOCH = 1  
NUMCLASS = 10 # Number of Answer classes  
  
# Network property  
DATANUM = 100  
NumLayer = 9  
Kernel = 3  
LayerIMG = [32, 32, 16, 16, 8, 8, 8, 4096, 1024, 10]  
LayerInChan = [3, 64, 64, 128, 128, 256, 256, 4096, 1024, 10]  
LayerType = ['Conv', 'Conv','Conv','Conv','Conv','Conv','Conv','FC','FC', 'END']  
IsPoolLayer = [0, 1, 0, 1, 0, 0, 1, 0, 0]  
SpeedUP = [64, 4, 4, 4, 2, 2, 2, 1, 16] # For NeuroSim Comparison  
  
# Elements especially related to the quantization  
Mode = 'Onchip\_Parallel' # What CIM supports (There are 'Inference\_Normal' , 'Onchip\_Normal', 'Onchip\_Parallel')  
Onchip\_ParMode = 'LTP\_only' # Use only LTP or LTD or both LTP, LTD : 'LTP\_only', 'LTD\_only', 'LTPLTD\_both'  
MUXSize = 1 # MUX\_Width Size (1 means Minimum Feature Size)  
ADCType = 'SAR'  
ADCPrecision = 5 # ADC Precision for CIM Acceleration  
ADCMax = 50  
DelEn = 1  
DelPrecision = 14  
DelADCMax = 0.1  
ADCCycle = 1e-9 # ADC clock cycle (SAR ADC)  
InputEncoding = 1 # How many bits per input cycle  
InputPrecesion = 8 # How many bit resolutions per input  
  
#Memory Device Specifications  
Cycle = 1e-9  
Memory = 'Flash' # Type of Memory (There are 'Flash' , 'ReRAM' , 'FeFET' 'IDEAL')  
if Memory == 'Flash':  
 HighVoltage = 9 # The voltage used for a level shifter for Memory Write Operations  
 MemHeight = 2  
 MemWidth = 4  
 ONCurrent = 1e-8  
 OnOffRatio = 10  
 WriteCurr = 100e-15  
 InhibCurr = 0  
 FullTimeLTP = 100 \* 1e-3  
 FUllTimeLTD = 200 \* 100e-6  
 OnePulseLTP = 1e-3  
 OnePulseLTD = 100e-6  
 LTP = 0 # LTP Nonlinearity (if linear, write 0)  
 LTD = 4.7 # LTD Nonlinearity (if linear, write 0)  
 Vari = 0.6 # Device variation  
 LEVEL = 200  
 ReadTimePerCycle = 5e-9 # Required Readtime per cycle  
 VRead = 1  
 UpdateLEVELs = min(2 \* math.floor(UpdateMax \* min(FullTimeLTP, FUllTimeLTD) / (2 \* Cycle)),  
 2 \*\* (InputPrecesion + DelPrecision - math.floor(math.log2(1 / UpdateMax))))  
  
  
if Memory == 'FeFET': # From "Ferroelectric FET Analog Synapse for Acceleration of Deep Neural Network Training"  
 HighVoltage = 3.7 # The voltage used for a level shifter for Memory Write Operations  
 MemHeight = 4 # Not Known  
 MemWidth = 4 # Not Known  
 ONCurrent = 8.9e-8  
 OnOffRatio = 45  
 WriteCurr = 100e-15  
 InhibCurr = 0  
 FullTimeLTP = 32 \* 75e-9  
 FUllTimeLTD = 32 \* 75e-9  
 OnePulseLTP = 75e-9  
 OnePulseLTD = 75e-9  
 LTP = -1.75 # LTP Nonlinearity (if linear, write 0)  
 LTD = 1.46 # LTD Nonlinearity (if linear, write 0)  
 Vari = 0.6 # Device variation  
 LEVEL = 32  
 ReadTimePerCycle = 5e-9 # Required Readtime per cycle  
 VRead = 0.05  
 UpdateLEVELs = 1 #min(2 \* math.floor(UpdateMax \* min(FullTimeLTP, FUllTimeLTD) / (2 \* Cycle)),  
 #2 \*\* (22 - math.floor(math.log2(1 / UpdateMax))))  
  
if Memory == 'ReRAM': # From "Nanoscale Memristor Device as Synapse in Neuromorphic Systems"  
 HighVoltage = 3.2 # The voltage used for a level shifter for Memory Write Operations  
 MemHeight = 4 # Not Known  
 MemWidth = 4 # Not Known  
 ONCurrent = 40e-9  
 OnOffRatio = 12.5  
 WriteCurr = 64e-9  
 InhibCurr = 32e-9  
 FullTimeLTP = 300e-6 \* 100  
 FUllTimeLTD = 300e-6 \* 100  
 OnePulseLTP = 300e-6  
 OnePulseLTD = 300e-6  
 LTP = 2.4 # LTP Nonlinearity (if linear, write 0)  
 LTD = 4.9 # LTD Nonlinearity (if linear, write 0)  
 Vari = 0.6 # Device variation  
 LEVEL = 100  
 ReadTimePerCycle = 5e-9 # Required Readtime per cycle  
 VRead = 2  
 UpdateLEVELs = min(2 \* math.floor(UpdateMax \* min(FullTimeLTP, FUllTimeLTD) / (2 \* Cycle)),  
 2 \*\* (22 - math.floor(math.log2(1 / UpdateMax))))  
  
if Memory == 'ReRAM2': # From "A methodology to improve linearity of analog RRAM for neuromorphic computing"  
 HighVoltage = 1.6 # The voltage used for a level shifter for Memory Write Operations  
 MemHeight = 2 # Not Known  
 MemWidth = 4 # Not Known  
 ONCurrent = 40e-9  
 OnOffRatio = 10  
 WriteCurr = 16e-6  
 InhibCurr = 8e-6  
 FullTimeLTP = 50e-9 \* 128  
 FUllTimeLTD = 50e-9 \* 128  
 OnePulseLTP = 50e-9  
 OnePulseLTD = 50e-9  
 LTP = 0 # LTP Nonlinearity (if linear, write 0)  
 LTD = 0.63 # LTD Nonlinearity (if linear, write 0)  
 Vari = 0.6 # Device variation  
 LEVEL = 128  
 ReadTimePerCycle = 5e-9 # Required Readtime per cycle  
 VRead = 0.8  
 UpdateLEVELs = min(2 \* math.floor(UpdateMax \* min(FullTimeLTP, FUllTimeLTD) / (2 \* Cycle)),  
 2 \*\* (22 - math.floor(math.log2(1 / UpdateMax))))  
  
  
if Memory == 'IDEAL': # From "Nanoscale Memristor Device as Synapse in Neuromorphic Systems"  
 HighVoltage = 1 # The voltage used for a level shifter for Memory Write Operations  
 MemHeight = 2 # Not Known  
 MemWidth = 2 # Not Known  
 ONCurrent = 10e-9  
 OnOffRatio = 10000  
 WriteCurr = 100e-15  
 InhibCurr = 0  
 FullTimeLTP = 50e-6 \* 100  
 FUllTimeLTD = 50e-6 \* 100  
 OnePulseLTP = 50e-6  
 OnePulseLTD = 50e-6  
 LTP = 0 # LTP Nonlinearity (if linear, write 0)  
 LTD = 0 # LTD Nonlinearity (if linear, write 0)  
 Vari = 0.6 # Device variation  
 LEVEL = 1000  
 ReadTimePerCycle = 5e-9 # Required Readtime per cycle  
 VRead = 0.2  
 UpdateLEVELs = min(2 \* math.floor(UpdateMax \* min(FullTimeLTP, FUllTimeLTD) / (2 \* Cycle)),  
 2 \*\* (22 - math.floor(math.log2(1 / UpdateMax))))  
  
if UpdateLEVELs < 2:  
 print('Too low Update Resolution!!')  
 #sys.exit()  
  
UPDEL = 0  
for i in range(InputPrecesion):  
 UPDEL += math.ceil(math.log2(UpdateLEVELs)) - InputPrecesion + i  
  
  
# Design Constraints  
Tech = 45 # Technology Node Chosen (There is 45 nm now (32, 28, 22 are also supported))  
if Tech == 45:  
 VDD = 1  
 EFFR = 1.05 # Effective Resistance multiplier we use  
 CONSTEFFR = 1.54 # Effective Resistance multiplier S.Yu use  
 WireWidth = 50  
 UnitWiRes = 4e-8 / (WireWidth \* WireWidth \* 1e-18 \* 1.9)  
 NONCurr = 45 \* 1e-9 \* 1.27e3 # NMOS On Current with mimimum feature size  
 PONCurr = 45 \* 1e-9 \* 1.08e3 # PMOS On Current with mimimum feature size  
 NONCurrS = 45 \* 1e-9 \* 1.15e3 # NMOS On Current with mimimum feature size (SPICE)  
 PONCurrS = 45 \* 1e-9 \* 0.70e3 # PMOS On Current with mimimum feature size (SPICE)  
 JuncCapP = 0.05e-15  
 JuncCapN = 0.05e-15  
 GateCapP = 0.04e-15  
 GateCapN = 0.04e-15  
 PNRATIO = 2 # Ratio Between PMOS and NMOS  
 RA = 0.18 # Unit Small wire resistance  
 MIGRATIONLIM = WireWidth \* WireWidth \* 1.5e-8  
  
PNGAP = 3 # GAP Between PMOS Diffusion Area and NMOS Diffusion Area  
GATEGAP = 3 # GAP Between Field GATEs  
GATEOVERLAP = 1 # Gate Length which is overlapped(extended) according to the Design RuleFu  
CELLHeight = 28 # Height of Standard Cell of Mask Layout  
GAP\_BET\_GATE\_POLY = 3 # Gate poly GAP for Width Calculation  
BUSFOLD = 4 # Folded Bus Ratio (4)  
MetalPitch = 3 # Metal Pitch  
  
# Physical Characteristics  
UnitWiCap = 1e-17/1e-6 # 0.01fF/um

\*\*\*\*\*\* Chip Specifications\*\*\*\*\*\*\*

The CIM options can be modified by altering the paramters in ‘Parameter.py’ file. The paramters in the file is explained as follows:

**GlobalBusWidth**: Global Bus width of the chip. Typically, the global buswidth of the industrial accelerators, such as GPU is given as 4096 or 8192.

**NumTilePerChip**: Number of tiles per CIM chip.

**NumPEperTile**: Number of processing elements per tile

**NumSubperPE**: Number of Subarrays per processing element.

**SubRow**: Row Size of a Subarray. (Larger or same size with the Output channel size of the largest convolutional layer is recommended.)

**SubCol**: Column Size of a Subarray. (Larger or same size with the Output channel size of the largest convolutional layer is recommended.)

**PEBuffer**: The type of Buffer used in the processing element. (Use ‘SRAM’ as buffer now. In some CIMs the register type using ‘DFF’ can be used as Buffer (Future work. Or user might add their own register by adding the features of their own register specifications.).)

**TileBuffer**: The type of Buffer used in the tile in CIM chip. (Use ‘SRAM’ as buffer now. In some CIMs the register type using ‘DFF’ can be used as Buffer (Future work. Or user might add their own register by adding the features of their own register specifications.).)

**RowsPerADC**: Number of shared rows by ADC. Typically, over 8 rows are shared by ADC since ADC requres large area overhead. (Rows per ADC parameter is required for the backpropagation prcocess.)

**ColumnsPerADC**: Number of shared columns by ADC. Typically, over 8 columns are shared by ADC since ADC requres large area overhead. (Columns per ADC parameter is required for the feed forward prcocess.)

**ColumnsPerWeight**: Number of columns for a single weight in the network. (Currently, we only support 1. However, the two synapse devices for positive (G+) and negative (G-) weight representation are used. 1 means that a single synapse is needed for each positve and negative weight representation.)

**SRAMRow**: The row size of SRAM buffer.

**SRAMRowF**: The veritcal feature size normalized by the minimum feature size (determined by the technology node) of a single SRAM cell.

**SRAMCol**: The column size of SRAM buffer.

**SRAMColF**: The horizontal feature size normalized by the minimum feature size (determined by the technology node) of a single SRAM cell.

**BusToTileT**: The type of Interconnection from global chip to Tile. (‘HTree’ means H-Tree interconnection method using the repeaters. ‘Bus’ means typical bus interconnection.)

**BusToPET**: The type of Interconnection from Tile to Processing element. (‘HTree’ means H-Tree interconnection method using the repeaters. ‘Bus’ means typical bus interconnection.)

**BusToSubT**: The type of Interconnection from Processing element to SubArray. (‘HTree’ means H-Tree interconnection method using the repeaters. ‘Bus’ means typical bus interconnection.)

**PEAccParr**: The Number of copied AdderTree in PE to operate accumulation parallely. (1 means only 1 adder tree exists, which will take the accumulation time much longer.)

**TileAccParr**: The Number of copied AdderTree in Tile to operate accumulation parallely. (1 means only 1 adder tree exists, which will take the accumulation time much longer.)

**GlobalAccParr**: The Number of copied AdderTree in Global CIM chip to operate accumulation parallely. (1 means only 1 adder tree exists, which will take the accumulation time much longer.)

\*\*\*\*\*\* Elements determined by Networks\*\*\*\*\*\*\*

**IMGSIZE**: Image size provided by dataset. For example, CIFAR-10 provides 32 by 32 image for each R, G, and B value. In that case, IMGSIZE is 32.

**MaxLayerInput**: MaximumLayer input size in the terms of memory size to store. In VGG-9, we use, 32 \* 32 \* 64 output image is the largest to store the results.

**BATCH**: Batch size of the network. For the DRAM-free CIM, BATCH should be 1. If BATCH is large, the data communications between the external DRAM becomes necessary as the CIM cannot store all the data in the cache of buffers.

**LearnRate**: Learning rate of the network.

**UpdateMax**: The maximum increase or decrease in conductance using POPU per single training iteration.

**EPOCH**: The epoch for on-chip training in CIM chips.

**NUMCLASS**: The number of answer classes. (CIFAR-10: 10, CIFAR-100: 100)

\*\*\*\*\*\*Network Properties\*\*\*\*\*\*

**DATANUM**: The data required for the entire Training. (User should set their own number. Maximum DATANUM per EPOCH is 50000 in CIFAR-10 dataset.)

**NumLayer**: Number of Layers in the network. (VGG-9 in here has 9 layer.)

**Kernel**: The size of convolutional filter used in the network.

**LayerIMG**: The size of image getting through each layer. In Fully connected layers, just write in the input size of the fully connected layer. The last element should be NUMCLASS.

**LayerInChan**: The size of the input channel of the convolutional layers. For the fully connected layers, just write in the input size of the fully connected layer. The last element should be NUMCLASS.

**LayerType**: Write in the Type of each layer. For the convolutional layers, write ‘Conv’ and for the fully connected layers write ‘FC’. The last element should be ‘END’.

**IsPoolLayer**: Write in 1 if the layer is followed by the pooling operation. Else, write in 0.

**SpeedUP**: Only used for the comparison with the NeuroSim. NeuroTorch do not use this parameter generally.

\*\*\*\*\*\*Parameters related to the quantization\*\*\*\*\*\*

**Mode**: CIM Mode (‘Inference\_Normal’: Mode of Inference in CIM system. ‘Onchip\_Parallel’: On-chip training using POPU method.)

**Onchip\_ParMode**: In the on-chip training scheme using POPU, the characteristics of synapse device update method can be chosen among (‘LTPLTD\_both’: Use LTP and LTD both, ‘LTP\_only’: Use LTP only, and ‘LTD\_only’: use LTD only)

**MUXSize**: The size of Transmission Gate used for MUX. If MUXSize = 1, the minimum feature size of Transmission gate is used.

**ADCType**: Type of ADC. Now, only ‘SAR’ ADC is available. Later some other types of ADCs will be provided.

**ADCPrecision**: The resolution bit of the ADC.

**ADCMax**: The maximum range of ADC. ADC quantizes the MAC results from 0 to ADCMax

**DelEn**: How many bits per delta cycle

**DelPrecision**: How many bit resolutions per delta value for the backpropagation.

**DelADCMax**: The maximum range of ADC for the backpropagation. ADC quantizes the backpropagation MAC results from 0 to DelADCMax.

**ADCCycle, Cycle**: The Clock Cycle of the ADCs. In NeuroTorch ADCCycle also means the minimum clock cycle of the hardware system.

**InputEncoding**: How many bits per delta cycle (Feedforward process)

**InputPrecesion**: How many bit resolutions per input value for the feedforward process.

\*\*\*\*\*\*Memory Device Specifications\*\*\*\*\*\*

**Memory**: Type of nonvolatile memory

**HighVoltage**: Write Voltage of the nonvolatile memory device.

**MemHeight**: The cell height of single nonvolatile memory cell represented by process minimum feature size (F).

**MemWidth**: The cell width of single nonvolatile memory cell represented by process minimum feature size (F).

**OnCurrent**: The maximum on-current of the nonvolatile memory cell. (unit: A)

**OnOffRatio**: The current ratio of nonvolatile memory cell between on-current and off-current.

**WriteCurr**: Current flowing through the nonvolatile memory cell when the write operation is performed.

**InhibCurr**: Current flowing through the nonvolatile memory cell when the inhibition operation is performed.

**FullTimeLTP**: Full Time of the LTP operation. It is calculated by the total number of pulses for LTP times duration of each pulse for LTP.

**FullTimeLTD**: Full Time of the LTD operation. It is calculated by the total number of pulses for LTD times duration of each pulse for LTD.

**OnePulseLTP**: Duration of each pulse for LTP.

**OnePulseLTD**: Duration of each pulse for LTD.

**LTP**: The nonlinearity factor of LTP

**LTD**: The nonlinearity factor of LTD

**Vari**: Device-to-device variation of the nonvolatile memory cells.

**LEVEL**: The available weight quantization using nonvolatile memory cell.

**ReadTimePerCycle**: The required readtime for each read operation.

**VRead**: Read voltage used for the MAC opeartion.

**UpdateLEVELs**: The available update levels for POPU. The conductance change amount smaller than the available with the minimum clock cycle cannot be supported. UpdateLEVELs represent the resolution of conductance change available by the minimum clock cycle.

\*\*\*\*\*\* Design Constraints (Technology Node Dependent) \*\*\*\*\*\*

**Tech**: Technology node of the hardware (if Tech = 45, the technology node is using 45 nm process)

**EFFR**: Effective resistance constant for the latency calibration. (Optimized through SPICE simulation)

**CONSTEFFR**: Effective resistance constant for the latency calibration. (This is used in NeuroSim and CACTI)

**WireWidth**: Metal Pitch of the technology node. In 45 nm technology, metal 1 layer pitch is 50 nm.

**UnitWiRes**: Unit Metal Layer Resistance.

**NONCurr**: NMOS On-Current with minimum feature size, provided by NeuroSim

**PONCurr**: PMOS On-Current with minimum feature size, provided by NeuroSim

**NONCurrS**: NMOS On-Current with minimum feature size, provided by SPICE simulation

**PONCurrS**: PMOS On-Current with minimum feature size, provided by SPICE simulation

**JuncCapP**: PMOS unit junction capacitance.

**JuncCapN**: NMOS unit junction capacitance

**GateCapP**: PMOS unit gate capacitance

**GateCapN**: NMOS unit gate capacitance

**PNRATIO**: PMOS / NMOS width Ratio for mobility calibration.

**RA**: Unit Wire Resistance used for nonvolatile memory array

**MIGRATIONLIM**: Electromigration current limit calculated with the current density limit and metal sheet area.

**PNGAP**: GAP between PMOS diffusion area and NMOS Diffusion area

**GATEGAP**: GAP between Field Gates

**GATEOVERLAP**: Gate length which is overlapped accroding to the design rule

**CELLHeight**: Height of the standard cell of mask layout

**GAP\_BET\_GATE\_POLY**: Gate poly gap for width calculation

**BUSFOLD**: Folded bus ratio

**MetalPitch**: Metal gap.

**UnitWiCap**: Unit wire parasitic capacitance.

1. **CIM Architecture, Schemes**

NeuroTorch assumes similar hierarchical CIM architectures to NeuroSim, comprising Tiles, Processing Elements, and Macros. Interconnects, adders, and buffers are included in each element as NeuroSim does. The main difference of NeuroTorch is that NeuroTorch assumes that the external DRAM of internal weight gradient computation (WGC) units are not essential. By applying parallel outer product update schemes, the operation of the WGC units is replaced and using the batch size 1, avoids the need for the external DRAM. However, the newly proposed parallel outer product update induces other nonidealties, such as instantaneous current during write operation and also suffers accuracy degradation caused by the nonlinear weight update. NeuroTorch considers the hardware specifications to implement POPU in the CIM architectures. Additionally, since ADC quantization in the on-chip training leads to the massive amount of simulation time (~2 weeks) the ADC quantization effect is not considered in the on-chip training in NeuroSim. In NeuroTorch, by approximating the ADC quantization operation, users can explore CIM design for on-chip training considering the approximated ADC quantization.

1. **Calculation Model**

**Area**: NeuroTorch adopts standard cell design for area estimation. Since the area of the mask layout is same as the real chip area, calculating the area of the standard cell mask layout means calculating the chip area of the CIM. Mask layout area of the standard cell can be calculated by multiplying the height and the width of the standard cell. The height of the standard cell is determined by the design industry and it is usually fixed value. Typically, the height is fixed by 28 F, which is depend on the process technology node. The width of the standard cell is more complicated. Usually, the width of each standard cell is determined by the multiplication of max (Device number of PMOS, Device number of NMOS) and size of each MOSFET (channel length-wise). In some cases, single MOSFET size varies, since, if the width of the MOSFET is too big to fit in the provided standard cell height, the MOSFET is folded and redesigned. By calculating the folded number of each MOSFET and the maximum width of the standard cell, NeuroTorch gets the width of the standard cell.

**Latency**: NeuroTorch uses Horowitz equation, elmore delay, and Pi-Model for the latency estimation for CIM chips. Horowitz equation is for the latency estimation in serial digital circuits considering the rise/fall time of the voltages and the equation fits comparable with the SPICE simulation when the capacitance and the resistance of the transistors are well defined. Elmore delay is used for the cases, when the series of the resistances and capacitances are connected and the latency in the certain node is required. Pi-Model is used for the simplification of serial resistances and capacitances. In the nonvolatile memory array, the parasitic resistances and capacitances are connected to the nonvolatile memories. By using Pi-Model, hundreds of the parasitic resistances and capacitances are modeled into a few resistance and capacitances. By combining, Elmore delay and Pi-Model, NeuroTorch can calculate the latency accurately.

**Energy**: For energy consumption, NeuroTorch uses the CV2 for the calculation. as other circuits simulators calculate. By calculating the capacitance of each element, which gets through the charging and discharging operations, and multiplying the power of the supply voltage, NeuroTorch gets the energy consumption. In addition to the method, NeuroTorch considers the activity factor. Since not all the nodes in the digital circuits are switched in each operation, the probability rate of the switching in the provided node should be multiplied. The probability rate is multiplied while the energy consumption is caculated, for accurate results, in NeuroTorch.

**Accuracy**: For the accuracy estimation in CIM systems, NeuroTorch considers nonvolatile memory nonildealities, such as conductance update nonlinearity, device-to-device variations, conductance update resolution, and on-off ratio as NeuroSim does. In addition to the nonvolatile memory nonidealities, NeuroTorch approximated the ADC quantization nonideality in CIM. On-chip training in NeuroSim does not consider the ADC nonideality as the simulation time matters. The simulation time problem is caused by the enrolling the weights into the nonvolatile memory array and considering the ADC quantization, array by array. NeuroTorch approximated the ADC quantization operation, by integrating the analog sums from each array, then quantizing them. Since the effect of the on-off ratio of nonvolatile memory can be overrated, the on-off ratio of nonvolatile memory is normalized accroding to the number of arrays. Also, the ADC quantization resolution is normilized with the number of the arrays. This is not precise method for considering ADC quantization method but this is a method that can obtain results that consider ADC quantization, to some extent, within a reasonable simulation time.