

# Simple Arithmetic Function Implementation on IBM Truenorth SNN

Zhao Zhixu

November 1, 2016

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Summary of Symbols . . . . .	3
1.2	Function Definations . . . . .	3
1.3	Neuron Specification Equations . . . . .	3
<b>2</b>	<b>Arithmetic Function Implementation</b>	<b>4</b>
2.1	Addition . . . . .	4
2.2	Fixed-gain Division/Multiplication . . . . .	5
2.3	Integer Multiplication . . . . .	5
2.4	Subtraction . . . . .	5
<b>3</b>	<b>Conclusion</b>	<b>5</b>

## 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 multiplication(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/favoryoung/TruenorthSNN-Simulator>. Detailed manual can be found in README.

For the convenience of description, I will list the symbol table, function definitions 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	$\rho_j^\lambda$	unsigned int
threshold PRN (drawn)	$\rho_j^T$	unsigned int
threshold PRN (masked)	$\eta_j$	unsigned int
leak direction variable	$\Omega$	$\{-1, 0, +1\}$
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	$b_j^{G_i}$	$\{0, 1\}$
leak-reversal flag	$\epsilon_j$	$\{0, 1\}$
leak weight/probability	$\lambda_j$	signed int
leak weight/probability select	$c_j^\lambda$	$\{0, 1\}$
positive $V_j(t)$ threshold	$\alpha_j$	unsigned int
negative $V_j(t)$ threshold/floor	$\beta_j$	unsigned int
threshold PRN masked	$M_j$	unsigned int
reset voltage	$R_j$	signed int
negative thresh: reset or saturate	$\kappa_j$	$\{0, 1\}$
$V_j(t)$ reset mode	$\gamma_j$	$\{0, 1, 2\}$
PRNG initial seed value	$\rho_j^{seed}$	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 Definations

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

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

Binary comparision operation:

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

Binary bitwise AND operation:  $\&$

Kronecker delta function:

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

## 1.3 Neuron Specification Equations

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

1. Synaptic Integration

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

2. Leak Integration

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

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

### 3. Threshold, Fire, Reset

$$\eta = \rho_j^T \& M_j \quad (7)$$

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

Spike

$$\begin{aligned} V_j(t) &= \delta(\gamma_j)R_j + \delta(\gamma_j - 1)(V_j(t) - (\alpha_j + \eta_j)) \\ &\quad + \delta(\gamma_j - 2)V_j(t) \end{aligned} \quad (9)$$

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

$$\begin{aligned} V_j(t) &= -\beta_j\kappa_j + [-\delta(\gamma_j)R_j + \delta(\gamma_j - 1)(V_j(t) \\ &\quad + (\beta_j + \eta_j)) + \delta(\gamma_j - 2)V_j(t)](1 - \kappa_j) \end{aligned} \quad (11)$$

endif

Table 2: FUNCTION CONFIGURATION

Name	$j$	$w_{i,j}$	$s_j^{G_i}$	$b_j^{G_i}$	$\epsilon_j$	$\lambda_j$	$c_j$	$\alpha_j$	$\beta_j$	$M_j$	$R_j$	$\kappa_j$	$\gamma_j$
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  $\lambda_j$ . 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:

$$P(\text{subtract 1 from } V(t)) = \frac{|\lambda_j|}{\rho_{j\max}^\lambda} \quad (12)$$

Because the max  $\rho$  is the max value that a 8-bit binary number can stand for (255), in our demo case  $\lambda = -128$  will make the probability 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.  $\lambda_j$  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 differences. First, the weights of 2 synapses are 1 and -1. Second,  $\kappa_j$  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 enhance its performance, if needed. Any bug reports or pull requests are welcome.

Contacts: Zhao Zhixu

- lion\_zzx@stu.xjtu.edu.cn
- <https://github.com/favoryoung/TruenorthSNN-Simulator>