Chapter 11
FPGA Coprocessor for Simulation of Neural Networks Using Compressed Matrix Storage

Jörg Bornschein
Goethe University Frankfurt am Main, Germany

ABSTRACT
An FPGA-based coprocessor has been implemented which simulates the dynamics of a large recurrent neural network composed of binary neurons. The design has been used for unsupervised learning of receptive fields. Since the number of neurons to be simulated (>10^4) exceeds the available FPGA logic capacity for direct implementation, a set of streaming processors has been designed. Given the state- and activity vectors of the neurons at time t and a sparse connectivity matrix, these streaming processors calculate the state- and activity vectors for time t + 1. The operation implemented by the streaming processors can be understood as a generalized form of a sparse matrix vector product (SpMxV). The largest dataset, the sparse connectivity matrix, is stored and processed in a compressed format to better utilize the available memory bandwidth.

INTRODUCTION
The connectionist model of information processing assumes that useful behavior is an emergent property of a huge number of relatively simple interacting units. A given connectionist model describes two things: On the one hand it describes the behavior of the individual units, the neurons; how they react to incoming stimuli and the output they generate. On the other hand it describes the connectivity between these units. Typically the exact structure of these connections varies over time; sometimes because of a learning process, sometimes, on smaller timescales, to facilitate dynamic plasticity.

For some models a mathematical abstraction can be found which allows the emergent macroscopic behavior to be understood without simulating the system at hand. In general, this is not the case and simulations are the predominant tool to understand the behavior of the modeled systems. Interesting systems often consist of more
than 10,000 neurons with non-trivial connectivity between them. The human brain for example consists of about $10^{11}$ neurons with more than $10^{14}$ synapses between them. When considering strategies how to simulate connectionist systems, we can think of two extreme cases:

1. Sequentially simulate one neuron after another. For every neuron all input signals are collected and the neuron’s output is computed.
2. Physically instantiate a simulation circuit for every neuron and make sure they all receive the input stimuli they need to compute a time step.

While in nature we find the second extreme realized, an arbitrary working point in between can be chosen. It is up to the software or hardware engineer to find a suitable trade-off, which makes best use of available hardware for a given system. In this work we assume that the number of neurons to be simulated exceeds the number of neurons that can be directly instantiated due to hardware resource limits. Given this assumption it is evident that, irrespective of the exact nature of the neuron simulator, be it software running on a CPU or a hardware implementation of a state-machine, some kind of multiplexing mechanism must exist to simulate a large number of neurons. The data needed to simulate a time step can be divided into three parts: A neural state vector $\mathbf{n}(t)$ which records the internal state of all neurons; an activity vector $\mathbf{a}(t)$, which can be directly derived from $\mathbf{n}(t)$ and stores if a neuron is firing or not, and a connection matrix $\mathbf{M}$ which determines if and how a neuron $j$ is connected to a neuron $i$. In this work we assume that the connection matrix $\mathbf{M}$ is best stored as a sparsely populated matrix, because only a small fraction of the matrix elements contain non-zero elements. We further assume, that the state and activity vectors $\mathbf{n}$ and $\mathbf{a}$ are stored as dense vectors.

It is crucial to understand that all currently available microprocessor architectures consist of processor elements with fast but relatively small local on-chip memory. External to these processor elements is larger, but much slower external DRAM (Wulf & McKee, 1995). Usually, at least for interesting systems, the data volume is larger than the size of the available internal on-chip memory. As a result the simulation performance is a function of memory bandwidth even with arbitrary fast processor elements.

Here we describe an architecture and the concrete implementation of an FPGA-based coprocessor which fully utilizes the available memory bandwidth to calculate $\mathbf{n}(t + 1)$ given $\mathbf{n}(t), \mathbf{a}(t)$ and $\mathbf{M}$. Each of the three datasets is read sequentially, without random access, from external memory. The largest dataset, the connection matrix $\mathbf{M}$, is stored in a compressed format using a variable-length entropy code to more efficiently use the available memory bandwidth.

This chapter is divided into 3 sections: In section 2 we introduce a model of the cortical macrocolumn, the artificial neural network we are going to simulate. This neural network has been successfully applied to unsupervised learning tasks and serves as an example for the types of networks the architecture can simulate.

Section 3 highlights the predominant operation when simulating cortical macrocolumns: The sparse matrix vector product (SpMxV). This operation is of general interest in engineering and computational science and we give a quick overview of the strategies deployed and the data formats used when implementing SpMxV. In this section we also introduce a new storage format for sparse matrices, the Compressed Incremental Row Storage (CIRS) format. This format is designed to reduce bandwidth demand between the external memory and the processor elements performing SpMxV.

In section 4 we describe our implementation of an FPGA coprocessor which has been specifically
Related Content

Multiobjective Optimization and Artificial Immune Systems: A Review
[www.igi-global.com/chapter/multiobjective-optimization-artificial-immune-systems/19637?camid=4v1a](www.igi-global.com/chapter/multiobjective-optimization-artificial-immune-systems/19637?camid=4v1a)

Optimization of a Three Degrees of Freedom DELTA Manipulator for Well-Conditioned Workspace with a Floating Point Genetic Algorithm

MP Modelling of Glucose-Insulin Interactions in the Intravenous Glucose Tolerance Test
Vincenzo Manca, Luca Marchetti and Roberto Pagliarini (2014). *Natural Computing for Simulation and Knowledge Discovery* (pp. 171-183).

Neural Networks in Cognitive Science: An Introduction
[www.igi-global.com/chapter/neural-networks-cognitive-science/58301?camid=4v1a](www.igi-global.com/chapter/neural-networks-cognitive-science/58301?camid=4v1a)