Abstraction in FPGA Implementation of Neural Networks

ARİF SELÇUK ÖĞRENCİ
Electronics Engineering
Kadir Has University
Fatih, 34230-Istanbul
TURKEY
ogrenci@khas.edu.tr

Abstract: - A model for FPGA implementation of multilayer perceptron neural networks is presented. The model tries to incorporate object oriented design principles in the analysis, training, and design of components using hardware description languages. The synthesis will be based on the tools supplied by the FPGA vendors. The results indicate that the method can be utilized, and it can be further improved to create a general methodology that bridges the gap between hardware and software in embedded system design.

Key-Words: - hardware synthesis, software architecture

1 Introduction

Embedded systems are special purpose computing structures that process signals originating from the complex environment in which they are "embedded." In that way, they can control other actions in the system. In the simplest case, an embedded system may be implemented using a single dedicated microprocessor or FPGA (Field Programmable Gate Array) whereas complex systems such as a distributed network of processors may also exist. The embedded system is mostly a reactive system designed to communicate with its environment in performing a specific job. Hence, general purpose processor architectures are not well suited for embedded systems.

The following groups of constraints have to be satisfied by embedded systems:

a) Functional requirements
b) Quantitative constraints (speed, area, power, etc.)
c) Interaction with the environment (communication, reconfigurability, interface, etc.)

Therefore, there is a need for a serious methodology to satisfy the constraints and to optimize the system. The traditional methodology incorporates a top-down behavioral/functional analysis followed by a bottom-up synthesis based on components. The methodology can be applied with a high level of efficiency and reliability for complex digital systems due to the existence of advanced computer aided design tools that offer the use of hardware description and synthesis languages (VHDL, Verilog, SystemC, etc.) Usually, the analysis and the design (synthesis) are integrated in a way that those tools map behavioral/functional definitions onto component models present in their libraries. All the other constraints are then used as ingredients of the optimization process during synthesis.

Software development methodologies based on object oriented analysis and design (utilizing design patterns); have allowed a high level of modeling and abstraction within the last decade. The efficiency of software development has increased substantially. UML (Unified Modeling Language) has emerged as a standard language to be used in the design of complex software systems within the object oriented paradigm. It is generally accepted that UML can also be used in hardware implementations, and embedded systems can be considered as the natural candidates for such implementations [1]. However, there exist several fundamental problems in this area:

a) Even though the level of abstraction for hardware models approaches the level required for object oriented systems (due to use of UML, SystemC etc.), hardware is still considered as made up of components interconnected using wires. On the contrary, object oriented design is about objects and the communication between them. Hence, there is a need for a methodology that combines components with objects, and wires with communication.
b) It is not clear how the constraints to be satisfied by embedded systems; have to be described in object oriented methodology. The automatic mapping of requirements such as speed, area, etc. onto models or patterns, is still not possible.
c) The transformation of objects and patterns into hardware and RTL (Register Transfer Level) definitions is not clear.

Neural networks form an excellent example for embedded systems that need to be implemented as special purpose computing units rather than as software in general purpose processors. Especially, use of reconfigurable FPGA for neural networks allows the designer to have flexibility. Several FPGA based

implementations of neural networks have been reported in the literature recently [2-4] that employ different design methodologies. However, they also do not address the problems mentioned above. Most of the implementations in the literature are concerned mainly about the optimization of the architecture depending on the problem so that the design fits into appropriate FPGA units. There exists also work about forming a library of software modules using object-oriented design and programming which would allow users to deploy prototype neural networks in order to solve specific tasks [5-6].

The motivation for this work is that a unified design methodology can be developed to incorporate software development principles to embedded system design of multilayer perceptron neural networks (Fig. 1). This would require that neural networks would be considered at a higher, abstract level as software components, which then will be mapped to actual hardware components on FPGAs. In the following sections, we first give a formal definition of a neural network software architecture which will be used for the abstraction. Then, the essential problems of training and optimization will be mentioned, and the mapping of software components to hardware descriptions will be discussed. The paper will conclude by analyzing the results and by discussing future research topics.

2 Abstract Neural Network Model

An open and distributed software architecture for neural networks has been derived in [7], in which an object oriented methodology is put forward where the information model is based on the requirements derived from neural networks. The proposed information model for the software architecture is given in Fig. 2. This architecture will be utilized in the abstraction of the problem realizing a multilayer perceptron neural network of Fig. 1 in an FPGA. The main goal of the methodology is to use training and optimization software for higher levels of abstraction and using a synthesis tool for a specific FPGA at the hardware level.

The system flow will be as follows:

a) Train the neural network (topology will be selected based on user input) to determine weights
b) Quantize input and output according to the desired level of precision as supplied by the user
c) Choose a model (serial or parallel) implementation based on the performance requirements
d) Generate VHDL models of the building blocks at behavioral level, such as, multiplier, adder, and sigmoid block
e) Run the FPGA synthesis tool to map the design onto a specific device
f) If the performance is not satisfactory (timing and area) then go to step c) and choose another model; if all models have been used, then go to step e) and choose a larger FPGA.

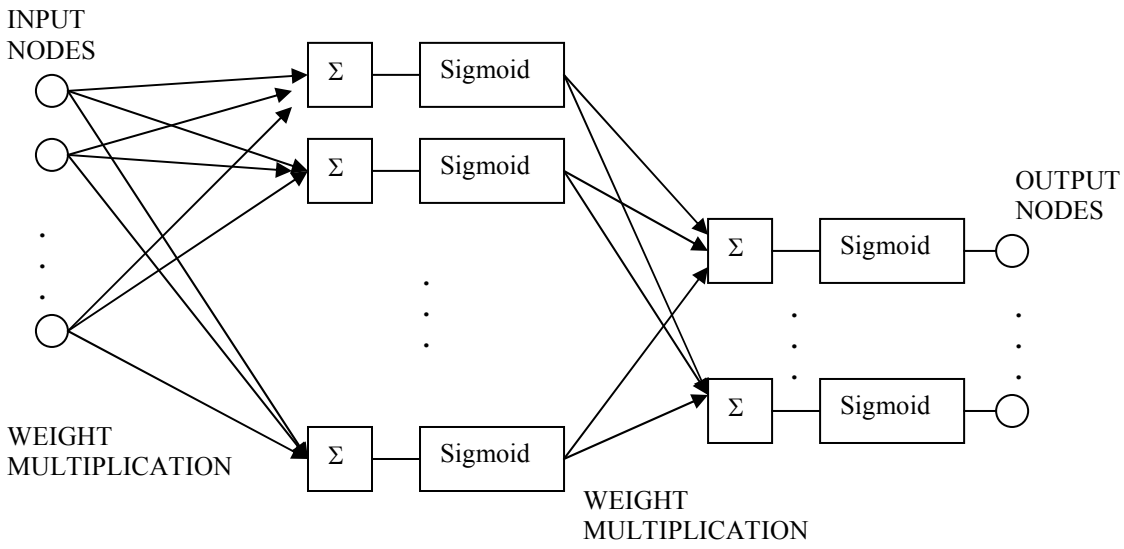

Fig. 1 Multilayer perceptron neural network model

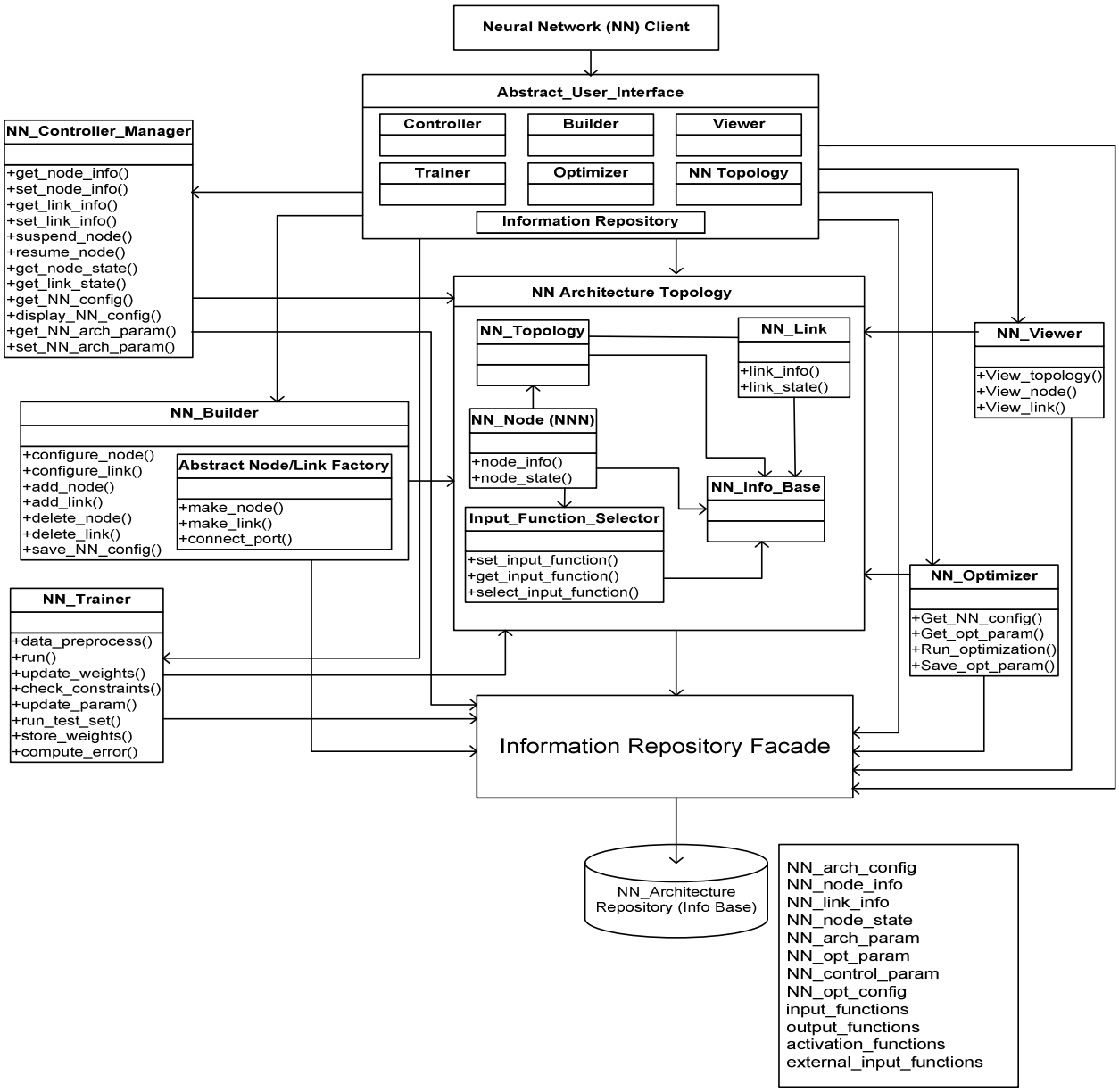

Fig. 2 Neural network software architecture information model

3 Example: Hardware Components of Neural Network

As a design example, a simple, nonlinear function approximation problem has been investigated. The function is the sin (x) where x ranges from 0 to $\pi/4$. The desired precision at the output has been set to be 1/64, so that the output needs to be quantized with 16 bits where 8 bits are used for the integer part and 8 bits are used for the floating part. Meanwhile the input has been quantized with 5 bits. A 1 x 8 x 1 structure has been employed, and the network has been trained using standard backpropagation for 10000 epochs. Then the weights and intermediate outputs have been quantized with 16 bits. Using the high level behavioral descriptions of the multiplier(s), adder(s), and the sigmoid blocks, a structural description of the neural network has been generated. The system has utilized a fast parallel multiplier using both Wallace Tree and Booth algorithms. On the other hand standard half-adder based adders have been utilized for the summation blocks. Finally, a quadratic approximation has been used for the sigmoid function. The VHDL code has been synthesized on a Xilinx Spartan family FPGA.

The synthesis results are given in Table 1. The simulations have verified that the neural network approximation has been successful, and the circuit can be mapped to a FPGA efficiently. The system was not capable of synthesizing the network on a smaller set of FPGA units.

Table 1. Output report of the synthesis tool

Resource	Used	Available	Utilization
IO's	96	176	55%
Function Generator	2073	2400	86%
CLB Slices	1037	1200	86%
Estimated Delay	267ns		

4 Conclusion and Future Work

In this paper, the possibility of employing an abstract software architecture model for the synthesis of neural networks in FPGA has been investigated. There is a lack of methods and tools for the full utilization of object oriented methods and software design patterns in the hardware implementation of embedded systems. Such a methodology would be very useful if the constraints of the system could be efficiently described within the software architecture. As an initial point towards that direction, a simple design example has been investigated and the results have been given. As future work, the speed and area requirements could be integrated into the architectural modeling so that the automatic generation of VHDL would consider macro models of building blocks and/or the available set of FPGA chips in the optimization process.

References:

[1] R. Damasevicius and V. Stuikys, Application of the object-oriented principles for hardware and embedded system design, *Integration, the VLSI Journal*, Vol. 38, 2004, pp. 309-339.

[2] V. Pandya, S. Areibi and M. Moussa, A Handel-C Implementation of the Back-Propagation Algorithm on FPGA, *Proceedings of the 2005 International Conference on Reconfigurable Computing and FPGAs (ReConFig 2005)*, 2005.

[3] S. Vitabile, V. Conti, F. Gennaro, and F. Sorbello, Efficient MLP Digital Implementation on FPGA, *Proceedings of the 2005 Euromicro Conference on Digital System Design (DSD 2005)*, 2005.

[4] D. Lettnin, A. Braun, M. Bogdan, J. Gerlach, and W. Rosenstiel, Synthesis of Embedded SystemC Design: A Case Study of Digital Neural Networks, *Proceedings of the Design Automation and Test in Europe Conference (DATE 2004)*, 2004.

[5] G. Valentini and F. Masulli, NEURObjects: An object-oriented library for neural network development, *Neurocomputing*, Vol. 48, 2002, pp. 623-646.

[6] T. P. Caudell et al, eLoom and Flatland: specification, simulation and visualization engines for the study of arbitrary hierarchical neural architectures, *Neural Networks*, Vol. 16, 2003, pp. 617-624.

[7] A. S. Ogrenci, T. Arsan and T. Saydam, An Open Software Architecture of Neural Networks: Neurosoft, *Proceedings of Software Engineering and Applications (SEA2004),* 2004.