Documentation of Crossbar Process Engine Simulator

1. Release Notes

XPEsim v0.1.0 (Nov, 2017)

This is a first release of Crossbar Process Engine Simulator (XPEsim), version 0.1.0. *RPE* is a deep learning accelerator by using Resistive Random-Access Memory (RRAM) as weights. *XPEsim* is a toolchain for RPE including training framework, mapping tool and hardware evaluation tool.

XPEsim features:

- Training framework with noise weights – Use Theano as a backend to realize GPU acceleration.
- Mapping the optimized weights to hardware loadable – *XPEsim* can map the weights of full-connect and convolutional layers to the conductances of RRAM arrays in the synaptic cores.
- Traced hardware evaluation of hardware - Generate the performance of the specific XPE configuration, including area, latency, power and computing performance.

2. Download and Installation

The latest development version of RPEsim have been uploaded to gitlab, available via:
https://git.cbicr.org/zhangwq/simulator.git (To move to Github)
This package includes the simulator itself (in simulator directory) and relative examples (in sim_example directory)

Installation requirement (Linux):

Python == 2.7 for training framework and Python == 2.7/3.5+ for mapping tools
Theano >=0.9
numpy >=1.11

Installation through pip (Recommended for Ubuntu 16.04)

Linux:

After installation of Theano (GPU version is required), *XPEsim* can be installed by the following command:

Step1: Download the *XPEsim* from GitHub

Step2: Change to simulator directory and install the package by:

```
pip setup.py install
```

3. Overview

XPEsim is developed in Python/C++ to model the system-level Crossbar Process Engine (XPE) for neuromorphic computing. It gives an extendible simulator architecture to train the given deep learning algorithm, map the optimized weights to *XPE* hardware and evaluate the hardware performance. During the simulation, all circuits in *XPEsim* are modelled using a numerical model.

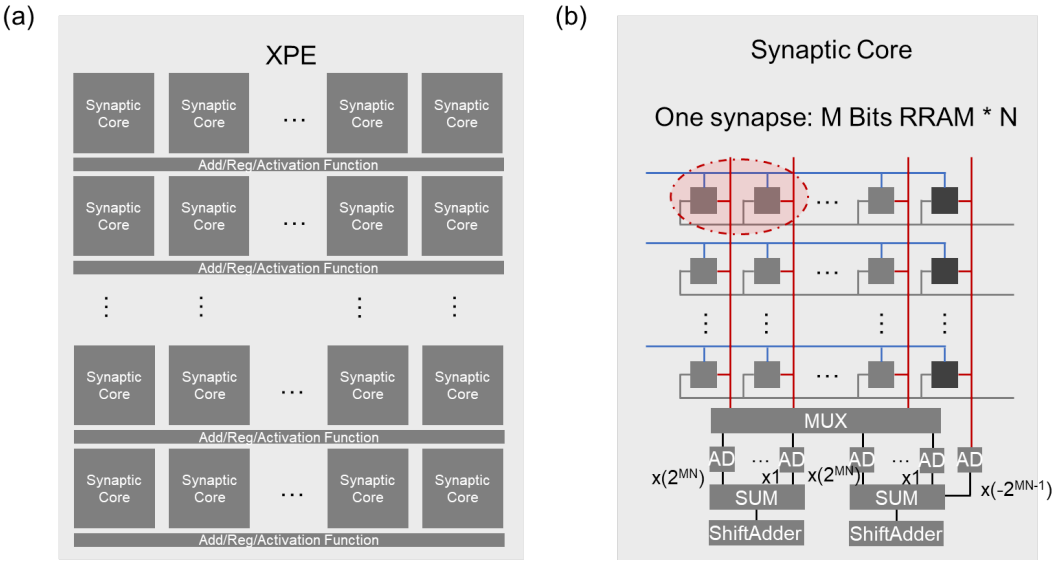

Figure 3.1 (a) Block diagram of XPE. (b) Block diagram of Synaptic Core.

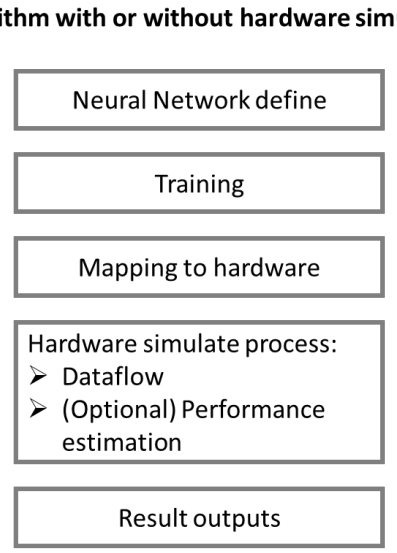

Figure 3.2 Work flow of *XPEsim*

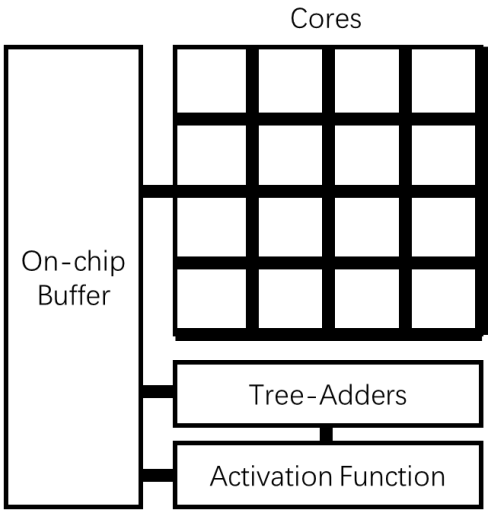

Figure 3.3 Architecture of *XPE*

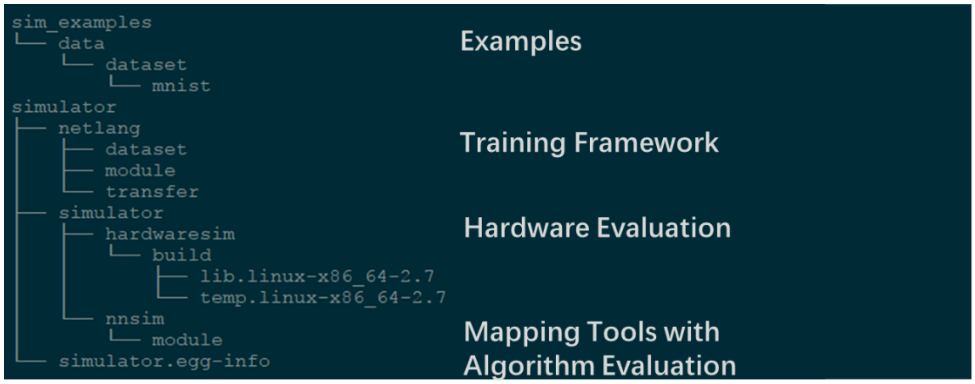

Figure 3.4 Code structure of *XPEsim*

4. Mapping tools

A simply mapping tool is implemented in *XPEsim*. As shown in Fig. 4.1, to fit into the RRAM arrays, the 784*500 weights are padded to 896*512 firstly (if the array size is 64*64, the used array number is 14*8). Then the 896*512 weights are mapped to the conductance of RRAM. The general mapping relation is given by:

$$G_{traget} = n \cdot G_{step} + G_{min}.$$

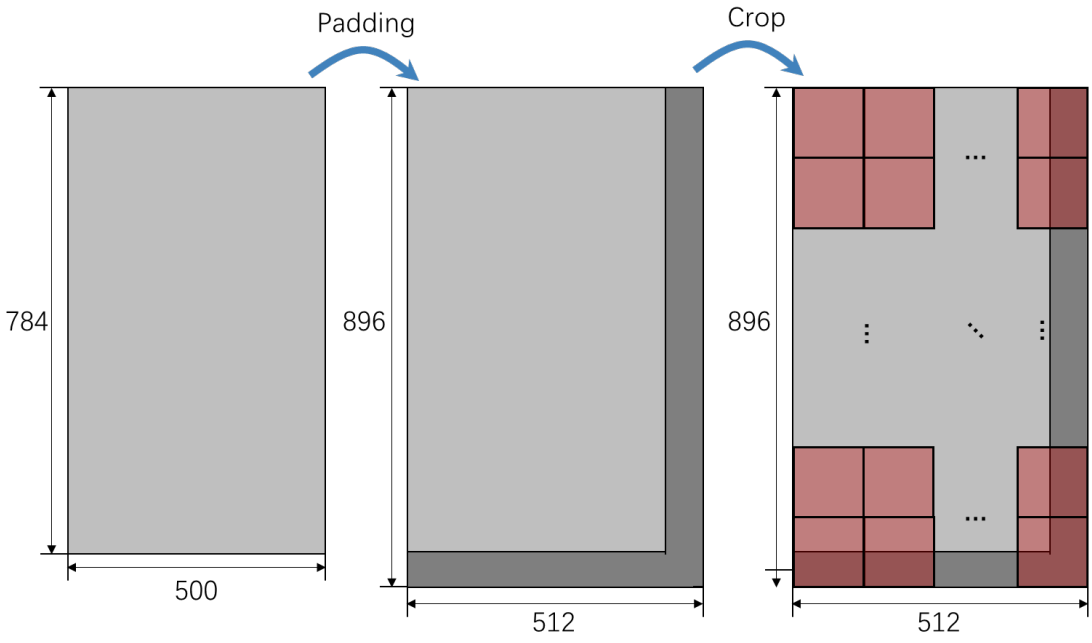

Figure 4.1 Mapping method implemented in *XPEsim*

5. Hardware evaluation

5.1 Device Characteristics

The device model in the simulator is a compact model, which is fitted by the measure data in 1K-bit TiN/TEL/HfOx/TiN RRAM array. A multi-level compact model can be given by fitting the relation between the mean and SD:

$$SD = -6.0 \times 10^{-4} \times Mean^2 + 6.2 \times 10^{-2} \times Mean + 7.2 \times 10^{-1}.$$

Figure 5.2 shows the experimental measurement results and relative simulation results.

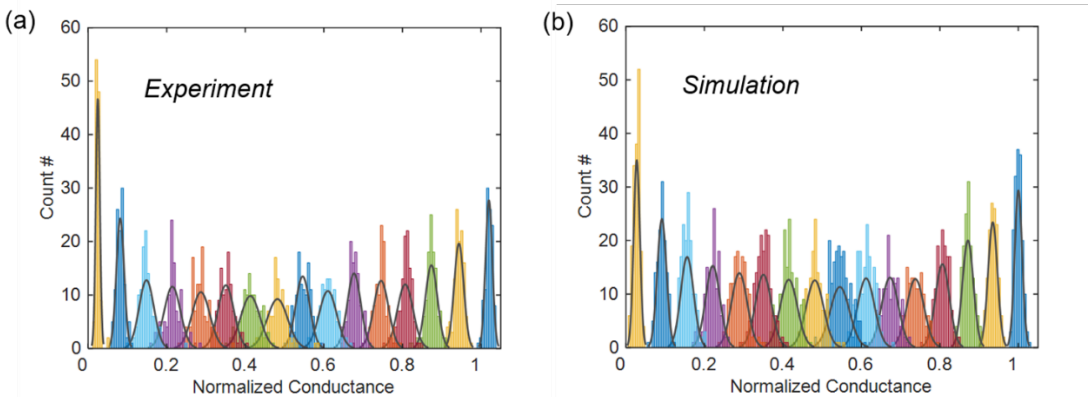

Figure 5.2. (a) The measured 16-level distributions in 1-Kbit array; (b) The simulated distributions of the proposed digital compact model. The conductance is normalized by $1/G_{level=max}$.

5.2 Synaptic Core

A synaptic core is specifically designed for weighted sum and weight update. It takes the digital input vector and gives out the weighted sum result. The synaptic core consists of the synaptic array and array periphery. The synaptic array is the core unit of weighted sum computation and the array periphery helps transform the results to be the digital format if necessary. The synaptic core supports analog eNVM based synaptic cores, in both 1T1R and crossbar structures, as shown in Fig. 5.3(a)-(b).

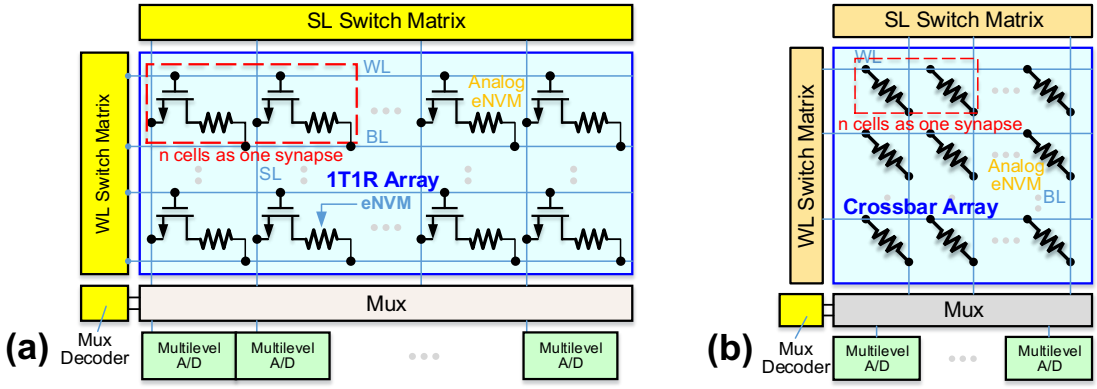

Figure 5.3. (a) Circuit block diagram of analog eNVM synaptic core with the 1T1R array. (b) Circuit block diagram of analog eNVM synaptic core with the crossbar array.

As shown in Fig. 1, there are several peripheral blocks in the synaptic core:

- WL Switch Matrix: a group of transmission gate pairs. During weighted sum operation, the given input vector that saved in registers (not shown here) decide the WLs/BLs to be connected to either a voltage source or ground.
- SL Switch Matrix: a group of transmission gate pairs. During weight update operation, a vector of control signals from the registers (not shown here) decide the SLs to be connected to either a write source or ground.
- Mux Decoder: a decoder that gives out control signals to mux, the number of bits of mux decoder is related to the number of columns in synaptic array that share one multi-level sense amplifier (to improve area efficiency), the relationship is $B_{muxDecoder} = \log_2 N$, where $B_{muxDecoder}$ is the number of bits of mux decoder,

N is the number of columns that share one multi-level sense amplifier.

- Mux: a group of transmission gates controlled by mux decoder. If N is the number of columns that share one multi-level sense amplifier, during weighted sum operation, there will be N times of sensing operation in total, at each time, a group of SLs will be connected to multi-level sense amplifier.
- Multi-level sense amplifier: formed by multiple stacking sense amplifiers with different reference, give out multi-bit digital results.

The 1T1R and crossbar array structures can achieve a high integration density. As shown in Fig. 6.3, if the input vector is encoded by read voltage signals, which will be passed through WL switch matrix, the weighted sum operation (matrix-vector multiplication) can be performed in a parallel fashion with synaptic 1T1R or crossbar array. The weighted sum result in terms of the output currents are then obtained at the end of each column, and the multi-level sense amplifiers (Multilevel A/D as shown in Fig. 6.3) at the end of each column will transform the weighted sum results into digital format.

The synaptic core can load in input vector and weight matrix data, one should be noted that the input vector should be stored in column vertically, while multiple input vectors can be stored in one single file, also, both input vector file and weight matrix file should be stored as pure data, no extra text can be included. During weighted sum operation, the input vector will define the read voltage signals that applied to each WL, if there are multiple input vectors, the synaptic core will simulate multiple times depending on different input vectors and give out multiple simulation results.

Also, the weight matrix data will be mapped into synaptic array, while the eNVM device can represent different weight values. It is important to notice that this simulator can represent height precision weight, by either form a synapse cell within multiple digital eNVM devices or form a synapse cell within multiple analog eNVM devices (e.g. a 5-bit synapse can be formed by two 3-bit analog eNVM devices, while one device represents the MSB, the other one represents the LSB).

The users can also modify many other important parameters such as:

- arrayRowSize/arrayColSize: synaptic array row/column size;
- synapsebit: number of bits of each synaptic cell;
- cellbit: number of bits of each eNVM device;
- Ron&Roff: on resistance and off resistance of eNVM device;
- readNoise: sigma of read noise in Gaussian distribution;
- technology node, wire width and temperature;
- read voltage and write voltage, read pulse width and write pulse width, etc.

To evaluate the performance of synaptic core accurately, one should consider about the sensing limitation of multi-level sense amplifier, that is, the multi-level sense amplifier can only sense a range of equivalent current to guarantee the sensing accuracy. In this simulator, assuming read voltage is not larger than 2 V, the range of equivalent resistance that the multi-level sense amplifier can support is from 100 Ω to 10M Ω.

After setting all the parameters, the synaptic core will provide typical circuit-level performance metrics include the area, latency, dynamic energy and leakage power. Compared to the time-consuming SPICE simulation, this estimation of the performance metrics using analytical models or pre-defined values provided by the user with reasonable

accuracy is quite efficient.

6. Examples

The examples can be found in ./sim_example directory, including FCNN and CNN.

7. Limitation release and future work

- Router of the XPE