Simple Arithmetic Function Implementation on IBM Truenorth SNN

Zhao Zhixu

November 1, 2016

Contents

1 Introduction

This article is based on *Cognitive Computing Building Block: A Versatile and Efficient Digital Neuron Model for Neurosynaptic Cores* by IBM Research. The neuron model used here is a variant leaky integrate-and-fire neural model with a constant leak, known as the Truenorth neuron model. For the details of neuron specification of Truenorth neuron model, it is recommended to refer to the paper mentioned above, especially section II.

This article introduces some methods to implement arithmetic functions mentioned in section III (table IV and Fig. 8) of the IBM paper, including

addition, fixed-gain div./mult., integer muliplication(not in the paper), and subtraction.

In order to facilitate debugging neuron parameters, I wrote a simulator in MATLAB in which you can easily construct networks, set parameters of neurons and axons, feed the axons with linearly varying rates, and watch the input and output in a graph. Here is the project page: https://github.com/zhaozhixu/TruenorthSNN-Simulator. Detailed manual can be found in README.

For the convenience of description, I will list the symbol table, function definations and neuron specification equations in the next few sections.

Table 1: Summary of Symbols

Variables and Parameters	Symbol	Format
membrane potential	$V_j(t)$	signed int
local timestep	t	unsigned int
input spikes on i^{th} axon	$A_i(t)$	$\{0,1\}$
synaptic PRN	$\rho_{i,j}$	unsigned int
leak PRN	ρ_j^{λ}	unsigned int
threshold PRN (drawn)	ρ_j^{T}	unsigned int
threshold PRN (masked)	η_j	unsigned int
leak direction variable	Ω	$\{-1,0,+1\}$
synapse (i^{th} axon, j^{th} neuron)	$w_{i,j}$	$\{0,1\}$
type of i^{th} axon	G_i	$\{0,1,2,3\}$
synaptic weight/probability	$s_j^{G_i}$	signed int
synaptic weight/probability select	$b_j^{G_i}$	$\{0,1\}$
leck-reversal flag	ϵ_j	$\{0,1\}$
leak weight/probability	λ_j	signed int
leak weight/probability select	c_j^{λ}	$\{0,1\}$
positive $V_j(t)$ threshold	α_j	unsigned int
negative $V_j(t)$ threshold/floor	β_j	unsigned int
threshold PRN masked	M_j	unsigned int
reset voltage	R_j	signed int
negative thresh: reset or saturate	κ_j	$\{0,1\}$
$V_j(t)$ reset mode	γ_j	$\{0,1,2\}$
PRNG initial seed value	ρ_j^{seed}	unsigned int

1.1 Summary of Symbols

In Table 1 are the variables and parameters used in neuron networks. Signed integers have 9 bits, and unsigned integers have 8 bits.

1.2 Function Defination

Functions used in neuron networks are as follows. Sign function:

$$\operatorname{sgn}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases} \tag{1}$$

Binary comparision operation:

$$\mathrm{F}(s, \rho) = \begin{cases} 1 & \text{if } |s| \geq \rho, \\ 0 & \text{else} \end{cases} \tag{2}$$

Binary bitwise AND operation: &
Kronecker delta function:

$$\delta(x) = \begin{cases} 1 & \text{if } x = 0, \\ 0 & \text{else} \end{cases} \tag{3}$$

1.3 Neuron Specification Equations

The leaky integrate-and-fire neuron model includes 3 stages, they are:

1. Synaptic Integration

$$V_j(t) = V_j(t-1) + \sum_{i=0}^{255} A_i(t) w_{i,j} [(1 - b_j^{G_i}) s_j^{G_i} + b_j^{G_i} \, \mathrm{F}(s_j^{G_i}, \rho_{i,j}) \operatorname{sgn}(s_j^{G_i})] \tag{4}$$

2. Leak Integration

$$\Omega = (1 - \epsilon_j) + \epsilon_j \operatorname{sgn}(V_j(t)) \tag{5}$$

$$V_j(t) = V_j(t) + \Omega[(1 - c_j^{\lambda})\lambda_j + c_j^{\lambda} \, \mathrm{F}(\lambda_j, \rho_j^{\lambda}) \operatorname{sgn}(\lambda_j)] \tag{6}$$

3. Threshold, Fire, Reset

$$\eta = \rho_j^T \& M_j \tag{7}$$

$$\text{if} \quad V_j(t) \geq \alpha_j + \eta_j \tag{8}$$

Spike

$$V_j(t) = \delta(\gamma_j) R_j + \delta(\gamma_j - 1)(V_j(t) - (\alpha_j + \eta_j)) + \delta(\gamma_j - 2) V_j(t) \tag{9}$$

$$\text{elseif} \quad V_j(t) < -[\beta_j \kappa_j + (\beta_j + \eta_j)(1 - \kappa_j)] \tag{10}$$

$$V_j(t) = -\beta_j \kappa_j + [-\delta(\gamma_j) R_j + \delta(\gamma_j - 1)(V_j(t) + (\beta_j + \eta_j)) + \delta(\gamma_j - 2) V_j(t)](1 - \kappa_j) \tag{11}$$

endif

Table 2: Function configuration

Name	j	$w_{i,j}$	$s_j^{G_i}$	$b_j^{G_i}$	ϵ_j	λ_j	c_j	α_j	β_j	M_j	R_j	κ_j	γ_j
Addition	0	$[1,1]$	$[1,1,0,0]$	$[0,0]$	0	0	0	1	1	0	0	0	1
Fixed-gain div./mult.	0	1	$[1,0,0,0]$	0	0	-128	1	1	1	0	0	0	0
Integer multiplication	0	1	$[2,0,0,0]$	0	0	0	0	1	1	0	0	0	1
Subtraction	0	$[1,1]$	$[1,-1,0,0]$	$[0,0]$	0	0	0	1	1	0	0	1	1

2 Arithmetic Function Implementation

In Table 2 are the variables and parameters used to configure the SNN to perform specific arithmetic functions. i is the index of neurons. j is the index of axons.

2.1 Addition

Addition needs 1 neuron connected with 2 axons. Each axon has a weight of 1. No leak. Positive threshold is 1. Reset mode is linear mode. So when 2 axons both input a spike, the membrane potential will be 2, and subsequently reset to 1. That means every single spike from two axons will cause an output spike. Therefore the output rate will be the sum of the input rates.

2.2 Fixed-gain Division/Multiplication

When a queue of spikes of some rate incomes, this operation outputs a queue of spikes of the input rate timing the fixed-gain. The gain is a number between 0 and 1.

Fixed-gain div./mult. needs 1 neuron connected with 1 axon. The axon provides input spikes, while the gain is provided indirectly by λ_j. Parameter c_j is set to 1, which means there is a probability whether 1 is to be subtracted from membrane potential. The probability can be calculated by the formula:

$$P(\text{subtract 1 from V(t)}) = \frac{\mid \lambda_j \mid}{\rho^{\lambda}_{j\,\max}} \tag{12}$$

Because the max ρ is the max value that a 8-bit binary number can stand for (255), in our demo case $\lambda = -128$ will make the probalility equals 0.5. When the spikes of some rate incoming, they can be "neutralized" by leak integration in a 50-50 chance. So the gain is 0.5 in our case. λ_j can be set to any meaningful number to generate the gain.

2.3 Integer Multiplication

This operation also performs multiplication, but with a fixed integer gain. The mechanism is the same with addition operation, but with only one axon input and synapse weight set to 2 (in our demo case). So the gain will be 2.

2.4 Subtraction

This operation inherits the methods of addition, but with two differernces. First, the weights of 2 synapses are 1 and -1. Second, κ_j parameter is set to 1, which means we choose saturate negative thresh: when membrane potential crosses negative thresh, it stays on that thresh. So the negative weights integrated by the second axons can be preserved to "neutralize" following positive weights.

3 Conclusion

This article provides some methods to implement simple arithmetic functions on Truenorth spike neuron network. Although they are far from complete, they can still give some inspiration on following work. Welcom to download my simulator code on github and configure SNN on it. I will be glad to

recode it in C to inhance its performance, if needed. Any bug reports or pull requests are welcome.

Contacts: Zhao Zhixu

- lion_zzx@stu.xjtu.edu.cn
- https://github.com/zhaozhixu/TruenorthSNN-Simulator