Neural controller implementation in embedded system with use of FPGA coprocessor

Karol Gugała, Aleksandra Świetlicka, Krzysztof Kolanowski, Igor Karoń,
Mateusz Majchrzycki and Andrzej Rybarczyk
Poznan University of Technology
Faculty of Computing
Chair of Computer Engineering
60-965 Poznań, ul. Piotrowo 3A
Poland
Email: {karol.gugala, aleksandra.swietlicka, krzysztof.kolanowski,
mateusz.majchrzycki, andrzej.rybarczyk}@put.poznan.pl,
igor.karon@doctorate.put.poznan.pl

***Abstract*—In this paper we propose implementation of neural control system as embedded system with coprocessor in extensible processing platform.**

I. Introduction

Neural control systems have proven their fitness especially in nonlinear systems. They are widely used in robotics in tasks like path finding [6], or improving fault tolerance of classic controllers [5]. Main problem with that kind of controllers is efficient implementation of a neural network calculations allowing to work of the controller in a real time.

FPGA implementation of a neural coprocessors can efficiently speed up neural calculations in the whole system [1], [2]. On the other hand handling of such aspects as communication protocols and interactions with users, e.g. providing User Interface (UI), are quite problematic using only the FPGA cores.

Embedded System running complex OS such as Linux on a dedicated hardware platform enables easy implementation of user interfaces or communication protocols handling. Combination of complex operating system running on a fast CPU and neural coprocessor implemented in reconfigurable logic can give a lot of benefits in work speed and ease of implementation of the whole system.

Extensible Processing Platform such as Xilinx Zynq-7000 chip unites advantages of the embedded system controlled by complex operating system and FPGA configurable logic allowing to implement custom hardware cores [7].

In this paper we propose implementation of neural motor controller in the extensible processing platform, with use of FPGA neural coprocessor and embedded Linux application. In the next sections components of the system are presented.

II. System architecture

Proposed system consists of FPGA implementation of neural coprocessor and embedded Linux based system running control application. System structure is depicted in Fig.1.

Control application is running in embedded Linux user space, and communicate through the system drivers with neural coprocessor and motor driver implemented in FPGA part of the system.

Motor driver and coprocessor are attached to AXI bus [4] and are mapped into I/O memory space in Linux system [3]. Drivers of those cores share device files in the $/dev$ directory of the Linux system and series of *ioctls* enabling control and data exchange between control application and FPGA cores.

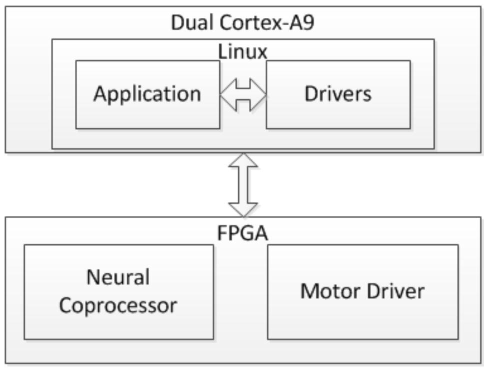

Fig. 1. System structure

III. Neural Coprocessor

Neural coprocessor used in the system is based on the one described in [1]. It was extended to support 32 bit data bus, also AXI controller was added to allow connection with the rest of the system. Structure of this core is presented in Fig. 2.

Coprocessor logic is an aritmetic core detailed in [1]. Data exchange is made through *Local Memory* module. System writes input and reads output data from coprocessor through AXI bus. Control writes and state reads of the coprocessor are made with control registers module.

IV. Motor driver

Motor driver is a hardware core enabling connection, through power amplifier, of the motor to the control system. The structure of this core is presented in Fig. 3.

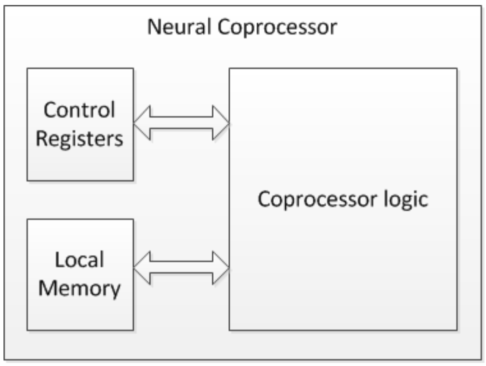

Fig. 2. Neural coprocessor structure

The *Encoder decoder* submodule is responsible for decoding signals from rotary encoder to information about speed and direction of motor rotation. The *PWM module* is responsible for generating PWM signal with given duty cycle. All data exchange and control commands are passed through the *Control Registers* submodule.

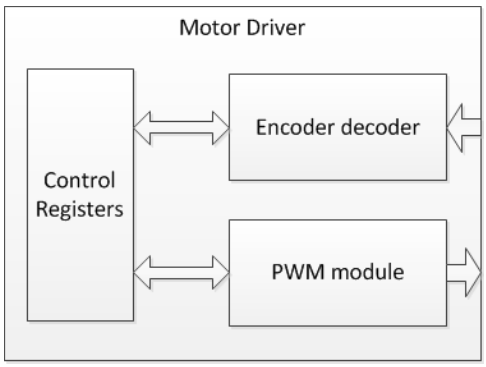

Fig. 3. Motor driver structure

V. Linux embedded system integration

Integration of the system as embedded system, controlled by Linux OS, required implementation of a device drivers of the *Neural Coprocessor* and the *Motor Driver* modules. Details about those drivers are presented in the following subsections:

A. Neural Coprocessor *device driver*

Device driver of the *Neural Coprocessor* module provides access to the *Local Memory*, and *Control Registers* submodules.

Access to the *Local Memory* is assured through block device file provided by the device driver in */dev* directory in Linux root file system [3]. To control work of the module the driver shares two *ioctls*: *COPROCESSOR_START_CALCULATION* and *COPROCESSOR_STOP_CALCULATION*.

The user space application can use those *ioctls* and block device to depute neural calculations to coprocessor, but it must assure correctness of input data itself - otherwise behavior of the coprocessor is unpredictable. Third *ioctl* - *COPROCESSOR_CALCULATION_STATUS* is used to read current status of the coprocessor.

TABLE I
Motor Driver DEVICE DRIVER *ioctls*

ioctl	Description
MOTOR_READ_SPEED	Returns current speed of the motor (read from the *Encoder Decoder* submodule)
MOTOR_READ_DIRECTION	Returns current direction of the motor (read from the *Encoder Decoder* submodule
MOTOR_READ_PWM_FREQ	Returns current PWM signal frequency
MOTOR_READ_PWM_DUTY	Returns current PWM signal duty cycle
MOTOR_SET_PWM_DUTY	Sets PWM signal duty cycle
MOTOR_SET_PWM_FREQ	Sets PWM signal frequency
MOTOR_START_PWM	Enables PWM signal generation
MOTOR_STOP_PWM	Disables PWM signal generation

B. Motor Driver *device driver*

The *Motor Driver* device driver is simpler than described above. It provides only access to the *Control Registers* submodule via *ioctls* detailed in Table I.

VI. Conclusion

We have presented an example implementation of the neural controller as an embedded system in the extensible processing platform. This solution combines benefits of fast coprocessor calculations and easiness of implementation of the control application in complex operating system.

References

[1] A. Rybarczyk and M. Szulc, *The concept of a microcontroller with neural-matrix coprocessor for control systems that exploits reconfigurable FPGAs*, Robot Motion and Control, 2002. RoMoCo '02. Proceedings of the Third International Workshop on., pp: 123-132, 2002

[2] M. Bogdan, H. Speckmann and W. Rosentiel, *Kobold: A Neural Coprocessor for Backpropagation with Online Learning*, Microelectronics for Neural Networks and Fuzzy Systems, 1994., Proceedings of the Fourth International Conference on, pp: 110 - 117, 1994

[3] J. Corbet, A. Rubini, and G. Kroah-Hartman, *Linux Device Drivers, Third Edition*, O'Reilly, 2005

[4] Xilinx Inc., *Zynq-7000 All Programmable SoC Technical Reference Manual*, 2012, http://www.xilinx.com

[5] N.S. Naikal, R. Panikkar, A.A. Pashilkar, and R. Nagaraj, *Improved Fault Tolerance for Autolanding Using Adaptive Backstepping Neural Controller*, Control Applications, 2007. CCA 2007. IEEE International Conference on, pp: 1203 - 1208, 2007

[6] A.S. Al-Araji, M.F. Abbod, H.S. Al-Raweshidy, *Neural autopilot predictive controller for nonholonomic wheeled mobile robot based on a pre-assigned posture identifier in the presence of disturbances*, Control, Instrumentation and Automation (ICCIA), 2011 2nd International Conference on, pp: 326-331, 2011

[7] K. DeHaven, *EPPs: The Ideal Solution for a Wide Range of Embedded Systems*, Xilinx 2012, http://www.xilinx.com