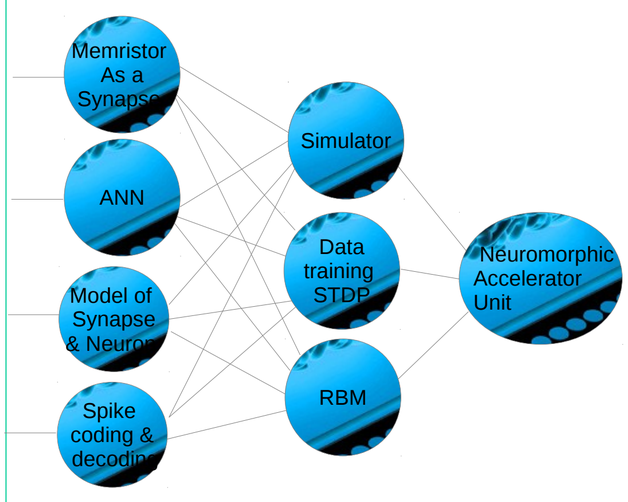
N2S3, a Simulator for the Architecture Exploration of Neuromorphic Accelerators

Mahyar Shahsavari, Philippe Devienne, Pierre Boulet

N2S3, a Simulator for the Architecture Exploration of Neuromorphic Accelerators

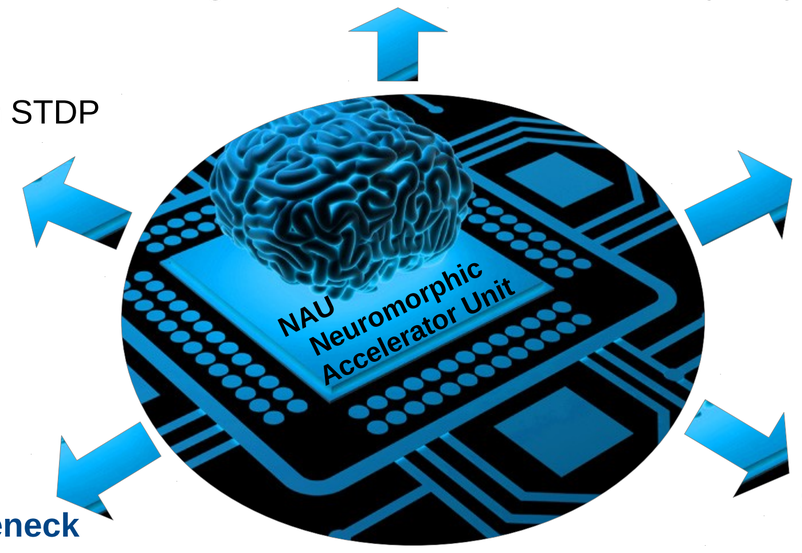
Mahyar Shahsavari, Philippe Devienne, Pierre Boulet (Lille University)

Neuromorphic accelerator



Reconfigurable/Flexible
Using Unsupervised Learning with STDP

Parallel Computing/Big Data
Using Restricted Boltzmann Machine (RBM)



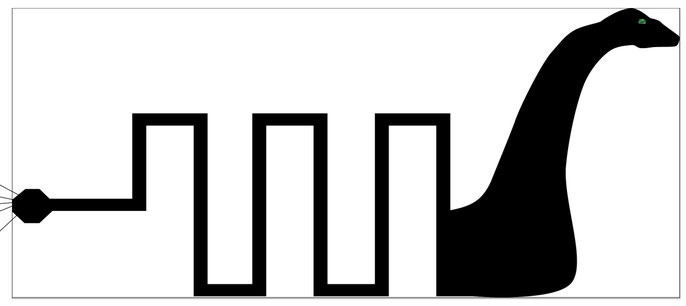
Low-Power
Using Event-driven Computing

Efficient in Speed/No Memory Bottleneck
Using Arithmetic Unit (Neuron) besides Memory (Synapse)

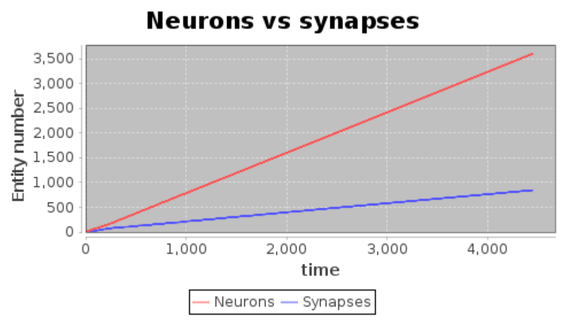
Scalable
Using Memristor Nanodevice as a Synapse

Neural Network Scalable Spiking Simulator (N2S3)

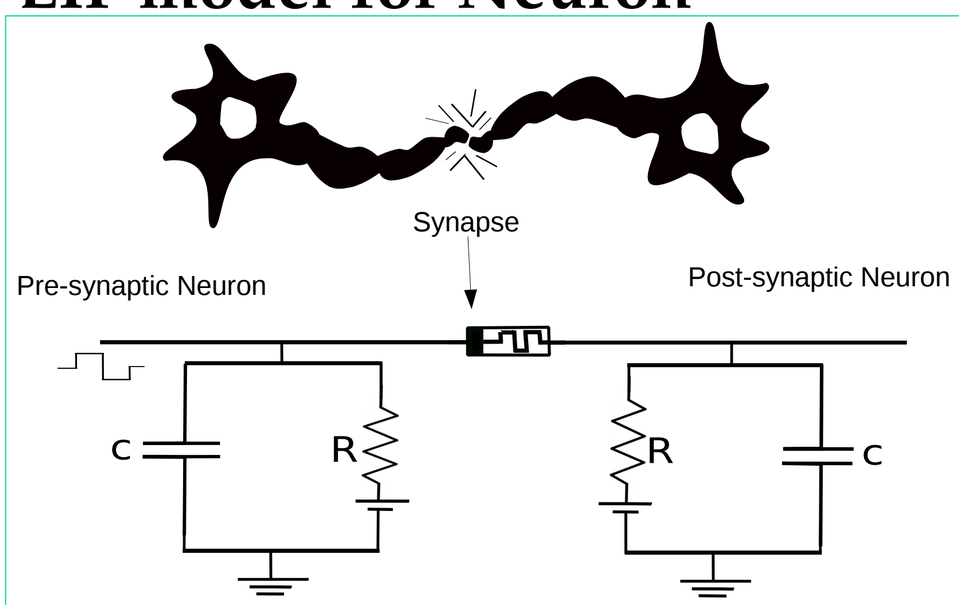
- Event-based
- Analytic Model
- Scalable
- Extensible
- Scala Language



- Using Address Event Representation (**AER**)
- Using **Scala** programming language
- **Akka actor** library for parallel processing and concurrency
- A synapse/neuron is presented by an **Actor**
- A spike is a **message** for the Actors
- Flexibility of the **synchronization**



LIF model for Neuron



Neuron Model:

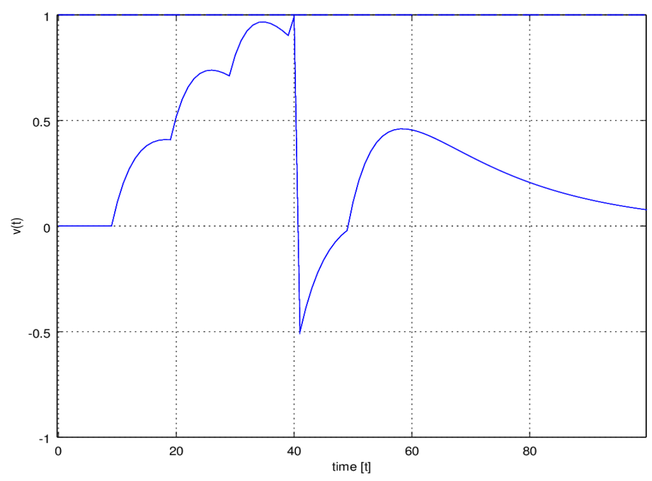
Leaky Integrate & fire

$$\tau_{neuron}\frac{dv}{dt}+G_{neuron}(t)V=I_{input}$$

$$G_{neuron}(t)=G_{leak}+G_{syn}(t)$$

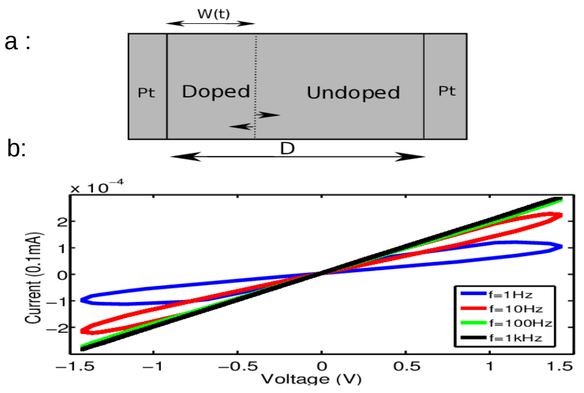
Synapse Model:

$$G_{syn}(t)=G_{syn}(t_0)\exp-(\frac{t-t_0}{\tau_{syn}})$$



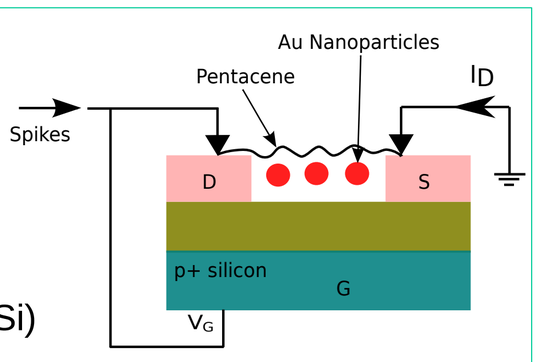
Model of Memristor as a Synapse

```
override def processSpike(s: Spike) =
    s match {
      case PreSynapticSpike(t, dest) =>
        preSpikes += s
        postSpikes retain (postS => t - postS.timestamp <= stdpWindow)
        postSpikes foreach { _ =>
          decreaseWeight(t)
        }
       fire(PostSynapticForwardSpike(t, post, g*v))
      case PostSynapticBackwardSpike(t, _, _) =>
        postSpikes += s
        preSpikes retain (preS => t - preS.timestamp <= stdpWindow)
        preSpikes foreach { _ =>
          increaseWeight(t)
        }
        done()
      case _ => super.processSpike(s)
    }
  def increaseWeight(timestamp: Int) = {
      var dg = alf_p * exp(-beta * ((g-g_min)/(g_max-g_min)))
      g = min(g_max,g+dg)
     }
  def decreaseWeight(timestamp: Int) = {
      var dg = alf_m * exp(-beta * ((g_max-g)/(g_max-g_min)))
      g = max(g_min,g-dg)}
```



a) the memristor structure, b) current versus voltage diagram, sinusoidal input with an amplitude of 1.5v, different frequencies, R_{ON}=100Ω, R_{OFF}=15kΩ, D=10nm, $\mu_v=10^{-10}cm^2s^{-1}V^{-1}$.

- Different Memristors:
  - TiO2
  - Spintronic
  - Organic (Polymeric)
  - Amorphous silicon (a-Si)
  - Ferroelectric Memristor



Organic memristor fabricated at Lille University (F.Alibart et.al 2010)

Université de Lille
cnrs
IRCICA Research Institute FR3024 CNRS
CRIStAL Centre de Recherche en Informatique, Signal et Automatique de Lille

Centre de Recherche en Informatique,
Signal et Automatique de Lille (CRIStAL)
Université de Lille 1,
Cité Scientifique F-59655,
Villeneuve d'Ascq cedex, France
Phone: +33 (0)3 20 33 59 51

HiPEAC COMPILATION ARCHITECTURE