UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

DIGITAL ARCHITECTURES FOR ADAPTIVE PROCESSING OF MEASUREMENT DATA

Andrea Boni, Dario Petri, Ivan Biasi

June 2004

Technical Report # DIT-04-061

Digital Architectures for Adaptive Processing of Measurement Data

Andrea Boni, Dario Petri, Ivan Biasi

DIT - University of Trento, Via Sommarive, 14, 38050 Trento Italy, andrea.boni@ing.unitn.it

Abstract- In this paper we describe the design of digital architectures suitable for the implementation of measurement data classification based on Support Vector Machines (SVMs). The performance of such architectures are then analyzed. The proposed approach can be applied for solving identification and inverse modelling problems, and for processing complex measurement data. Two very different case studies where real-time processing is of paramount importance are discussed: a nonlinear channel equalization and a high energy physics classification task.

I. Introduction

In the last few years the area of Machine Learning (ML) has gained the attention of the scientific community, especially when used to solve difficult industrial or scientific applications, such as fault recognition, digital signal processing, pattern recognition, etc. [4]. By using ML--algorithms a non--linear unknown relationship is estimated on the basis of a set of input/output measurements *Z*. Some examples are the well--known Artificial Neural Networks. Recently, new inductive paradigms have been proposed with success: the Kernel--Based Methods (KBMs) [9]. KBMs are studied in the context of the Statistical Learning Theory (SLT), formulated by Vapnik and Chervonenkis in the '70s [10], and are applied in many fields due to their robustness and the presence of a solid theoretical background. An example of KBMs are the well known Support Vector Machines (SVMs). Basically, an SVM induces an input/output relationship from i/o measurements *Z* by means of an optimization algorithm (OA), which consists in the resolution of a constrained quadratic problem, and generates a vector of parameters used to provide the output of a new input during the so called *estimation phase* (EP). Usually, in the EP the real--time processing is of paramount importance for the application, therefore the hardware implementation of KBMs has become an important area of research. After the first preliminary studies, several analog and digital implementations of KBMs for both OA and EP have been proposed [3], [6]. Here, we discuss the digital implementations on FPGA--based devices of the EP. Our aim is the implementation of a processing system for solving inverse modeling problems and for the classification of complex measurement data. As a case study, here we consider two different applications: a communication channel equalization, characterized by few parameters, and a scientific application, such as a tagging problem of High Energy Physics (HEPs) experiments, where a big amount of measures must be processed in real time. In this paper our aim is to describe the architectures, their characteristics and performances when used to solve the above problems.

In section II we briefly describe the problem formulation and the basic notation. In section III we provide a brief description of architectures, whilst in section IV we discuss simulation results and performance for the two considered applications. Finally in section V some conclusions are given.

II. Problem formulation and basic notation

A general inverse modelling problem is defined as follows: a discrete signal $u(n)=\pm 1$, acting as the input of a non linear discrete system, having unknown dynamics, must be estimated on the basis of the signal $x(n)$ observed at the output of the system itself. As the unknown system can introduce a correlation on the signals, several output samples are observed in order to provide an input signal estimate $\hat{u}(n-D)$, where D represents the intrinsic delay of the estimator. In particular such an estimate is given on the basis of the r-dimensional feature vector: $x_n^{(r)} = [x(n), x(n-1), \ldots, x(n-r+1)]^T$, where r is the number of samples used to provide a single estimate. In order to build such an estimator KBMs use a set of input/output samples, also called the training set: $Z = \{(x_i^{(r)}, u_i)\}_{i=1}^{N}$, N is the number of input measures. This is also the case of a general measurement system, where a set of measures each one composed by an r-attributes vector must be classified according to a given rule. Typical classification problem are in the field of pattern recognition, or the classification of High Energy Physics [1]. Thus, our problem can be defined in the following way: given a set of measures *Z*, obtained by an unknown measurement system, provide its "best" relationship, from a "generalization"

point of view. The SLT establishes the conditions under which such a relationship is the best [8]. SVMs are a special kind of KBMs, and their classification functions belong to the following class:

$$\hat{u}(n-D) = \mathrm{sgn}\left(\sum_{i \in SV} \alpha_i u_i K\left(x_i^{(r)}, x_n^{(r)}\right) + b \right) \tag{1}$$

that is a weighted summation of kernels $K(\ )$, where $SV = \{i : \alpha_i \neq 0; i = 1, \ldots, N\}$ is the index set of the so called Support Vectors, that are the only input samples important for the classification. Typical classes of Kernel functions are the linear, the Gaussian (having width σ^2), and the polynomial ones [9]. Let us detail in the following how (1) is induced from *Z* (the reader can find more details in [9],[10]). In general, the estimation function for a two class classification problem is defined by finding a separating hyperplane

$$\hat{u}(n-D) = \mathrm{sgn}\left(\left(w^{(r)}\right)^T \cdot x_n^{(r)} + b \right) \tag{2}$$

According to the SLT the best hyperplane is the one that maximizes the *margin*, defined as the maximum distance between the closest samples belonging to the two different classes of samples [10], and lying on the hyperplanes $\left(w^{(r)}\right)^T \cdot x_n^{(r)} + b = \pm 1$. These samples are the most important for the classification and are indicated as the Support Vectors (SVs). Such a margin is defined as $1/\left\|w^{(r)}\right\|$. Unfortunately, in real-world problems, the input samples are often no-linearly separable (it is impossible a classification of all the input measurements), thus one should map the input vector onto a new *feature space*, by using a non-linear function $\varphi(\ )$, where the separating hyperplane there exists. Moreover, for model complexity reasons, usually one prefers to accept some errors on the input measures *Z*, so a complexity parameter *C* is introduced to trade-off the generalization performance (the error on a new vector) and the error on *Z*. The final problem to be optimize is given by the following *primal*:

$$\begin{aligned} &\min_{w^{(r)}, b, \xi^{(r)}} \left\{ \frac{1}{2} \left\| w^{(r)} \right\|^2 + C \sum_{i=1}^{N} \xi_i \right\} \\ &u_i \left(\left(w^{(r)}\right)^T \cdot \varphi\left(x_i^{(r)}\right) + b \right) \geq 1 - \xi_i \qquad i = 1, \ldots, N \end{aligned} \tag{3}$$

Usually, for practical reason, one does not directly solve (3), but its *dual* obtained by introducing *N* Lagrange multipliers for each constrains, and by minimizing the Lagrangian function [9][10]:

$$\begin{aligned} &\min_{\alpha^{(N)}} \left\{ \left(\alpha^{(N)}\right)^T Q \alpha^{(N)} - 1^T \alpha^{(N)} \right\} \\ &0 \leq \alpha^{(N)} \leq C \qquad \sum_{i=1}^{N} u_i \alpha_i = 0 \end{aligned} \tag{4}$$

Furthermore, it is easy to see that $w^{(r)} = \sum_{i \in SV} \alpha_i u_i \varphi\left(x_i^{(r)}\right)$, whilst $q_{ij} = \varphi\left(x_i^{(r)}\right) \cdot \varphi\left(x_j^{(r)}\right)$. Thus, instead of using dot products of the form $\varphi\left(x_i^{(r)}\right) \cdot \varphi\left(x_j^{(r)}\right)$, one can introduce kernel functions $K(\ )$ realizing dot-products in the feature space (in the mathematical literature this is known as the Reproducing Kernel Hilbert Space). Thus, the final estimation function is given by (1).

III. System Architectures

In order to implement eq. (1), we designed and tested several FPGA based architectures using a Xilinx Virtex XC2V1000 [11]. Hardware platforms suitable for the execution of the tasks described in the previous section are systems able to change at run-time their configuration in order to carry out different processing algorithms. Our choice is motivated by the versatility offered by the emerging FPGA technologies, which permit one to use reconfigurable hardware with embedded block RAMs, multipliers, hard-processors (see for example the Xilinx Virtex II-Pro) and advanced soft-microprocessors cores (see for example the Xilinx Pico and Micro Blaze). Such hardware platforms allows high-level hardware/software co-synthesis of complete Systems-on-Chip (SoC) to implement complex digital signal processing algorithms like the SVM [4]. Examples of FPGA implementation of neural hardware are also reported in the literature [7].

The advantage of our proposal consists in the full digital design implementation. In fact the high level description of the design (VHDL), allows one to easily change the implementation parameters in order to fit the requirements of the application to be solved. In particular, we designed several versions for the implementation of the Gaussian Kernel by using both a special purpose module based on the CORDIC algorithm [2] and the embedded Block RAMs (BRAMs) of the FPGA, used in order to map a

simple Look Up Table (LUT). These implementations provide different performance in terms of efficiency (such as the classification error), speed and area utilization.

In [4] we discussed the performance of a SVM-based reconfigurable architecture for solving an inverse modeling problem, in particular we focused our attention on the design of the SVM model. Here we provide details on the digital architecture. The general block-scheme of such architecture is provided in Figure 1. A Xilinx MicroBlaze [11] microprocessor acts as the supervisor of the System, connecting the on-chip memory, which stores the alphas, the OFF-chip memory, used when the number of the SVs exceeds the available on-chip memory, the UART for I/O communication and the KTRON core, implementing eq. (1). The MicroBlaze is a 32-bit Harvard Bus RISC architecture, of 900-cells size working at the speed of 125 MHz, and having 32 general purpose registers, with 3 operand instruction format, and several special purpose buses, as reported in Figure 1. As a development platform we used the Xilinx Embedded Development Kit (EDK), which permits one to easily integrate C/VHDL code. In practice, the MicroBlaze is programmed using C code, whilst the KTRON is described by using the VHDL code.

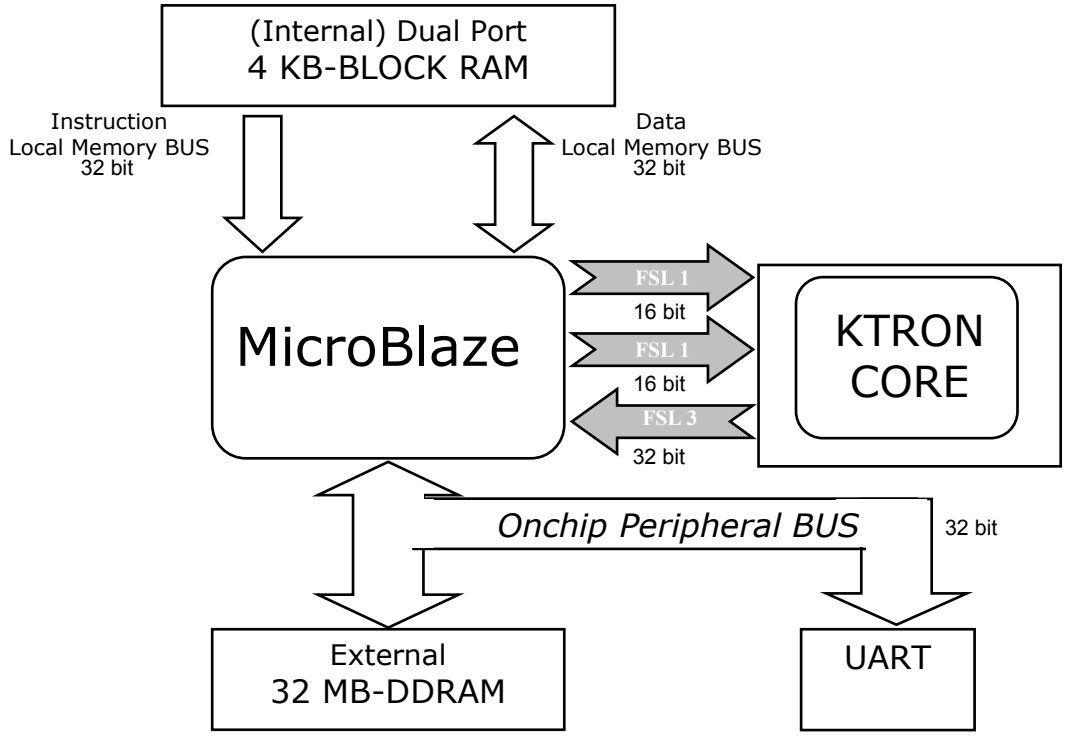

Figure 1. General Architecture

The basic structure of the KTRON core is showed in Figure 2. K_type_RAM is a simple flip-flop containing a flag indicating which of the two kernels is used. Ktron_drive contains the RAMs for both the set of support vectors (when the on-chip memory is used) and the input $x^{(r)}$ to be processed. Pre_kernel is the first computation-unit. It computes an inner product or a squared norm according to the value in K_type_RAM. The division by $2\sigma^2$ can be implemented in two different way: 1) by approximating $2\sigma^2$ to the nearest power of 2 and thus only a simple shift register is necessary for the computation; 2) by getting and storing in a register the value $\gamma = 1/2\sigma^2$, and by using the Virtex II embedded fast 18x18MULT multiplier. The latter solution is characterized by more precision; the only drawback is the use of an important resource like the 18x18MULT. Kernel is the second computational unit; if a Gaussian kernel is selected, it computes the exponential function. We used two different implementations: in a first version we mapped the exponential function onto a simple Look-Up-Table (LUT) by using the embedded block RAMs of the Virtex II; in a second version we designed a VHDL

CORDIC-based module [2], partially modified in order to extend the convergence range [6]. The CORDIC algorithm (COordinate Rotational DIgital Computer) is an elegant method to iteratively compute complex functions by using only adders and shift registers. Its main advantage consists in the fact that no multipliers or memory are required, thus allowing exact mathematical computations by using few FPGA slices.
The last computation unit is out_mac, a simple multiply-and-accumulate block. Ktron_ctrl is the main control unit of the whole system; it decodes the received commands and manages control signals connected to input data.

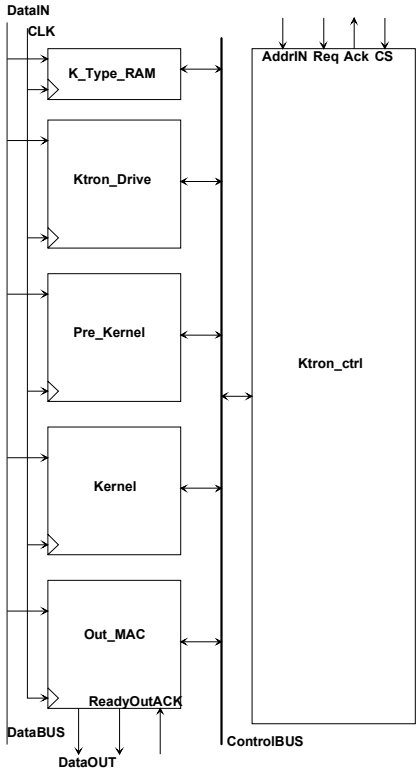

Figure 2. The KTRON architecture.

IV. Applications, Simulation results and performance

The first of our two applications is a typical equalization problem (EQP) of a transmission channel, characterized by few measures, and where the non linear effects of the channel are modelled as a FIR filter [3],[5]. We used a model as defined in [5], by using r=2 and different sample data (N=32,64,128). The second application is an HEP class of experiments for top/anti-top ($t\bar{t}$) quark couples detection. Data are composed by a two-classes set of samples organized as follows: a first half, used for training comes from the "RUN I" experiments at the Collider Detector at Fermilab (CDF), and the other class comes from a simulation of the $t\bar{t}$ generation event. A real time hardware is designed in order to fast classify the generation events and neglect the real-world background events [1]. Each sample is composed by r=8 measurements representing the main characteristics of the experiment. We used N=4190 training samples and 4190 test samples. Model selection to find the hyper-parameters C and $2\sigma^2$ has been carried out by using bootstraps techniques [1]. These, are two very different application. The former, is characterized by few parameters. In particular we obtained a results characterized by less then 20 SVs, whilst the latter is characterized by 2200 SVs. Details on classification and model selection can be found in references [1],[3],[5], here we report the performance of the architectures.

The architectures described in the previous section have been implemented on a Xilinx Virtex II V2MB1000 MEMEC development board by using the Xilinx ISE 6.2i as development tool, and the Xilinx XST as a synthesis tool [11]. Virtex -IIV2MB1000 provides a complete platform and allows one to build complex designs and applications and full Systems-on-Chip with partial reconfiguration. The system Board includes a 16MBx16 DDR memory, the 1 million gate Xilinx Virtex II FPGA device is

composed by a matrix of 40x32 CLBs, 40 MULT18x18s multipliers and 40 BRAMs [11]. These features allowed us to quickly and effectively meet our design requirements.

In the EQP case we designed both a simple 1024x16 LUT and a 12-bit CORDIC module to map the exponential function. The results on the classification rate are comparable, whilst the results on the synthesis are quite different, as reported in table 1. In practice, the main difference consists in the clock speed, that is 100 MHz and 50 MHz for the LUT and CORDIC case respectively.

	KTRON-LUT	KTRON-CORDIC
Slices	5.1%	8.7%
BRAMs	10%	7.5%
MULT18x18s	5%	7.5%
speed	100 MHz	50 MHz

Table 1. Synthesis results for the transmission channel equalization problem.

In the HEP case, in order to fulfil classification rate requirements, we designed a 14-bit input LUT, by obtaining a classification error of 24.1%, and a 14-bit CORDIC module that allowed us to obtain a classification error of 23.5%. The results of the synthesis are summarized in Table 2

	KTRON-LUT	KTRON-CORDIC
Slices	5.5%	10%
BRAMs	95%	20%
MULT18x18s	5%	7.5%
speed	100 MHz	50 MHz

Table 2. Synthesis results for the HEP problem.

In practice, in this case almost all the available BRAMs are used in order to store the SVs and LUT. This means that few memory is available for other modules (such as the microprocessor, for example). The CORDIC requires less memory, but works with a lower clock speed (50 MHz instead of 100 MHz), and is characterized by a slightly better classification error.

V. Conclusions and discussion

This work deals with the problem of fast classify sets of measurement data from two different real-world applications. Our results show the efficiency and robustness of the proposed method, and the optimal performance of the architectures implementing a Kernel-based classification algorithm. In particular, several different implementation are proposed. The user can choose the preferred one on the basis of speed/area trade-off criteria.

Our design and implementation illustrates a typical example of the advantages achieved by using FPGA. Fast development times and reduced efforts are not the only merit of the ‘soft’ approach in implementing advanced chips for adaptive processing of measurement data. The generic nature of the FPGA building blocks enables the integration of the arithmetic core of the neural processor, its logic control and the general purpose processor. The programmable nature of FPGAs opens up opportunities for exploring the architecture and implementing a choice of interfacing protocols at negligible costs. Moreover, VHDL design provides a typical example of design re-use and the merits of design portability.

Future work would concentrate on the advantages of integrating a host RISC processor, a module for SVM learning and the KTRON, into the FPGA, so as to complete a stand-alone system based on the neural approach to solve a variety of application problems. This level of integration has become even simpler than ever with the introduction of advanced techniques for dynamic partial reconfiguration which allow one to fully exploit the computational power and the area of the FPGA

chips [11]. High-density FPGAs also make it possible to implement larger SVMs by incorporating more than one instance of KTRON on a single device to speed up the computation.

References

[1] Amerio S., Anguita D., Lazzizzera I., Ridella S., Rivieccio F. and Zunino R., "Model Selection in Top Quark Tagging with a Support Vector Classifier", *International Joint Conference on Neural Networks*, Budapest July 2004.

[2] Andraka R., "A survey of CORDIC algorithms for FPGAs" *FPGA '98*. Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field programmable gate arrays, Feb. 22-24, Monterey, CA. pp. 191-200, 1998.

[3] Anguita, D., Boni, A., and Ridella, S.: "A Digital Architecture for Support Vector Machines: Theory, algorithm and FPGA Implementation," *IEEE Trans. on Neural Networks*, vol. 14, n. 5, p. 993-1009, 2003.

[4] Bishop, C., *Neural Networks for Pattern Recognition*, Clarendon Press., Oxford, 1995.

[5] Boni, A., Pianegiani, F., Petri, D., "Inverse Modeling with SVMs-based Dynamically Reconfigurable Systems", *IEEE Instr. and Measurement Technology Conference*, Como, Italy, May 18-20, 2004.

[6] Genov R. and Cauwenberghs G.: "Kerneltron: Support Vector Machine in Silicon," *IEEE Trans. on Neural Networks*, v. 14, n. 5, p. 1426-1434, 2003.

[7] Hu X., Haber R.G., and Bass S.c., "Expanding the Range of Convergence of the CORDIC Algorithm", *IEEE Transactions on Computers*, Vol. 40, No. 1, 1991.

[8] McBader S., Lee P., and A. Sartori, "The Impact of Modern FPGA Architectures on Neural Hardware: a Case Study of the TOTEM Neural Processor", *International Joint Conference on Neural Networks*, Budapest July 2004.

[9] B.Schölkopf, A.Smola, *Learning with Kernels*,The MIT Press, 2002.

[10] Vapnik, V.N.: *Statistical Learning Theory*' John Wiley & Sons, NY, USA, 1998.

[11] Xilinx website, www.xilinx.com