Modular Signal Processing Unit for Motion Control Applications Based on System-on-Chip with FPGA

Vlastimil Šetka*, Ondřej Ježek*, Rihards Novickis†
*NTIS – New Technologies for the Information Society Research Center,
Faculty of Applied Sciences, University of West Bohemia, Technická 8, Pilsen, Czech Republic
E-mail: {setka,ojezek}@ntis.zcu.cz
†EDI – Institute of Electronics and Computer Science,
14 Dzerbenes St., LV-1006, Riga, Latvia
E-mail: rihards.novickis@edi.lv

***Abstract*—Motion control systems with distributed architecture where multiple input/output devices are connected to the upper layer controller by fast digital communication (fieldbus) became an industrial standard. This paper presents design of a modular input/output device which can process signals from multiple sensors, drive multiple actuators and act as a Slave or Master node in EtherCAT fieldbus network.**

User-defined algorithms can be easily implemented to pre-process input signals, combine multiple signals or close local control loops with extremely high sampling rates which makes the difference to standard off-the-shelf solutions. To meet these requirements and simplify hardware design, our device is based on System-on-Chip with both programmable logic (FPGA) and classic processor (CPU) ARM cores. Data processing including user algorithms can be done entirely in FPGA which provides very low latency and no jitter, and also on CPU for more complex computations with advantage of tight integration between FPGA and CPU. In this paper we provide description of hardware design, system architecture and typical applications.

***Index Terms*—FPGA, System-on-Chip, Industrial I/O Device, Motion Control, Real-time Systems, EtherCAT, Asymmetric Multiprocessing**

I. Introduction

The paper is organized as follows. Section II discusses our solution next to a standard commercially available platforms and defines the key requirements. Section III deals with selection of suitable hardware platform for device implementation. Section IV describes the system architecture in detail. Modular hardware concept, FPGA subsystem, Management subsystem and Real-time application subsystem design are elaborated. Section V brings conclusions and directions for future work.

II. Problem Statement, Use Cases, Requirements

Modern motion control systems are designed in a distributed manner. Central controller (PLC, PC, Motion Controller – terminology depends on vendor) is managing motion trajectory planning and higher level MIMO control loops (technology, position, sometimes velocity). Sensors, actuators or devices like servo drives (which integrates servo motor power stage, encoder interface and torque/velocity/position control loops) are connected to the central controller by means of high-speed real-time data communication interface referred to as a fieldbus. The most widespread fieldbus technologies for high-end motion control are based on Ethernet standard – EtherCAT [11], PROFINET [12] or POWERLINK [13].

Examples of such platforms are these provided by Beckhoff (TwinCAT, EtherCAT), B&R (Automation Runtime, POWERLINK), Siemens (SIMOTION, PROFINET) or Omron (Sysmac NX/NY, EtherCAT) corporations. Distributed modular I/O systems for each fieldbus with a wide variety of field interfaces – analog and digital, encoders, communication – are also available as a standard off-the-shelf products.

This paper deals with design of a modular signal processing unit which can be integrated with standard motion control systems as outlined above or with open platforms like the one being developed in the I-MECH project [1] [10]. It is intended for applications which requirements go beyond standard I/O systems. Although EtherCAT is able to run with communication cycle times under 50 µs [5] and central controllers of platforms mentioned above are usualy based on multicore x86-64 CPUs which provides large computing performance, I-MECH applications proved there are situations which can not be solved with standard I/O technology and fully centralized computing. Here we provide a few examples on this topic:

- *Real-world implementation limits* – EtherCAT itself can handle very low cycle times, but practically not within larger network or with larger data payloads. Also not many commercial devices like servo drives are designed for fieldbus cycle times under 125 µs.
- *Latency sensitive sensor fusion and signal conversion* – Typical I-MECH use-case is input from multiple encoders with analog sin/cos inerface to be sampled at 100 kSps to 1 MSps rate and processed to a new "virtual sensor" signal (based on machine kinematics or signal conditions) for motor commutation in external servo drive or other external system. Latency introduced by transmission of such signals over fieldbus would break the control performance.
- *Local feedback loops critical on latency or reliability* – Feedback loops with very high sampling rates (above 20 kHz, i.e. 50 µs cycle) and total input to output latency below one sampling period can be hardly closed in central controller over fieldbus under real-world conditions [6],

[8]. Another case is when some control loops are required to run non-stop even if central controller is not available due to algorithm reconfiguration or fieldbus failure.
- *Application-level customization* – The functions described in the two points above are usually specific for each machine or application. Thus these functions needs to be defined or modified – customized – by application engineer without assistance of platform vendor.

There are a few commercial platforms which can solve this kind of tasks like NI CompactRIO [14] or Speedgoat [15] which can be also integrated with Ethernet-based fieldbus systems. But these are targeted more on prototyping or simulation scenarios. Compared to that, our solution is designed with cost-effectivity strongly in mind as it is intended for deployment to machines produced in medium series.

Thanks to the best performance characteristics [7] and fully open specification, EtherCAT fieldbus was selected as a basis for I-MECH Reference Platform.

Based on the facts above and the I-MECH landscape which details are out of this paper scope, basic requirements on the proposed device can be defined:

- Modular architecture which allows tailored combination of input/output interfaces for each application and easy design of a new interface modules.
- Typical input signals: analog sensors sampled up to 40 kSps, encoders with analog sin/cos output sampled at 1 MSps / quadrature output (S0S90) / BiSS-C [16] digital communication protocol.
- Typical output signal: emulated encoder with quadrature or BiSS-C slave output, analog sampled up to 40 kSps.
- Integration of wireless communication interfaces with proprietary protocol based on various chipsets.
- EtherCAT device (slave) interface capable at least 20 kHz cycle with processing latency as low as possible. Support of other fieldbuses like PROFINET is an advantage.
- Full time synchronization from EtherCAT Distributed Clock reference to all input/output sampling with sub-microsecond precision.
- Easy deployment of customized data processing tasks.
- Optional use as a stand-alone controller with local I/O and remote I/O over embedded EtherCAT master interface.
- Utilisation of standard operating system software would be an advantage for management functions like web server or firmware updates. Linux with RT (Real-Time) extensions could be a good choice.
- Industrial-grade design, ready to mount into machine control cabinet, with respect to typical environment conditions, industrial automation standards and best-practices.

III. Hardware Platform Selection

Many requirements outlined above clearly lead to utilisation of FPGA programmable logic for the following features:

- Interfacing of ADCs with very high sampling rates (1 MSps), processing of quadrature encoder signals.
- Specialized communication protocols – like BiSS-C.
- Synthesis of virtual sensor signals (quadrature encoder).
- EtherCAT slave controller with tight integration to the rest of system – implementation directly on FPGA provides the lowest possible data latency.
- Precise control over timing and time synchronization of all components.

Other requirements lead to utilisation of a standard CPU, performance class of ARM Cortex -A or -R, preferably with multiple cores, with hundreds MB of RAM, tightly integrated with the FPGA.

Optimal combination of FPGA and CPU corresponding to our requirements in a single chip is represented by System-on-Chip (SoC) devices available from both major FPGA vendors – Intel [17] and Xilinx [18] – in several performance classes. These are highly suitable for industrial control applications [2] [3] [4]. Due to a typical SoC hardware design complexity and high-frequency design issues (e.g. DDR3 RAM) it is out of our scope to design a complete SoC board from scratch. Also many components needed would be hardly available in small quantities. To save development effort, a few vendors provide so called System-on-Module (SoM) boards which integrates SoC device with all typical peripherals like power supplies, clock, RAM, Flash, Ethernet and USB PHYs in the form of compact module which can be easily integrated into custom design.

In-depth market research was performed to compare System-on-Modules from multiple vendors. Finally, the Mercury platform from Enclustra company [19] was chosen. The most important feature of this platform is compatibility of module interface across Intel and Xilinx and across different SoC classes. This allow us to design single type of carrier board which can be freely combined with multiple types of SoC devices from both vendors. Selected System-on-Modules and their features are summarized in the table I. This is very important as FPGA vendor (and related tooling) preference is often strongly enforced by corporate policy and know-how. Photo of such module with Intel Cyclone V series SoC (Enclustra SA1) is in the figure 1.

IV. System Architecture and Design

A. Modular Hardware Concept

Proposed platform is intended to be used in a wide range of applications which differs in types and number of required I/O interfaces. User needs to choose suitable set of I/O modules for particular application. Or even use the very minimal set of modules, potentially tailor-made for the application, when there is no need for rich I/Os but very limited space constraints.

Based on analysis of I-MECH project applications and feature requests defined by application owners for future applications, hardware architecture consisting of following components – see also diagram in the figure 2 – arised:

CPU (Central Processing Unit) Board:

- System-on-Module board - with System-on-Chip, RAM, Flash, and basic peripherals.
- Power input and main DC/DC power supply converter.

TABLE I
FEATURE COMPARISON OF SELECTED SYSTEM-ON-MODULE BOARDS

SoM type	**Mercury SA1**	**Mercury+ AA1**	**Mercury ZX5**	**Mercury XU5**
SoC vendor	Intel		Xilinx	
SoC class	low-cost	mid-range	low-cost	low-cost - mid-range
SoC series	Cyclone V	Arria 10	Zynq-7000	Zynq UltraScale+
CPU cores	2 × ARM Cortex A9	2 × ARM Cortex A9	2 × ARM Cortex A9	4 × ARM Cortex A53 2 × ARM Cortex R5
CPU clock	600 - 800 MHz	1000 - 1500 MHz	667 - 1000 MHz	A53: 1200 - 1500 MHz R5: 500 - 600 MHz
CPU RAM size	1024 MB	2048 - 4096 MB + ECC	1024 MB	2048 - 8192 MB + ECC
CPU RAM speed	3200 MB/s	7464 - 9600 MB/s	4264 - 5333 MB/s	19200 MB/s
Onboard flash	64 MB QSPI	64 MB QSPI, 16 GB eMMC	64 MB QSPI, 512 MB NAND	64 MB QSPI, 16 GB eMMC
CPU peripherals	2 × Gb ETH, 2 × CAN 2 × USB 2.0	3 × Gb ETH 2 × USB 2.0	2 × Gb ETH, 2 × CAN 2 × USB 2.0	4 × Gb ETH, 2 × CAN 2 × USB 2.0 / 3.0 Mali GPU, DisplayPort x2 PCIe Gen2 x4, SATA 3.1 x2
Onboard PHYs	1 × Gb ETH, 1 × USB 2.0	1 × Gb ETH, 1 × USB 2.0 1 × USB 3.0 device	1 × Gb ETH, 1 × USB 2.0	2 × Gb ETH (on PS, on PL) 2 × USB 2.0
FPGA capacity	85k - 110k LEs	270k - 480k LEs	74k - 125k LEs	103k - 256k LEs
FPGA speed	cca 150 MHz	cca 300 MHz	cca 150 MHz	cca 350 MHz
FPGA RAM size	-	-	-	512 - 2048 MB + ECC
FPGA RAM speed	-	-	-	4266 - 4800 MB/s
FPGA transceivers	6 × 3 Gb/s PCIe Gen1 x4	12 × 10.3 / 12 Gb/s PCIe Gen3 x8	4 × 6.25 - 6.6 Gb/s PCIe Gen2 x4	none - 4 × 12.5 Gb/s PCIe Gen3 x4
Dimensions	56 mm × 54 mm	74 mm × 54 mm	56 mm × 54 mm	56 mm × 54 mm
Unit price from	181 EUR, at 30 pcs	417 EUR, at 30 pcs	252 EUR, at 30 pcs	305 EUR, at 30 pcs

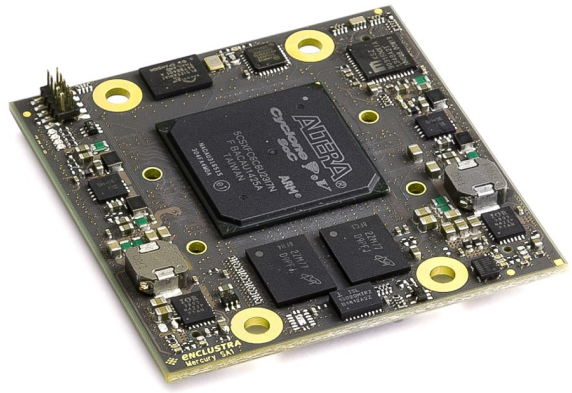

Fig. 1. System-on-module board with Intel Cyclone V SoC [19].

- Ethernet connectors and Ethernet PHYs – for EtherCAT, connected to the FPGA, and for other services, connected to the CPU.
- Service console interface on micro-USB connector.
- Basic diagnostics interface (LEDs, display, mode switch).
- MicroSD card slot.

Single variant of CPU Board for all selected System-on-Module types (see table I) was designed, available pin count and pinout compatibility was verified.

Submodule Boards:

Can be mounted directly on CPU Board to provide various additional interfaces which do not need so many signals and large space like standard I/O Cards (see below). Due to limited space on CPU Board and differences in interfaces available at individual SoC variants (for example fast serial transceivers and functions like PCIe, SATA, USB 3.0), these interfaces are routed to Submodule Boards space and provides an additional layer of flexibility. Multiple types of I/O Boards are already designed to cover typical basic interfaces provided by SoCs:

- USB 2.0 Host.
- CAN bus (1 or 2 channels).
- RS-422 / RS-485 interface (1 to 3 channels).

Backplane Board:

Allows connection of multiple (up to 6) I/O Card Boards to the system. Provides passive distribution of total 72 dedicated FPGA signals from CPU Board to individual I/O Card Boards – 12 signals per card in case of 6-Slot Backplane, 36 signals per card in case of 2-Slot Backplane. Multiple power supply rails, service and diagnostics signals are also available at each Card Slot. Two types of Backplane Board with 2 and 6 Slots was designed.

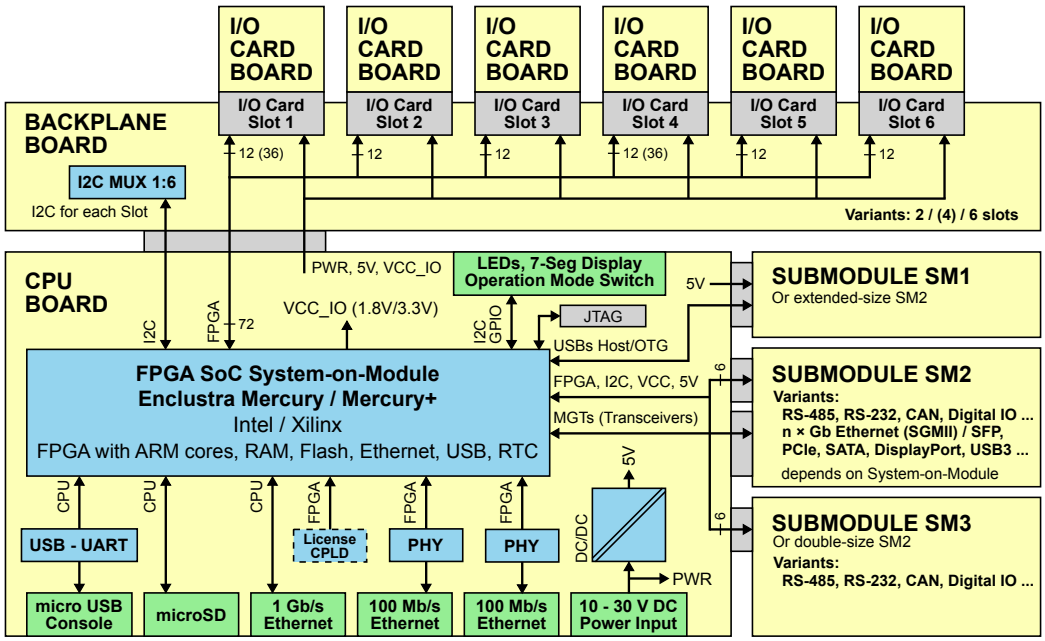

Fig. 2. Proposed modular hardware structure diagram.

I/O (Input / Output) Card Boards:

Provides interfaces to external analog, digital and communication signals of various types. Typical I/O Card contains physical inputs (connectors, terminals), diagnostic indicators, input protection, signal conditioning, conversion and electrical isolation to logic signals suitable for direct connection to the FPGA. Multiple types of I/O Boards are designed to cover requirements for all types of signals, for example:

- Universal Digital In/Out, IEC 61131-2, 16 channels.
- Analog In, 8 channels, ±10 V / 0-21 mA, 24-bit Sigma-Delta ADCs, up to 40 kSps sampling rate.
- Analog Out, 4 channels, ±10 V / ±24 mA, 16-bit DAC, 5 µs settling time.
- Sin/Cos Encoder Input, 2 × 2 channels, 1.3 Vpp differential, 14-bit ADC, up to 1 MSps sampling rate.
- RS-422 In/Out, 2 + 2 channels, up to 50 Mbps.
- Wireless Transceiver with EFR32 2.4 GHz chipsets, support of various standard or proprietary protocols.

B. FPGA Subsystem

For many functions with required level of performance and flexibility, implementation on FPGA is the only choice:

- Processing signals from/to I/O Cards, create process-image data abstraction over them. From simple digital inputs to control of AD / DA converters and digital filters.
- Connection to upper layer by EtherCAT communication protocol. EtherCAT Slave Controller is implemented in the FPGA for the best possible performance.
- Control of data exchange between process-images of individual I/O Cards, process-images of EtherCAT, and optionally process-images of real-time application running on ARM CPU cores.

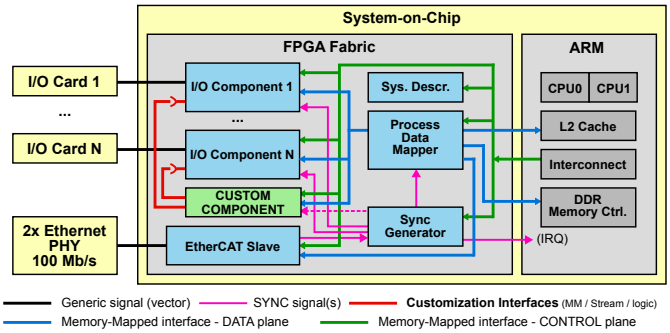

Fig. 3. FPGA Subsystem top level structure.

- Control of system sample timing. In high-performance feedback control applications, everything from input sampling over filters, communication, control algorithms and output latching should be executed in periodic cycle synchronized across the whole plant.
- Optionally, application-specific components can be integrated into the FPGA Subsystem which should provide suitable tooling for such task. Typical example is data fusion from multiple sensors and nerly-zero latency output of the computed value as a "virtual sensor".

FPGA System is modelled as a hierarchical set of Components (IP Cores in FPGA terminology). Top level of hierarchy is outlined in the figure 3. Several types of interconnect between top level IP Cores are defined:

- *CONTROL plane* – Dedicated for non real-time service communication (initialization, parametrization, diagnostics), each component has one MM-Slave port, everything

is connected to one MM-Master port on ARM CPU.

- *DATA plane* – Dedicated purely for real-time critical transport of process data, with no other interference. Each component has typically one (can be more) MM-Slave port. Central component of DATA plane is the *Process Data Mapper* IP Core (or even multiple instances of it to increase throughput) with multiple MM-Master ports connected to: *I/O Component* IP Core instances, EtherCAT Slave Controller, ARM CPU L2 Cache port. Each I/O component provides set of input and/or output registers representing outside physical signals values – process data
- *SYNC plane* – As desribed above, special care needs to be given to time synchronization of all data-related components. In our system design, all cyclic timing is coordinated by the *Sync Generator* IP Core, which provides configurable timing signals to all subordinate components. External signal, typically from EtherCAT Distributed Clock, can be used as a master clock for the Sync Generator.

I/O Component IP Cores:

These IPs implement interface to outside real-world signals through I/O Cards. Each I/O Card type needs its own implementation. Standard internal interfaces of I/O Component are: set of discovery and configuration registers on a CONTROL plane; set of input and output process data registers on a DATA plane; one or more SYNC inputs, used for example to trigger sampling, or even to clock external devices like ADCs synchronously with master clock.

I/O Component can also provide Customization Interfaces which can be used to connect Custom Components with application-specific logic. This is the way how Custom Components can interact with external signals. Depending on use-case, this interface can be in form of plain logic (vector) signals, streaming or memory-mapped.

Sync Generator IP Core:

Sync generator provides vector or widely configurable Trigger signals to all other components which can be used to trigger sampling, data transfer, interrupt in ARM CPU, or even clock for example external ADC. Master clock from EtherCAT or other sources can be used to synchronize all generated signals.

The core of the Sync Generator is timestamp counter. It counts with every FPGA base clock period which is 10 ns:

- normally, increment timestamp by 10,
- if master clock is fast, incmement timestamp by 11 every n-th period,
- if master clock is slow, incmement timestamp by 9 every n-th period.

Thus timestamp provides absolute time value with 1 ns resolution. Correction-th period is generated by a control loop with master clock reference.

Output of Sync generator is vector of Triggers based on timestamp. Start time and period of each Trigger can be individually configured over CONTROL plane registers.

Process Data Mapper IP Core:

Process Data Mapper is controlled by Trigger(s) from Sync Generator. It handles bidirectional data copying between I/O Components and all other targets – EtherCAT Slave Controller memory, ARM L2 Cache and Custom Components. Which data are copied at which Trigger can be configured during runtime by descriptor table available at CONTROL plane registers

Implementation is based on a Scatter-Gather DMA. DMA channel is used to copy actual data, DMA prefetcher commands DMA channel based on Triggers, and Descriptor memory contains configuration and state: Trigger mask, Source address, Destination Address, Transfer length, Control word.

When any Trigger input is activated, DMA prefetcher scan through descriptor table. If the Trigger inputs match the descriptor mask, the descriptor is executed in the DMA channel and data are copied from source address to destination address. Depending on control word, source and destination addresses can be incremented until Mapper restart. This architecture is very flexible and allows many patterns like oversampling, when given input channel is sampled n-times faster than EtherCAT communication cycle and all individual sub-samples data are then transferred in one communication cycle.

C. Management Subsystem

In scope of proposed device, important set of functional requirements exists outside FPGA and real-time domain, mostly to cover system lifecycle management and servicing in industrial environment, notably:

- System bring-up, initialization and configuration of FPGA modules according to configuration data.
- Firmware updates (FPGA, real-time application, system configuration data) over several protocols (FoE - File-over-EtherCAT, HTTP, eventually others).
- Asynchronous (non real-time, non cyclic) communication of diagnostics and auxiliary data, protocols like OPC UA or HTTP REST.
- Optional extensions like basic commissioning web-interface for quick diagnostics without (or not yet configured) upper layer system.

These functions are more or less coupled with real-time domain, but most of them requires, or at least strongly benefits from, services provided by operating system like filesystem, IP network stack or multiple task scheduling.

Linux operating system with PREEMPT-RT real-time extension was chosen as a base for Management Subsystem. It is well supported on all of our SoC platforms and provides access to wide ecosystem of related open source projects.

Thanks to RT extension, also less demanding real-time control tasks, with about 1 ms cycle time, can be reliably executed as a part of Management Subsystem. Several measures can be done to improve real-time characteristics of critical task execution under RT Linux: CPU core isolation only for that task, interrupt threads priority and CPU affinity and L2 cache allocation policy. Even with all these tuning, scheduling jitter of real-time task introduced by adjacent non-real-time load and

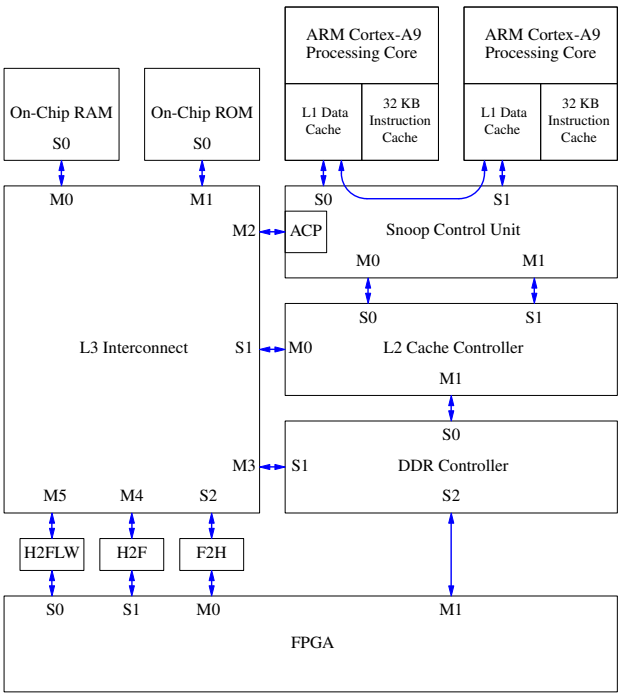

Fig. 4. High-Level Structure of Intel Cyclone V SoC.

in-kernel operations is about 100 µs worst-case for bigger than trivial task according to our tests. Practically, this do not allow cycle time shorter than about 200 µs.

D. Real-time Application Subsystem

Performance limits of Linux task scheduling described in previous section directed us to an another solution for running real-time applications on the ARM CPU next to Linux OS. Higher level of isolation can be reached by dedicating 2nd CPU (CPU1) fully to the real-time application running as a "bare-metal", i.e. out of Linux control without any operating system, next to standard Linux on CPU0. This approach is called Asymmetric Multiprocessing (AMP) and it can provide required performance level on our SoC platform [9].

Our solution is based on Linux kernel driver (module) to enable on-the-fly configuration of the 2nd CPU core. Linux driver provides file-based interface to configure and communicate with AMP core. It is responsible for managing core's state, constructing virtual memory structures, communicating with AMP core and parsing RT-application's executable file (if ELF file is used). This part is really low-level and differs between SoC variants. We have started with Intel Cyclone V with dual-core Cortex A9 (ARMv7). Other selected variants are very similar, with except to Xilinx UltraScale+ based on newer ARMv8 architecture and has completely different structure and cache coherency mechanisms.

Interaction between CPU0 (Management Subsystem), CPU1 (Real-time Application Subsystem) and FPGA Subsystem is outlined in the figure 5 and can be described as follows:

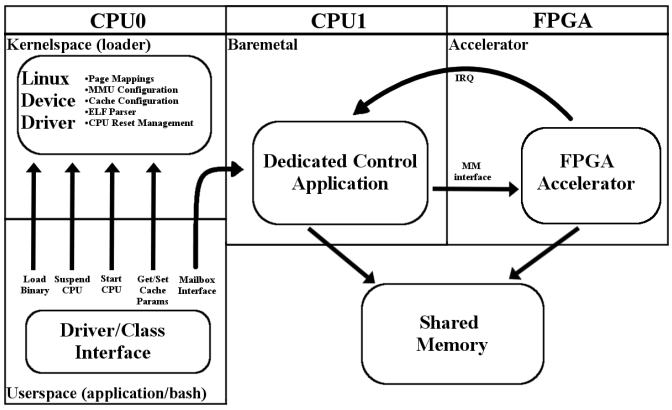

Fig. 5. Structure and interaction of proposed AMP setup.

- CPU0 is responsible for SoC boot, using U-Boot bootloader, downloading configuration into FPGA and initialization of FPGA components according to application configuration and parameters received from fieldbus. Linux is configured to keep CPU1 on reset during boot.
- CPU1 can be initialized by loader running on CPU0. Loader configures virtual memory (different size pages and sections are utilized to eliminate TLB misses), downloads configured executable file to it and starts CPU1.
- Code on CPU1 is responsible for core initialization (caches, FPU, interrupts) and application execution. Typically one cyclic task on CPU1 is triggered by interrupt from FPGA or by ARM timer. Process data communication with outside world is managed by Process Data Mapper in the FPGA, which has direct coherent access to the ARM L2 cache by means of Acceleration Coherency Port (ACP, see figure 4).
  Data access consistency is naturally secured by cyclic operation: input sampling – copy from FPGA I/O to L2 cache – interrupt from FPGA – feedback loop calculation and writing results to L2 cache – cycle time check and overrun detection from ARM side – copy from L2 cache to FPGA I/O. Alternatively some kind of triple-buffer can be implemented in the FPGA for data consistency.
- For asynchronous control and diagnostics of real-time application, bidirectional shared memory mailbox interface is implemented between CPU0 and CPU1. This is implemented as multiple unidirectional FIFO (first-in-first-out) queues which takes advantage of asynchronous inter-processor-interrupts. FIFOs can be used to transport stdin / stdout / stderr streams from CPU1 application to CPU0 and for non-real-time diagnostics like online parameters tuning or data tracing.

CPU core can configure only its own interrupt interface, therefore every AMP core is responsible for configuring its interrupts. Bringing up caches and Memory Management Unit (MMU) can be done only by the core itself via coprocessor configuration interface. To bring up AMP core it is necessary to have direct access to the address on 0x00000000,

this memory region is reserved during bootup and special "trampoline" code is copied to this region. This code directs AMP core to the RT application code. When reset signal is removed the AMP core is not configured to use caching mechanisms, therefore it is necessary to flush L1 and L2 caches after deploying core's target code. Only the processing core itself can fully configure itself, therefore RT application is responsible for invalidating and enabling cache, configuring interrupts and enabling MMU (if it is necessary).

So far, we have working prototype which proved feasibility of this approach and first performance tests indicates no problems with expected performance level for control tasks at 50 µs cycle period next to the Linux on CPU0.

V. Conclusion and future work

This paper presented a new modular hardware platform designed for high-performance I/O level signal processing mostly in motion control applications. Thanks to a modern System-on-Chip semiconductor device which integrates FPGA programmeble logic and multiple ARM CPU cores, our solution is very simple on hardware side and thus cost effective for embedding into machines produces in medium series. By integration of all data procesing including EtherCAT Slave Controller into a single chip, data latency is reduced to the minimum.

Architecture of FPGA part allows full time and frequency synchronization of all data-related parts with master clock from EtherCAT fieldbus or other source. Process Data Mapper component allows flexible, runtime configurable, composition of process data image between I/O nterfaces, EtherCAT and local control application running on dedicated ARM CPU core.

Linux operating system with RT extension is used on one of the ARM CPU cores for management purposes. This brings us features like filesystem, TCP/IP stack and ecosystem of open-source tools at no additional costs. Advanced diagnostics and data interfaces like web server or OPC UA can be built easily on top of Linux.

To overcome fundamental performance limits of real-time tasks scheduled by Linux, Asymmetric Multiprocessing approach, which dedicates one CPU fully to real-time computing task out of Linux sheduling, was explored and partilly verified.

Further effort should be made on performance verification and optimization of FPGA components and EtherCAT fieldbus. Integration workflow for custom FPGA components needs to be specified with respect to high-level synthesis and simulation tools used by control system engineers. Performnace of Asymmetric Multiprocessing technology should be verified with much more details. Some API for process data and parameters access and integration workflow needs to be defined for real-time application code.

Acknowledgment

This work was supported by the H2020 ECSEL JU grant agreement No. 737453 I-MECH project "Intelligent Motion Control Platform for Smart Mechatronic Systems" and the project LO1506 of the Czech Ministry of Education, Youth and Sports under the program NPU I. The support is gratefully acknowledged.

References

[1] M. Cech, A-J. Beltman, K. Ozols. "I-MECH – Smart System Integration for Mechatronic Applications". *Proceedings of IEEE ETFA 2019, Zaragoza, Spain*, Zaragoza, 2019. Paper In Press.

[2] C. Economakos, G. Kiokes and G. Economakos. "Using Advanced FPGA SoC Technologies for the Design of Industrial Control Applications". *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, Corfu, 2015, pp. 1-6. doi: 10.1109/IISA.2015.7388129

[3] L. Gomes, E. Monmasson, M. Cirstea and J. J. Rodriguez-Andina. "Industrial electronic control: FPGAs and embedded systems solutions". *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, Vienna, 2013, pp. 60-65. doi: 10.1109/IECON.2013.6699112

[4] E. Monmasson, L. Idkhajine, M. N. Cirstea, I. Bahri, A. Tisan and M. W. Naouar. "FPGAs in Industrial Control Applications". *In IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 224-243, May 2011. doi: 10.1109/TII.2011.2123908

[5] D. Orfanus, R. Indergaard, G. Prytz and T. Wien. "EtherCAT-based platform for distributed control in high-performance industrial applications". *2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, Cagliari, 2013, pp. 1-8. doi: 10.1109/ETFA.2013.6647972

[6] J. Jasperneite, M. Schumacher and K. Weber. "Limits of increasing the performance of Industrial Ethernet protocols". *2007 IEEE Conference on Emerging Technologies and Factory Automation (EFTA 2007)*, Patras, 2007, pp. 17-24. doi: 10.1109/EFTA.2007.4416748

[7] G. Prytz. "A performance analysis of EtherCAT and PROFINET IRT". *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, Hamburg, 2008, pp. 408-415. doi: 10.1109/ETFA.2008.4638425

[8] X. Wu and L. Xie. "End-to-End Delay Evaluation of Industrial Automation Systems Based on EtherCAT". *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*, Singapore, 2017, pp. 70-77. doi: 10.1109/LCN.2017.14

[9] X. Chen, Y. Gu, C. Wang and X. Guan. "Asymmetric multiprocessing for motion control based on Zynq SoC". *2016 International Conference on Field-Programmable Technology (FPT)*, Xi'an, 2016, pp. 315-318. doi: 10.1109/FPT.2016.7929570

[10] I-MECH Consortium. "I-MECH – Intelligent Motion Control Platform for Smart Mechatronic Systems". [Online]. https://www.i-mech.eu/

[11] EtherCAT Technology Group. "EtherCAT – the Ethernet Fieldbus". [Online]. https://www.ethercat.org/en/technology.html

[12] PROFIBUS & PROFINET International. "PROFINET – the leading Industrial Ethernet Standard". [Online]. https://www.profibus.com/technology/profinet/

[13] EPSG. "ETHERNET POWERLINK". [Online]. https://www.ethernet-powerlink.org/

[14] National Instruments. "CompactRIO". [Online]. http://www.ni.com/compactrio/

[15] Speedgoat GmbH. "Speedgoat - real-time simulation and testing". [Online]. https://www.speedgoat.com/

[16] BiSS Association e.V. i.G. "About BiSS" [Online]. http://biss-interface.com/about-biss/

[17] Intel Corporation. "INTEL SOC FPGAs". [Online]. https://www.intel.com/content/www/us/en/products/programmable/soc.html

[18] Xilinx Inc. "SoC Portfolio". [Online]. https://www.xilinx.com/products/silicon-devices/soc.html

[19] Enclustra FPGA Solutions. "System-on-Chip Modules". [Online]. https://www.enclustra.com/en/products/system-on-chip-modules/