

A Temporal Coding Hardware Implementation for Spiking Neural Networks

Marco Aurelio Nuño-Maganda¹, Cesar Torres-Huitzil²

¹Universidad Politécnica de Victoria (UPV)

**²Centro de Investigación y Estudios Avanzados de Tamaulipas
(CINVESTAV-TAMAULIPAS)**

Ciudad Victoria, México




mnunom@upv.edu.mx

ctorres@tamps.cinvestav.mx

Outline

1. Introduction
2. GRF-Based Temporal Coding
3. Hardware Implementation
4. Results
5. Discussion
6. Conclusion and Future Work

1. Introduction

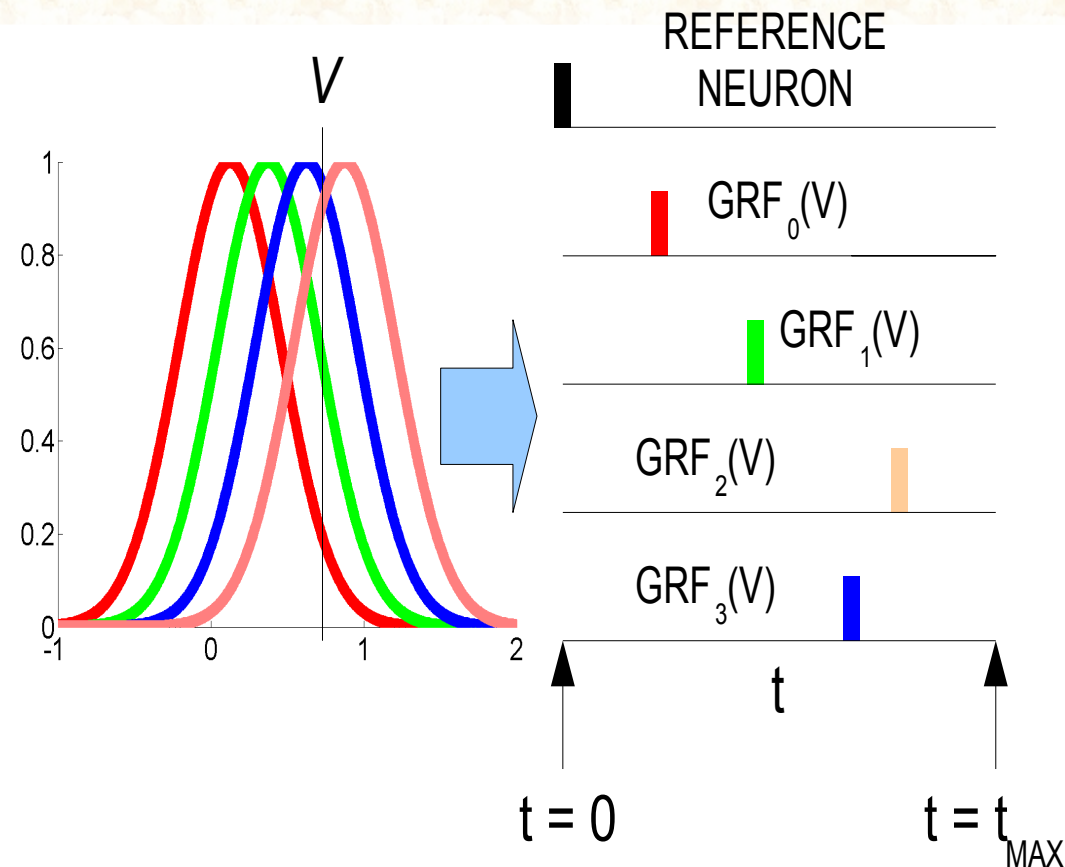
-  Spiking Neural Networks (SNNs) states that information among neurons is interchanged via pulses or spikes.
-  SNNs have the ability for processing static patterns and dynamic patterns that exhibits rich temporal characteristics.
-  Important issue: information coding.

1. Introduction

- 🧠 Main approaches:
- 🧠 Rate coding. The information is encoded in the neuron firing time.
- 🧠 Temporal coding. The information is encoded by the timing of spike
- 🧠 Population coding. The information is encoded by the activity of different pools of neurons.
- 🧠 There are strong debates about the question of which neural codes are used for biological neural systems
- 🧠 There is a growing evidence that the brain may use all of them

2. GRF-based Temporal Coding

- 🧠 A technique for coding input variables
- 🧠 Inspired in the Local Receptive Fields of Biological Neurons
- 🧠 Each dimension is associated to a graded overlapping profile.
- 🧠 Key parameters: width and center position

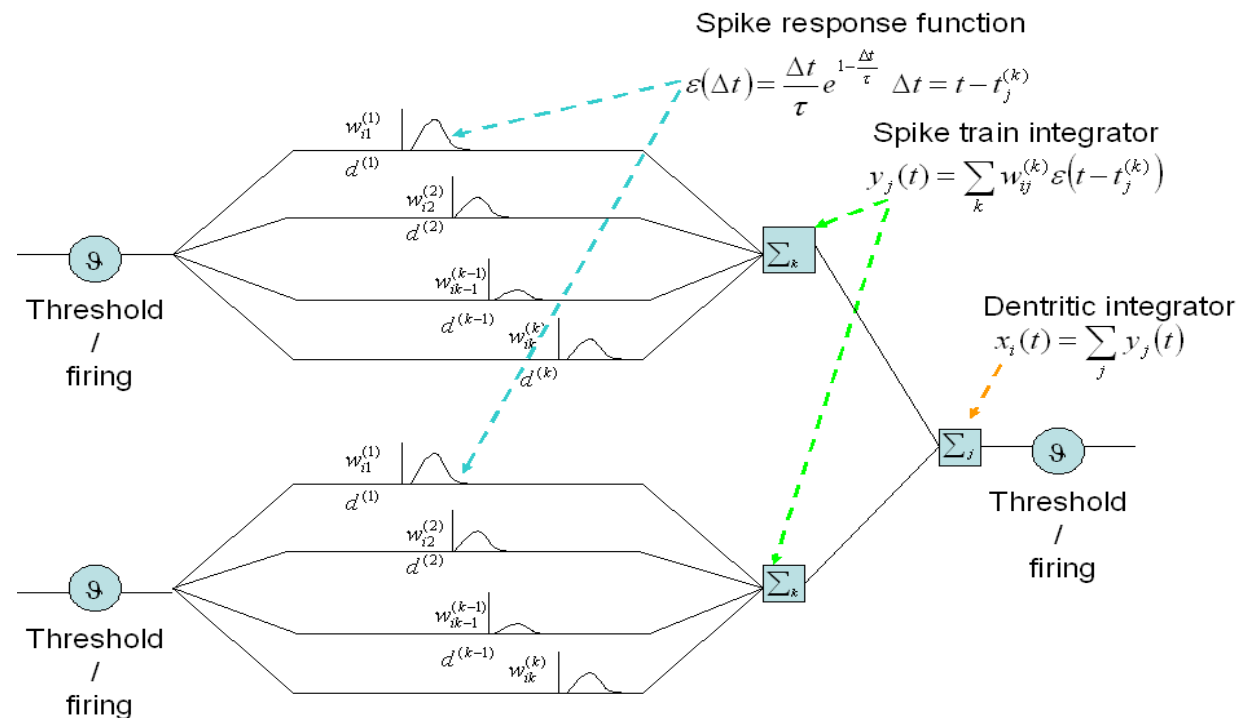


- 🧠 Arguments in Favor
- 🧠 Not sensitive to scale changes
- 🧠 Performs a sparse coding

2. GRF-based Temporal Coding

Application: Multilayer FF-SNNs

- The GRF output in the input firing time of a neuron in one Feed-Forward SNN.
- Each connection is divided in a set of multiple synaptic connections
- Weight and delays associated to each terminal



2. Gaussian Receptive Fields (GRFs) for Neuron Coding

- A real value is encoded by an array of receptive fields.
- For a variable with a range $[I_{min}^n, \dots, I_{max}^n]$, a set of m Gaussian Receptive Fields are used. The center b_i is given by:

$$b_i = I_{min} + \frac{1}{m-2} \frac{2i-3}{(I_{max} - I_{min})},$$

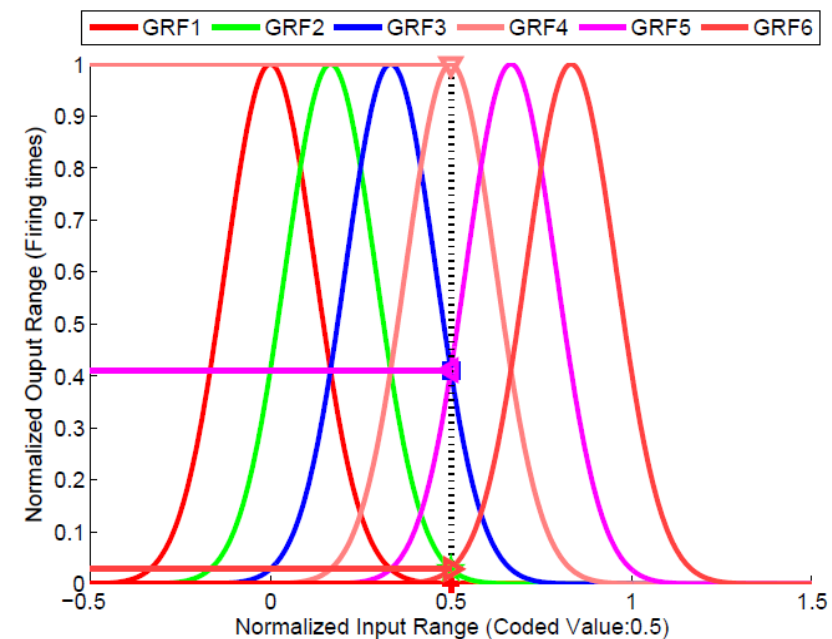
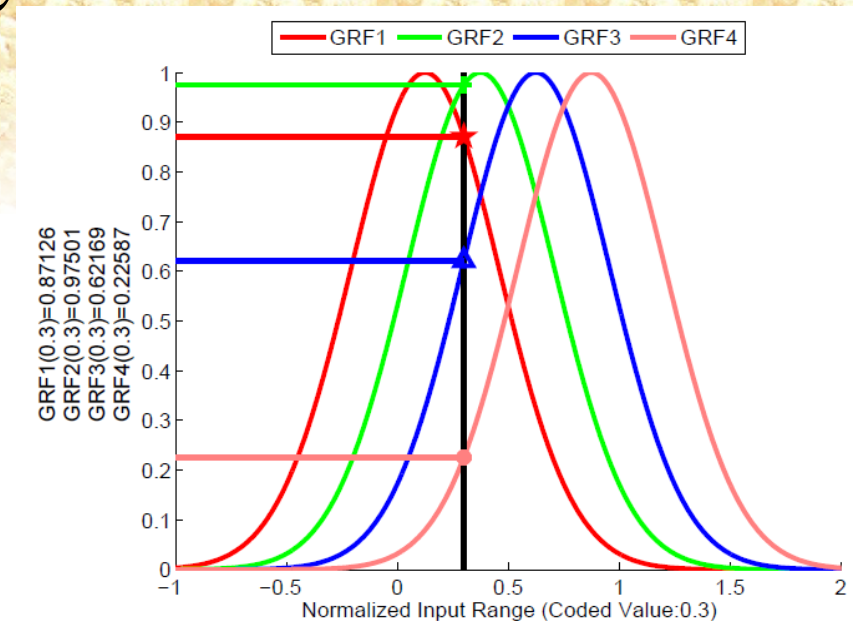
- And the width σ of each RF neuron i is given by:

$$\sigma = \frac{1}{(m+2)} \frac{1}{\beta(I_{max} - I_{min})},$$

- Where the proposed value for β belongs to the range $[1,2]$

2. Gaussian Receptive Fields Coding Examples

- Steps for coding a set of input values:
 - Each input value is normalized.
 - Each normalized value is evaluated in each GRF.
 - The evaluation obtained in each GRF is assigned to one single input neuron.



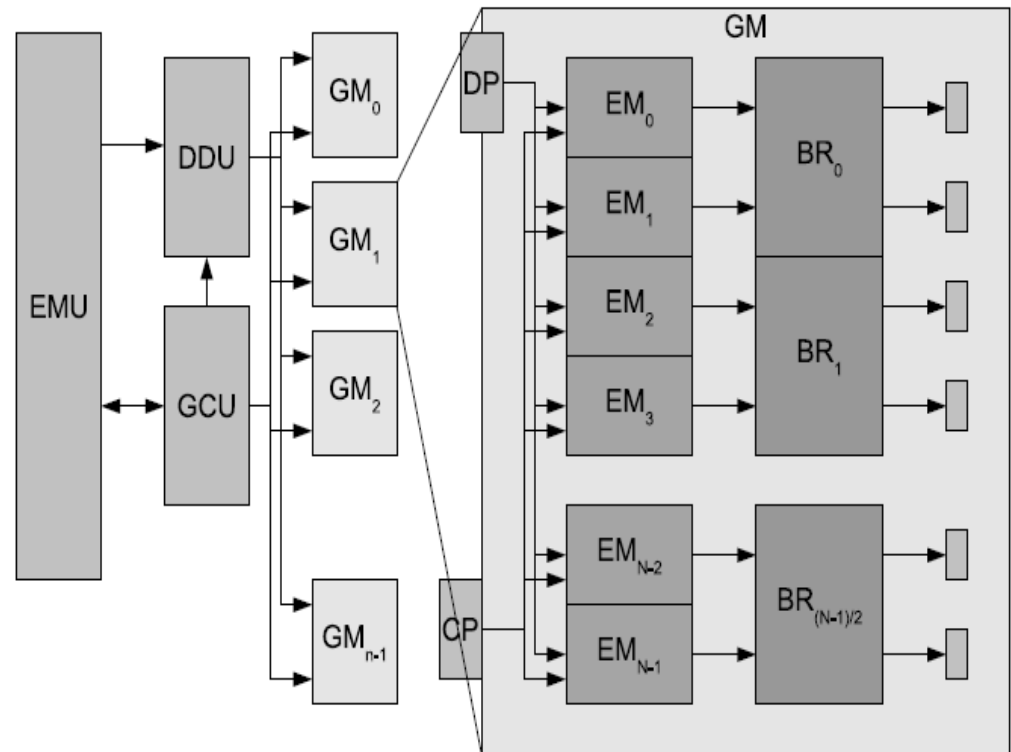
2. Gaussian Receptive Fields

Issues to be addressed

- Hardware resource simplification
- Scalability and Flexibility
- Data representation
- Types of Parallelism

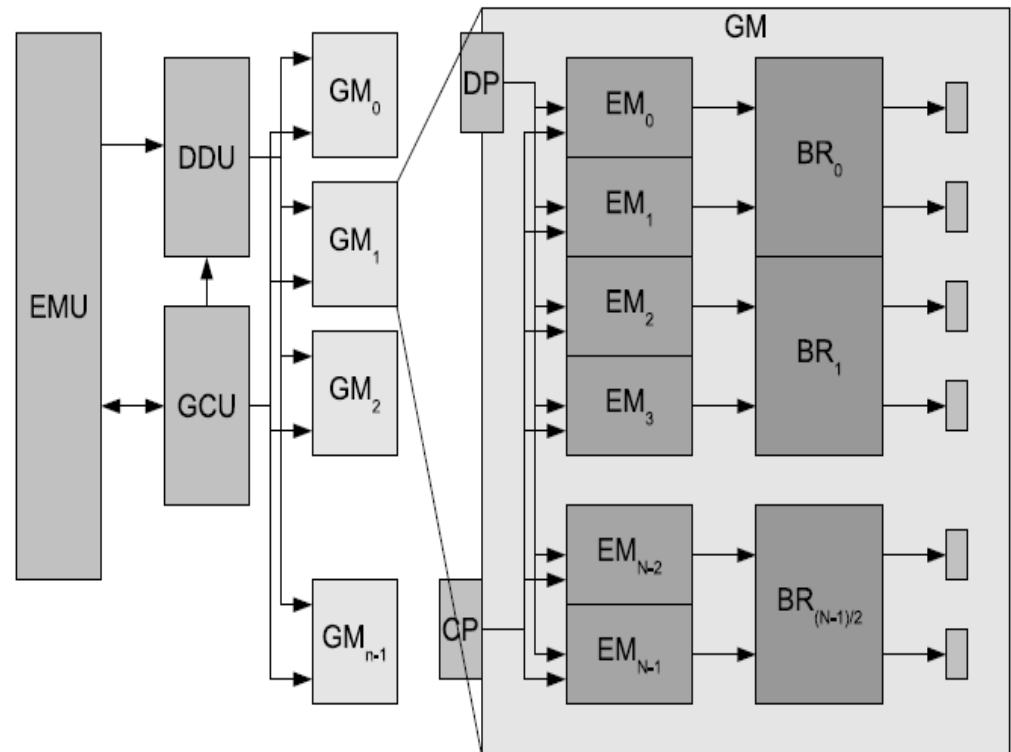
3. Hardware Implementation

- 🧠 The Input data set is stored in external memory (accessed through the External Memory Unit - EMU)
- 🧠 The input data set is analyzed for obtaining relevant information (Data Distribution Unit DDU)
- 🧠 The Global Control Unit (GCU) generates the synchronization signals for the components



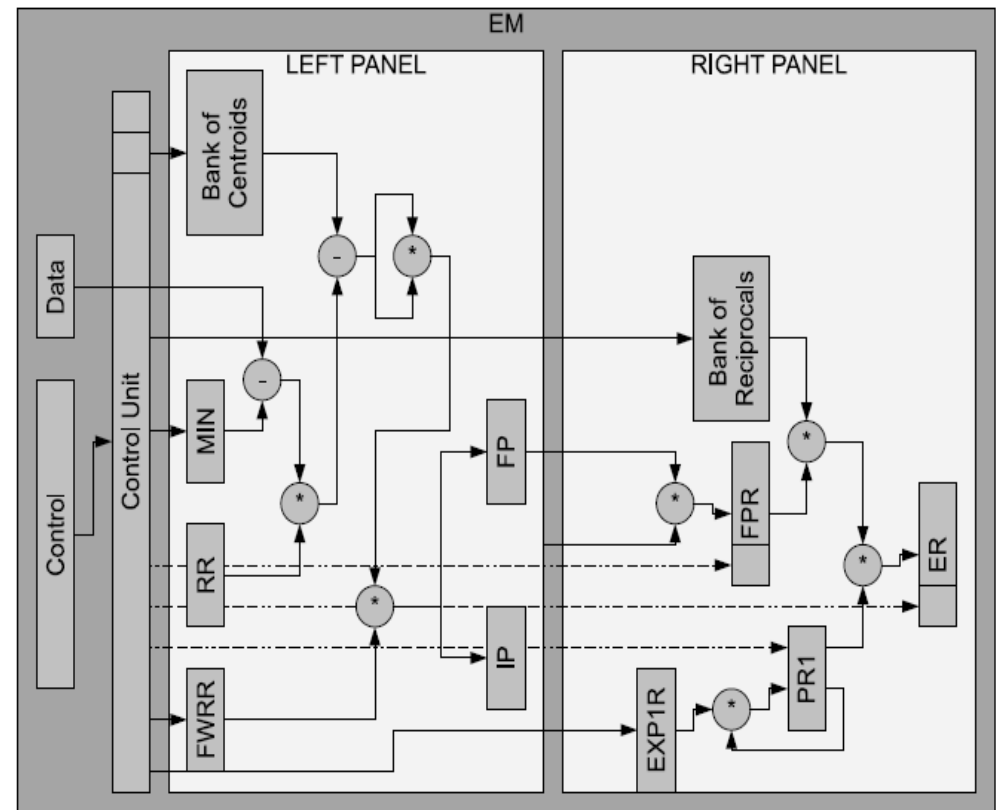
3. Hardware Implementation

- Each sample of the dataset is sent to the Gaussian Modules (GMs) for obtaining the coding.
- Each GM has the following input ports:
 - Data Port (DP) - Contains the data to be processed.
 - Control Port (CP) - Contains several synchronization signals



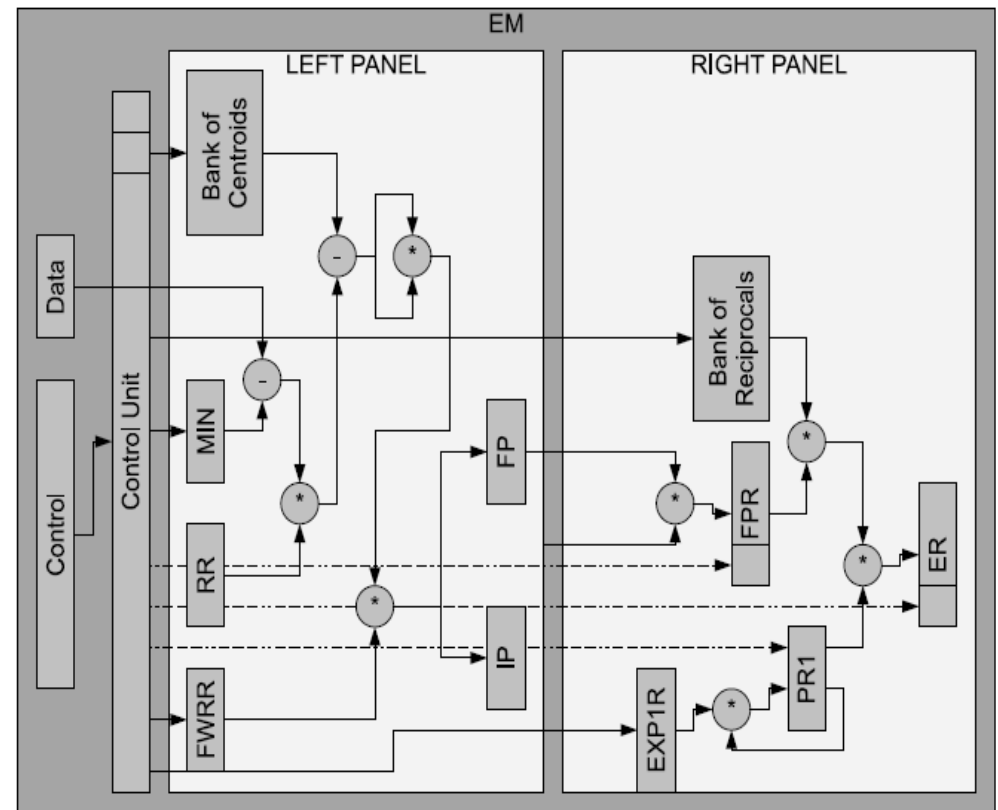
3. Hardware Implementation

- 🍿 The main components of the GM are:
- 🍿 Control Unit -
- 🍿 Min Register (MR) -
- 🍿 Bank of Centroids (BCs)
- 🍿 Integer Part Register (IPR)
- 🍿 Fractional Part Register (FPR)
- 🍿 Bank of Reciprocals (BRs)



3. Hardware Implementation

- Other components of the GM are:
- Reciprocal Register (RR) -
- N-Power of FP Register (NPFPR)
- Exponential of 1 Register (E1R)
- Exponential Register (ER)



4. Results

Tools for HW implementation:

- Target FPGA: Virtex II Pro
- Target Board: Alphadata AXM-XPL
- Handel-C modeling
- VHDL synthesizing

Tools for SW implementation:

- PC with Pentium IV Processor running at 3.66 GHz.
- Visual C++

Implementation of GMs with several EMs

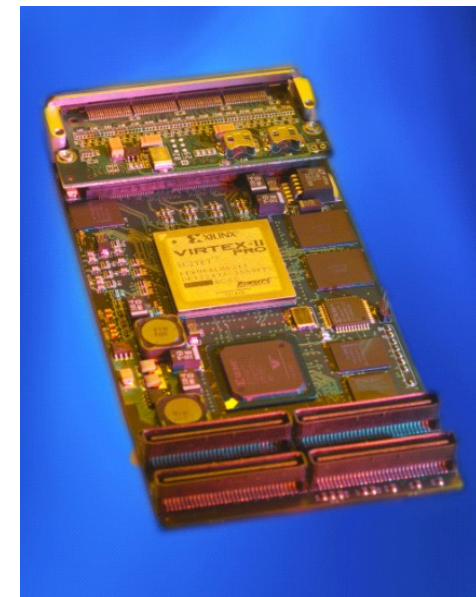
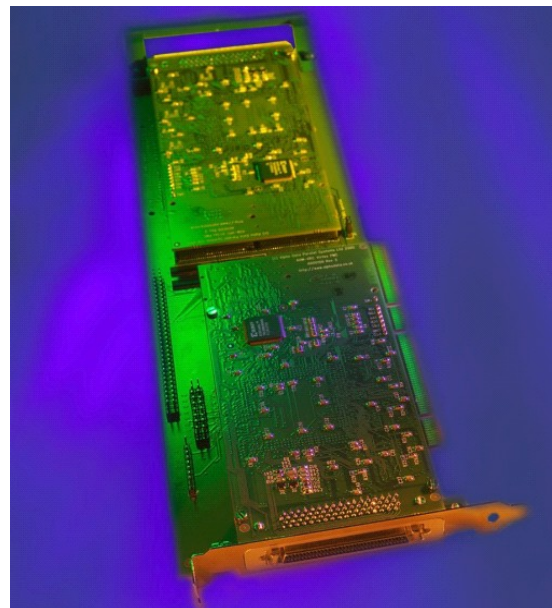
–At least 4 EMs with each GMs

4. Results – Hardware Platform

🧠 Resource available in the target FPGA device:

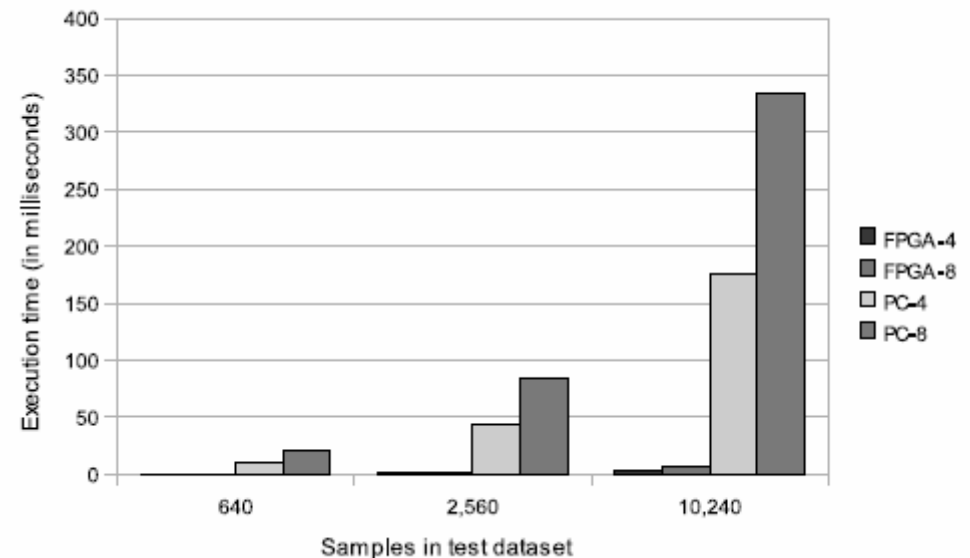
FPGA	4-Input LUTs	Slice-FFs	Total Slices	Total BRAMs	Total MULT18x18s
xc2vp30-6ff896	27,392	27,392	13,696	136	136

🧠 Target FPGA platform (Alphadata ADMXPL PMC board)



4. Results - Performance

- Both software and hardware implementations are compared
- Several dataset sizes (rows and columns) are compared
- There is a performance improvement of at least 50x when using moderate hardware resources




4. Results - Precision

- For comparing both SW and HW implementation, the MSE metric was used
- The architecture is flexible for supporting several precisions
- Several bit precision (for fractional part) have been evaluated
- The 8-bit precision is enough for several machine learning applications

bit precision	MSE
8-bit	16.5 e-1
10-bit	9.5 e-2
12-bit	6.17 e-2
14-bit	7.23 e-3
16-bit	9.45 e-3

4. Results – Hardware utilization

 Hardware resources and maximum clock frequency for each variation of the proposed architecture were obtained

Processors	Slices	Slice-FF	4-input LUTs	BRAMs	MULT18X18s	Gate count	Maximum clock frequency
4	1,515 11 %	1,583 6 %	2,153 8 %	1 1 %	25 18 %	408,807	85.7 MHz
8	2,409 18 %	2,398 9 %	3,546 13 %	2 1 %	50 37 %	527,468	84.2 MHz
12	3,272 24 %	3,212 12 %	4,805 18 %	3 2 %	75 55 %	645,143	80.5 MHz
16	4,018 29 %	4,031 15 %	6,054 22 %	4 3 %	100 74 %	762,048	76.5 MHz

5. Discussion

- 🧠 The base architecture is designed to be flexible for processing several GRFs with potential applications in different domains.
- 🧠 The proposed architecture is designed to work with several columns of the source dataset.
- 🧠 The importance of computing in parallel the temporal coding consists on the possibility of integrating the proposed architecture with any SNN in a pipelined fashion,

6. Conclusion and Future Directions

- 🧠 At least 50X is obtained with the proposed architecture. Several performance-resource trade-offs can be established, since dedicated multiplier resources are the most demanded ones in the current implementation.
- 🧠 The integration of the proposed architecture with other processing modules for a complete implementation of SNNs will be analyzed.

A decorative header with a repeating pattern of small, light yellow flowers on a slightly darker yellow background.

Thank you !

Marco Nuno-Maganda

mnunom@upv.edu.mx