MANCHESTER 1824
The University of Manchester

TECHNISCHE
UNIVERSITÄT
DRESDEN

Next Generation SpiNNaker Chip Update: Developing Numerical Accelerators

Mantas Mikaitis

SpiNNaker2
Universal Spiking Neural Network Architecture

Human Brain Project

Contents

- Details about the next generation SpiNNaker chip
- Historical overview of hardware numerical units
- Why we need exponential function in neuromorphic chips
- Algorithm for exp() and ln()
- Hardware design
- Results
- Further work

SpiNNaker-2 chip pathway

- Prototype chip 1 (codename Santos) was already tested in 2016/2017.
- Prototype chip 2 (codename JIB1) will be produced Q1 2018
- Prototype chip 3 will be produced Q1 2019
- Final SpiNNaker-2 chip will be produced Q2 2020

Currently in the process of fixing bugs with our Dresden colleagues to finalize JIB1 testchip!

Feature comparison

SpiNNaker-1

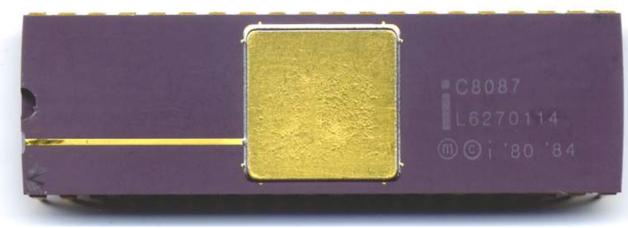
- 18 ARM968 cores
- 96K memory per core
- 128MB Off-chip memory
- 1W power

SpiNNaker-2

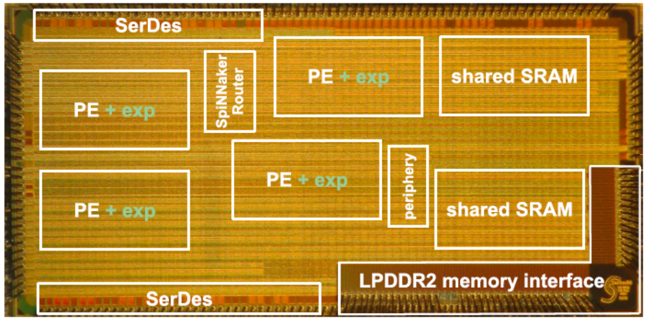
- 144 ARM M4F cores
- 128K memory per core (With capability to use other core’s memories)
- ~2GB Off-chip memory
- Single precision floating point operations
- Random Number Generators
- Machine Learning Accelerator
- Elementary Functions
- 1W power

Hardware elementary functions in various systems

- 1972: HP-35 Scientific pocket calculator contains exp and log with various bases.[3]
- 1980: Intel 8087 Math Coprocessor contains floating-point 2^x and $\log_2$.[3]
- 2010: NVIDIA Fermi/Kepler SFUs contain double precision exp and log with various bases (30-80 cycles).[1,2]
- 2012: Intel Xeon Phi Coprocessors contain single-precision 2^x (8 cycles) and $\log_2$ (4 cycles) functions.
- 2016: SpiNNaker-2 prototype chip Santos contains 15 fractional bits fixed-point exponential (6 cycles).[4]



3



4

3

[1] Demystifying GPU Microarchitecture through Microbenchmarking, Wong et al, 2010; [3] wikipedia.com

[2] A High-Performance Area-Efficient Multifunction Interpolator, NVIDIA, Oberman et al, 2005; [4] Partzsch et al, 2017

Neuromorphic chips bring back hardware elementary functions

Neuromorphic systems are one of the best candidates to simulate a model capable of **general AI**.

This means **super large scale** neural networks with **millions of events in real-time**, using **economically feasible amount of power**.

Most importantly, it means **biologically learning networks**, with **synaptic and neuronal plasticity**, as well as **structural dynamics**.

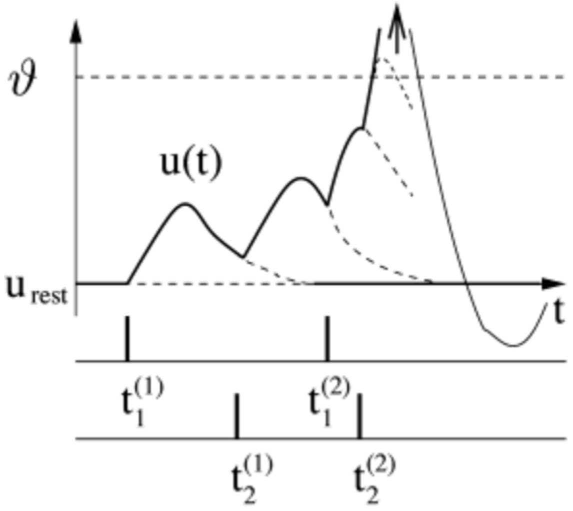
Scales of **10^5 inputs per neuron**, which can be **multicompartment and with dynamic intrinsic properties**.

We can realize this by helping neuromorphic software designers and modellers which comes down to **optimising hardware**.

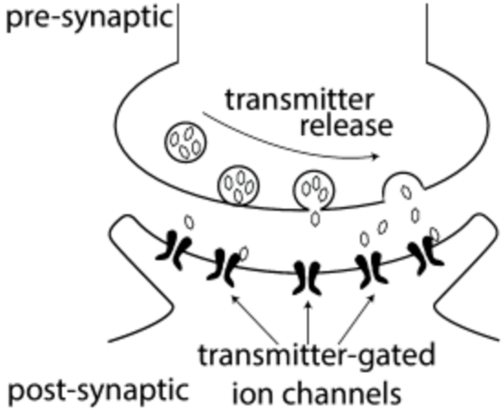
Accelerate most common functions at the lowest level of transistors.

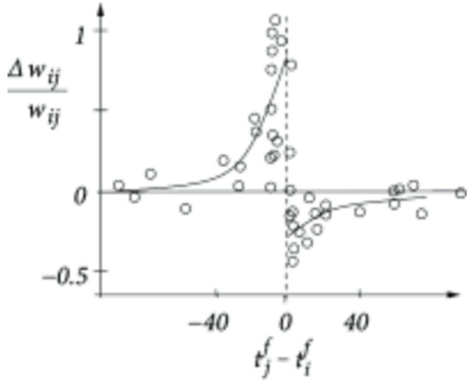
Usage of exponential function

Neuron models [1]



Biophysics of postsynaptic ion channels [1]





Learning: STDP [1]

[1] Neuronal Dynamics, Gerstner et al. 2014

Exponential in SpiNNaker-1

Two main options available:

1. Pre-calculated look-up-tables
2. Software library – ~100 cycles.

Issues:

1. Limited number and size of time constants; Cannot be changed in run-time.
2. Too slow for the current ~31 cycle[(1)] update of a single pair of spikes in STDP processing.

[1] Synapse-Centric Mapping of Cortical Models to the SpiNNaker Neuromorphic Architecture, Knight et al, 2016

Proposed exp/log unit for JIB1

- Exponential and natural logarithm
- I/O: S16.15, S0.31 (Can mix input and output format)
- Precision control
- 3-10 cycle exp
- 3-6 cycle logarithm

Experiment with **approximate computing** techniques - speed up learning rules with some loss in accuracy and compare how well it works.[1]

[1] Is a 4-Bit Synaptic Weight Resolution Enough? – Constraints on Enabling Spike-Timing Dependent Plasticity in Neuromorphic Hardware, Pfeil et al, 2012

Iterative algorithm for exp/log[1]

$$
\begin{aligned}
L_{n+1} &= L_n - \ln\left(1 + d_n 2^{-n}\right) \\
E_{n+1} &= E_n\left(1 + d_n 2^{-n}\right) \\
d_n &= \begin{cases} 1 \text{ if } L_n \geq \ln\left(1 + 2^{-n}\right) \\ 0 \text{ otherwise.} \end{cases}
\end{aligned}
$$

Small LUT

Shift + add

If $L_0 = t$ is less than $\sum_{n=0}^{\infty} \ln\left(1 + 2^{-n}\right)$, this gives:

$$
\begin{aligned}
L_n &\rightarrow 0 \\
n &\rightarrow +\infty.
\end{aligned}
$$

and

$$
\begin{aligned}
E_n &\rightarrow E_0 e^{L_0} \\
n &\rightarrow +\infty.
\end{aligned}
$$

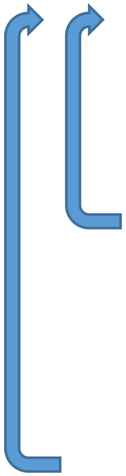
Convergence to 32-bit accuracy in ~32 iterations

Iterative part can be done without ripple carry.

[1] Elementary Functions – Algorithms and Implementation 3rd ed., Muller, 2016

Hardware design process

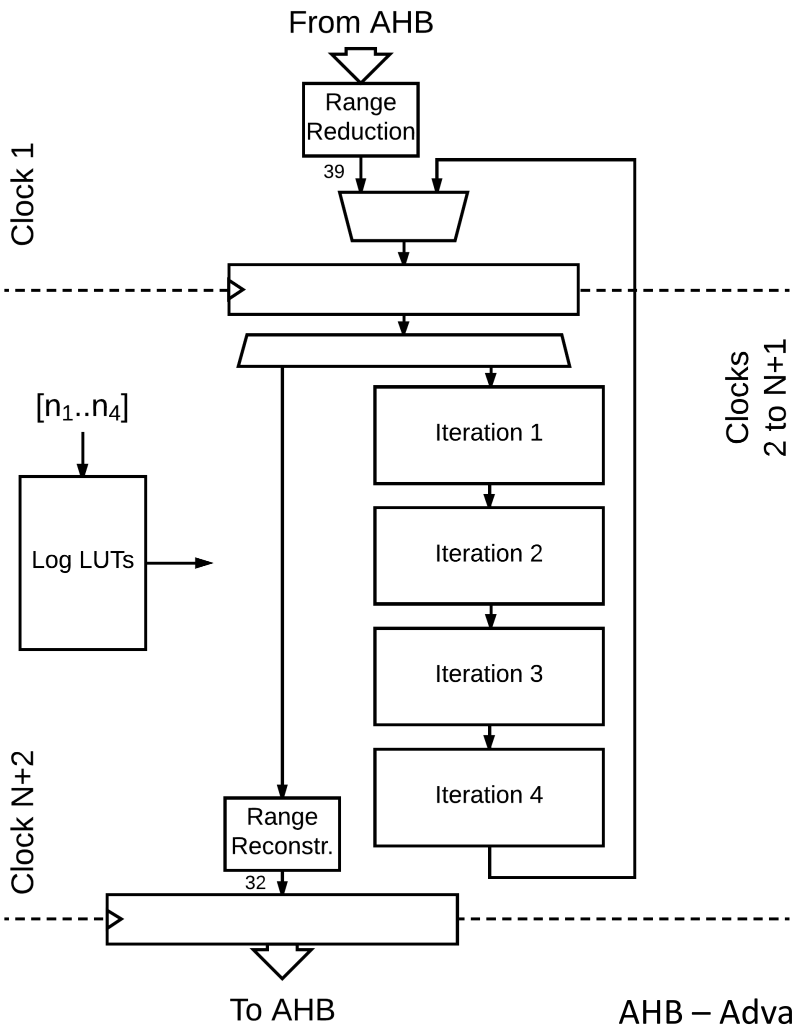
Hardware design process simplified



1. Specification/algorithms/models
2. Register-Transfer-Level design (Verilog)
3. Simulation and comparison of results (Using EDA* tools)
4. Synthesis
    1. Map RTL to physical gate libraries
    2. Check area/leakage/timing of the design
5. …
6. Manufacture

* Electronic Design Automation

Register-Transfer-Level design



AHB – Advanced High-performance Bus
(Interface between processor and slave units)

Results: Quality of outputs (s16.15)

		exp		ln	
Iterations	**Speed (CPU cycles)**	**Accuracy(LSB*)**	**Monotonic**	**Accuracy**	**Monotonic**
32	10	2	Yes	2	Yes
28	9	4	Yes	2	Yes
24	8	8	Yes	2	Yes
20	7	11	Yes	2	Yes
16	6	15	Yes	2	No
12	5	19	No	3	No
8	4	23	No	7	No
4	3	27	No	11	No

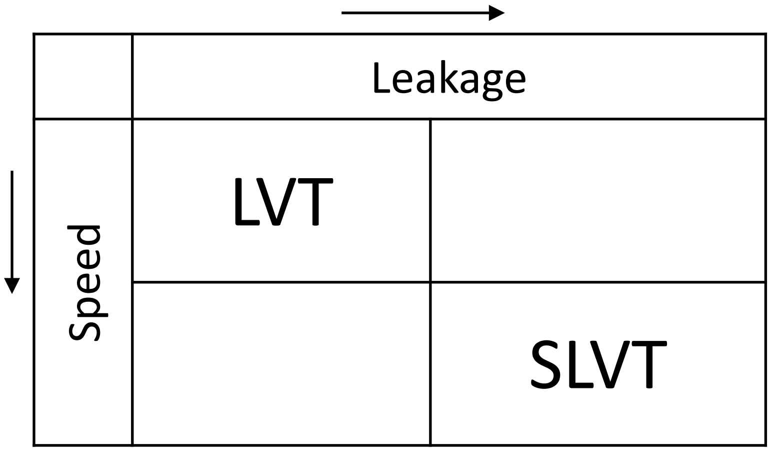
* 1 LSB gives a result with maximum absolute error of 2^{-15}=0.000030517578125;
2 LSB: max error is $2^{-14}+2^{-15}$=0.000091552734375 etc.

Physical library types

RTL design has to be mapped into physical logic gates or standard cells.

Two main categories of cells are usually available: **LVT (Low-Voltage-Threshold)** and **Super-LVT**.

Priority: Minimise leakage as it is causing energy loss even when device is static.



	Leakage	
Speed	LVT	
		SLVT

Results: Area/Speed/Leakage (Stand-alone synthesis 250MHz@0.5V)

Iterations per clock cycle	Area (µm²)	Leakage (mW)	SLVT cells	Latency for full s16.15 accuracy (CPU cycles)
1	7157	0.026	29%	34
2	9905	0.026	29%	18
3	13071	0.05	39.4%	13
4	12254	0.062	49%	10
6	19290	0.232	71%	8
8	19458	0.252	70%	6

Further work

- Floating-point interface.
- Compare to some other similar systems.
- Higher radix number system to improve latency
- Investigate speed/accuracy of spike history trace processing in STDP.
- Trigonometric function wrapper.

Summary

- Hardware elementary functions are back in silicon in full force
- Iterative algorithm and implementation proposed
- Area/Leakage/Speed balance unclear – depends on many factors – even what kind of software is going to be modelled.
- Multi-precision opens research in approximate computing in neuromorphic applications
- Many improvements are possible for the next testchip

Questions