
FPGA IMPLEMENTATIONS OF NEURAL NETWORKS

FPGA IMPLEMENTATIONS OF NEURAL NETWORKS

Edited by
AMOS R. OMONDI
Flinders University, Australia

JAGATH RAJAPAKSE
Nanyang Technological University, Singapore

Kluwer Academic Publishers
Boston/Dordrecht/London

Contents

Preface	ix
1	
FPGA Neurocomputers	1
<i>Amos R. Omondi, Jagath C. Rajapakse and Mariusz Bajger</i>	
1. Introduction	1
2. Review of neural-network basics	3
3. ASIC vs. FPGA neurocomputers	9
4. Parallelism in neural networks	12
5. Xilinx Virtex-4 FPGA	13
6. Arithmetic	15
7. Activation-function implementation: unipolar sigmoid	21
8. Performance evaluation	32
9. Conclusions	34
2	
Arithmetic precision for BP networks	37
<i>Medhat Moussa and Shawki Areibi and Kristian Nichols</i>	
1. Introduction	37
2. Background	39
3. Architecture design and implementation	43
4. Experiments using logical-XOR problem	48
5. Results and discussion	50
6. Conclusions	56
3	
FPNA: Concepts and properties	63
<i>Bernard Girau</i>	
1. Introduction	63
2. Choosing FPGAs	65
3. FPNAs, FPNNs	71
4. Correctness	86
5. Underparameterized convolutions by FPNNs	88
6. Conclusions	96
References	99
4	
FPNA: Applications and implementations	105

Bernard Girau

1.	Summary of Chapter 3	106
2.	Towards simplified architectures: symmetric boolean functions by FPNAs	107
3.	Benchmark applications	111
4.	Other applications	115
5.	General FPGA implementation	118
6.	Synchronous FPNNs	123
7.	Implementations of synchronous FPNNs	127
8.	Implementation performances	133
9.	Conclusions	136

References	139
------------	-----

5

Back-Propagation Algorithm Achieving 5 GOPS on the Virtex-E	143
---	-----

Kolin Paul and Sanjay Rajopadhye

1.	Introduction	144
2.	Problem specification	145
3.	Systolic implementation of matrix-vector multiply	147
4.	Pipelined back-propagation architecture	148
5.	Implementation	150
6.	MMA ALPHA design environment	153
7.	Architecture derivation	155
8.	Hardware generation	161
9.	Performance evaluation	163
10.	Related work	165
11.	Conclusion	166
Appendix	168	

References	171
------------	-----

6

FPGA Implementation of Very Large Associative Memories	175
--	-----

*Dan Hammerstrom, Changjian Gao, Shaojuan Zhu, Mike Butts**

1.	Introduction	175
2.	Associative memory	176
3.	PC Performance Evaluation	186
4.	FPGA Implementation	192
5.	Performance comparisons	197
6.	Summary and conclusions	198
References	200	

7

FPGA Implementations of Neural Networks - Neocognitron Case	203
---	-----

Alessandro Noriaki Ide and José Hiroki Saito

1.	Introduction	203
2.	Neocognitron	204
3.	Alternative neocognitron	207

<i>Contents</i>	vii
4. Reconfigurable computer	211
5. Reconfigurable orthogonal memory multiprocessor	212
6. Alternative neocognitron hardware implementation	215
7. Performance analysis	221
8. Applications	224
9. Conclusions	227
References	228
8	
Self Organizing Feature Map for Color Quantization on FPGA	231
<i>Chip-Hong Chang, Menon Shibu and Rui Xiao</i>	
1. Introduction	231
2. Algorithmic adjustment	234
3. Architecture	237
4. Implementation	241
5. Experimental results	245
6. Conclusions	248
References	248
9	
Implementation of Self-Organizing Feature Maps in Reconfigurable Hardware	253
<i>Mario Porrman, Ulf Witkowski, and Ulrich Rückert</i>	
1. Introduction	253
2. Using reconfigurable hardware for neural networks	254
3. The dynamically reconfigurable rapid prototyping system RAPTOR2000	256
4. Implementing self-organizing feature maps on RAPTOR2000	258
5. Conclusions	273
References	273
10	
FPGA Implementation of a Fully and Partially Connected MLP	277
<i>Antonio Canas, Eva M. Ortigosa, Eduardo Ros and Pilar M. Ortigosa</i>	
1. Introduction	277
2. MLP/XMLP and speech recognition	279
3. Activation functions and discretization problem	282
4. Hardware implementations of MLP	290
5. Hardware implementations of XMLP	297
6. Conclusions	300
Acknowledgments	301
References	303
11	
FPGA Implementation of Non-Linear Predictors	305
<i>Rafael Gadea-Girones and Agustín Ramírez-Agundis</i>	
1. Introduction	306
2. Pipeline and back-propagation algorithm	307
3. Synthesis and FPGAs	312

4.	Implementation on FPGA	321
5.	Conclusions	328
	References	328
12		
	The REMAP reconfigurable architecture: a retrospective	331
	<i>Lars Bengtsson, Arne Linde, Tomas Nordstrøm, Bertil Svensson, and Mikael Taveniku</i>	
1.	Introduction	332
2.	Target Application Area	333
3.	REMAP- β – design and implementation	341
4.	Neural networks mapped on REMAP- β	352
5.	REMAP- γ architecture	359
6.	Discussion	360
7.	Conclusions	363
	Acknowledgments	363
	References	363

Preface

During the 1980s and early 1990s there was significant work in the design and implementation of hardware neurocomputers. Nevertheless, most of these efforts may be judged to have been unsuccessful: at no time have hardware neurocomputers been in wide use. This lack of success may be largely attributed to the fact that earlier work was almost entirely aimed at developing custom neurocomputers, based on ASIC technology, but this technology was never sufficiently developed or competitive enough to justify large-scale adoption. On the other hand, gate-arrays of the period mentioned were never large enough nor fast enough for serious artificial-neural-network (ANN) applications. But technology has now improved: the capacity and performance of current FPGAs are such that they present a much more realistic alternative. Consequently neurocomputers based on FPGAs are now a much more practical proposition than they have been in the past. This book summarizes some work towards this goal and consists of 12 papers that were selected, after review, from a number of submissions. The book is nominally divided into three parts: Chapters 1 through 4 deal with foundational issues; Chapters 5 through 11 deal with a variety of implementations; and Chapter 12 looks at the lessons learned from a large-scale project and also reconsiders design issues in light of current and future technology.

Chapter 1 reviews the basics of artificial-neural-network theory, discusses various aspects of the hardware implementation of neural networks (in both ASIC and FPGA technologies, with a focus on special features of artificial neural networks), and concludes with a brief note on performance-evaluation. Special points are the exploitation of the parallelism inherent in neural networks and the appropriate implementation of arithmetic functions, especially the sigmoid function. With respect to the sigmoid function, the chapter includes a significant contribution.

Certain sequences of arithmetic operations form the core of neural-network computations, and the second chapter deals with a foundational issue: how to determine the numerical precision format that allows an optimum tradeoff between precision and implementation (cost and performance). Standard single or double precision floating-point representations minimize quantization

errors while requiring significant hardware resources. Less precise fixed-point representation may require less hardware resources but add quantization errors that may prevent learning from taking place, especially in regression problems. Chapter 2 examines this issue and reports on a recent experiment where we implemented a multi-layer perceptron on an FPGA using both fixed and floating point precision.

A basic problem in all forms of parallel computing is how best to map applications onto hardware. In the case of FPGAs the difficulty is aggravated by the relatively rigid interconnection structures of the basic computing cells. Chapters 3 and 4 consider this problem: an appropriate theoretical and practical framework to reconcile simple hardware topologies with complex neural architectures is discussed. The basic concept is that of *Field Programmable Neural Arrays* (FPNA) that lead to powerful neural architectures that are easy to map onto FPGAs, by means of a simplified topology and an original data exchange scheme. The first of the two chapters gives the basic definition and results of the theoretical framework. The subsequent chapter shows how FPNAs lead to powerful neural architectures that are easy to map onto digital hardware. applications and implementations are described, focusing on a class

Chapter 5 presents a systolic architecture for the complete back propagation algorithm. For a neural network with N input neurons, P hidden layer neurons and M output neurons, the proposed architecture with P processors, has a running time of $(2N + 2M + P + \max(M, P))$ for each training set vector. This is the first such implementation of the back propagation algorithm which completely parallelizes the entire computation of learning phase. The array has been implemented on an Annapolis FPGA based coprocessor and it achieves very favorable performance with range of 5 GOPS. The proposed new design targets Virtex boards. A description is given of the process of automatically deriving these high performance architectures using the systolic array design tool MMA_{ALPHA}. This makes it easy to specify the system in a very high level language (ALPHA) and to perform design exploration to obtain architectures whose performance is comparable to that obtained using hand optimized VHDL code.

Associative networks have a number of properties, including a rapid, compute efficient best-match and intrinsic fault tolerance, that make them ideal for many applications. However, large networks can be slow to emulate because of their storage and bandwidth requirements. Chapter 6 presents a simple but effective model of association and then discusses a performance analysis of the implementation this model on a single high-end PC workstation, a PC cluster, and FPGA hardware.

Chapter 7 describes the implementation of an artificial neural network in a reconfigurable parallel computer architecture using FPGA's, named Reconfigurable Orthogonal Memory Multiprocessor (REOMP), which uses p^2 memory

modules connected to p reconfigurable processors, in row access mode, and column access mode. REOMP is considered as an alternative model of the neural network neocognitron. The chapter consists of a description of the REOMP architecture, a the case study of alternative neocognitron mapping, and a performance performance analysis with systems systems consisting of 1 to 64 processors.

Chapter 8 presents an efficient architecture of Kohonen Self-Organizing Feature Map (SOFM) based on a new Frequency Adaptive Learning (FAL) algorithm which efficiently replaces the neighborhood adaptation function of the conventional SOFM. The proposed SOFM architecture is prototyped on Xilinx Virtex FPGA using the prototyping environment provided by XESS. A robust functional verification environment is developed for rapid prototype development. Various experimental results are given for the quantization of a 512 X 512 pixel color image.

Chapter 9 consists of another discussion of an implementation of SOFMs in reconfigurable hardware. Based on the universal rapid prototyping system, RAPTOR2000, a hardware accelerator for self-organizing feature maps has been developed. Using Xilinx Virtex-E FPGAs, RAPTOR2000 is capable of emulating hardware implementations with a complexity of more than 15 million system gates. RAPTOR2000 is linked to its host – a standard personal computer or workstation – via the PCI bus. A speed-up of up to 190 is achieved with five FPGA modules on the RAPTOR2000 system compared to a software implementation on a state of the art personal computer for typical applications of SOFMs.

Chapter 10 presents several hardware implementations of a standard Multi-Layer Perceptron (MLP) and a modified version called eXtended Multi-Layer Perceptron (XMLP). This extended version is an MLP-like feed-forward network with two-dimensional layers and configurable connection pathways. The discussion includes a description of hardware implementations have been developed and tested on an FPGA prototyping board and includes systems specifications using two different abstraction levels: register transfer level (VHDL) and a higher algorithmic-like level (Handel-C) as well as the exploitation of varying degrees of parallelism. The main test bed application addressed is speech recognition.

Chapter 11 describes the implementation of a systolic array for a non-linear predictor for image and video compression. The implementation is based on a multilayer perceptron with a hardware-friendly learning algorithm. It is shown that even with relatively modest FPGA devices, the architecture attains the speeds necessary for real-time training in video applications and enabling more typical applications to be added to the image compression processing

The final chapter consists of a retrospective look at the REMAP project, which was the construction of design, implementation, and use of large-scale

parallel architecture for neural-network applications. The chapter gives an overview of the computational requirements found in algorithms in general and motivates the use of regular processor arrays for the efficient execution of such algorithms. The architecture, following the SIMD principle (Single Instruction stream, Multiple Data streams), is described, as well as the mapping of some important and representative ANN algorithms. Implemented in FPGA, the system served as an architecture laboratory. Variations of the architecture are discussed, as well as scalability of fully synchronous SIMD architectures. The design principles of a VLSI-implemented successor of REMAP- β are described, and the paper concludes with a discussion of how the more powerful FPGA circuits of today could be used in a similar architecture.

AMOS R. OMONDI AND JAGATH C. RAJAPAKSE