Bulletin of Electrical Engineering and Informatics
ISSN: 2302-9285
Vol. 6, No. 4, December 2017, pp. 358~363, DOI: 10.11591/eei.v6i4.867

System on Chip Based RTC in Power Electronics

R. Dorothy*, Sasilatha. T
Department of Electrical and Electronics Engineering (Marine), AMET University, Chennai, India
*Corresponding author, e-mail: eei@journal.uad.ac.id

Abstract
Current control systems and emulation systems (Hardware-in-the-Loop, HIL or Processor-in-the-Loop, PIL) for high-end power-electronic applications often consist of numerous components and interlinking busses: a micro controller for communication and high level control, a DSP for real-time control, an FPGA section for fast parallel actions and data acquisition, multiport RAM structures or bus systems as interconnecting structure. System-on-Chip (SoC) combines many of these functions on a single die. This gives the advantage of space reduction combined with cost reduction and very fast internal communication. Such systems become very relevant for research and also for industrial applications. The SoC used here as an example combines a Dual-Core ARM 9 hard processor system (HPS) and an FPGA, including fast interlinks between these components. SoC systems require careful software and firmware concepts to provide real-time control and emulation capability. This paper demonstrates an optimal way to use the resources of the SoC and discusses challenges caused by the internal structure of SoC. The key idea is to use asymmetric multiprocessing: One core uses a bare-metal operating system for hard real time. The other core runs a "real-time" Linux for service functions and communication. The FPGA is used for flexible process-oriented interfaces (A/D, D/A, switching signals), quasi-hard-wired protection and the precise timing of the real-time control cycle. This way of implementation is generally known and sometimes even suggested–but to the knowledge of the author's seldomly implemented and documented in the context of demanding real-time control or emulation. The paper details the way of implementation, including process interfaces, and discusses the advantages and disadvantages of the chosen concept. Measurement results demonstrate the properties of the solution.

Keywords: *SoC, control, cache interference, multiprocessing*

1. Introduction

A system on a chip includes a real time clock (RTC) module, a crystal oscillation circuit and a voltage supply circuit. The RTC module is coupled to provide timing functions and the crystal oscillation circuit is coupled to produce an oscillation. The voltage supply circuit is coupled to produce a supply voltage for at least a portion of the RTC module and the crystal oscillation circuit.The use of multicore architectures for real-time control applications is a challenging field and forces the developers to take special care of the decomposition to leverage the availability of parallel processing [1-2]. For rapid prototyping systems and/or low effort development projects these kinds of optimizations are up to now seldomly reported. The coupled use of two kernels in a multiprocessing environment puts a too high demand on the implementation of the control code and the code for communication.

The interaction of tasks and the advantages gained are hard to predict. Interrupt service routines may cause interesting side effects. This paper presents a general breakup of the service tasks and the control tasks, leading to a strict function-based parallelization and nearly complete decoupling of the real-time control cycle from ancillary services. For strict timing the real-time core directly interacts with the FPGA–a proven solution [3-4]. In the first part the requirements for real- time control systems are evaluated. Afterwards a structure fulfilling these requirements is developed. Interfaces for coupling to processes, e.g. a test-bench setup, are presented. Finally the functionality of the platform and its peripherals is demonstrated by benchmark tests and measuring results taken at a laboratory test set-up. A similar approach–not requiring real- time functionality–is detailed in [5], also providing nearly 70 further citations. This paper consequently concentrates on few selected citations [6].

The demand for rapid prototyping and small series control system has special requirements to the flexibility of the hard-and software of the control system. During a

Received August 30, 2017; Revised October 31, 2017; Accepted November 14, 2017

development process the computational performance has to be higher–the need to optimize for performance issues should be avoided, a user friendly coding should be provided and additional resources for debugging tasks may be required. Furthermore it is necessary to implement standard functions like measurement, pulse-pattern generation (e.g. PWM), trace functionality and the user-interface out of the box–without the need of an individual implementation [7].

On the process interface side the demands could change quickly–adding additional measurements or other types of physical measuring interfaces have to be possible at any time. However, the system and its costs should be as lightweight as possible. A modular adaptable concept with a selection of standard modules might be the best choice.

Testing new control algorithms introduces risks for the power electronic components. Therefore a fast and flexible HPS-independent safety layer is reasonable. An FPGA-based limit check allows reactions in a few microseconds–fast enough to protect sensible semiconductors–and can be programmed independently from the control code. All together these requirements can be summarized as following and will be met by the introduced system:

Currently in most of the SoC design data layout is also performed manually and it has two major problems:(1) the development time is significant–not acceptable for current-day time to market requirements, (2) quality of solution varies based on the expertise.

2. The Proposed Solution

SoC's are designed to boot from a single source. This significantly simplifies the boot process and firmware updates compared to current heterogeneous multi-chip control systems [8]. User-friendly and simple integration of tasks is best achieved by using a "high-level" operating system like Linux. For added speed and more predictable reaction time a "real-time" variant of Linux is sensible. In this context the meaning of "real-time" has to be discussed in more detail:

- Power-electronic systems often require "real-time" cycles of 100 μs and below, the bare-metal system used here and presented later on in this paper easily reaches 20 μs (if the control code is compact enough)
- "real-time" Linux avoids long blocking of interrupts by service tasks (e.g. network traffic), but it does not allow for full priorization of certain tasks as would be required for short cycle time. From the point of view of power electronics it is not "real-time"
- Commercial real-time operating systems provide a similar kind of real-time functionality as presented here–but are normally not able to avoid all interrupts and fully concentrate on the control code.

It is evident that from the perspective of usability a "high- level" operating system like Linux with many pre-fabricated services and functions is preferable. The interaction of these service tasks with the most important real-time control process is, however, severe. An operating system giving the control over all interrupt services to the control process would be optimal–but this would not allow for easy integration of service functions. Also, re-programming of all components (firmware, software, FPGA structure) would be hard to realize and cause high effort [9]. The concept used here combines the advantages of both systems, Figure 1. One of the cores is running "real-time" Linux and provides service functions. These include software and firmware updates via TCP/IP, process control like set-point value transmission and logging functions (e.g. streaming measured data to network-attached storage, NAS). The second core runs a bare-metal application containing the control tasks [10]. It has no direct link to external devices (with the exception of a very fast FPGA link to transmit data in both directions for process interaction). In consequence the control latency is nearly independent from the service tasks being executed. However, as will be seen, some unavoidable interference induced by the shared access to the Level 2 cache and the on-board SD RAM influences the control execution, Figure 2. The amount of interference, and possible reasons and countermeasures, is analyzed later on in this paper.

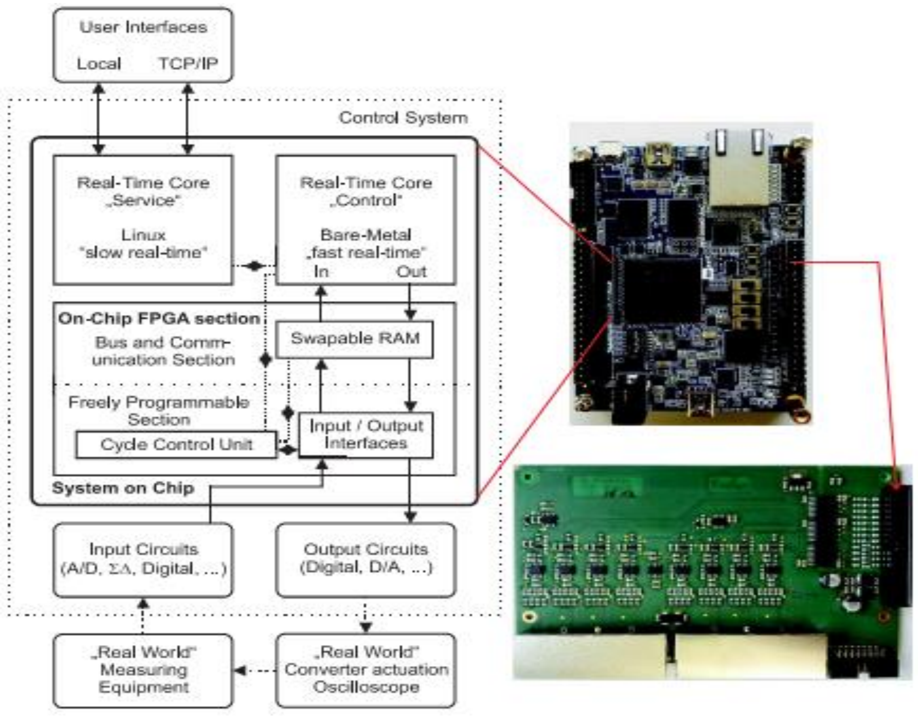
User Interfaces
Local
TCP/IP
Control System
Real-Time Core „Service"
Linux "slow real-time"
Real-Time Core „Control"
Bare-Metal „fast real-time"
In
Out
On-Chip FPGA section
Bus and Communication Section
Swapable RAM
Freely Programmable Section
Cycle Control Unit
Input / Output Interfaces
System on Chip
Input Circuits (A/D, ΣΔ, Digital, ...)
Output Circuits (Digital, D/A, ...)
„Real World" Measuring Equipment
„Real World" Converter actuation Oscilloscope

Figure 1. left: Basic structure of the used SoC platform, right: photo of DE0-Nano-SoC and Peripheral (8 Channel ADC-board)

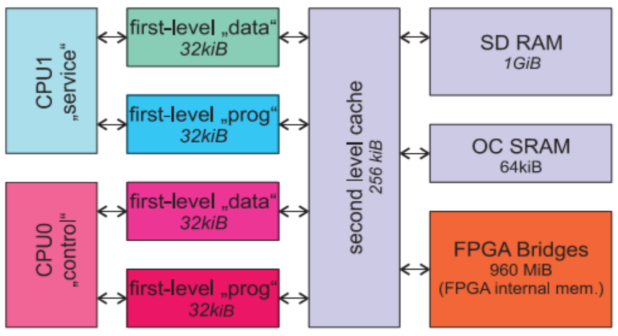

Figure 2: Cache and interface structure of the SoC

3. Memory Interfaces and Interferences

The ARM9 utilizes a common second level cache connecting both CPU's with the memories (cp. Figure 2). The on chip RAM (OC RAM) is faster than the external SD RAM and is selected for the data exchange between both CPU. The SD RAM provides one GiB of Ram. 256MiB are reserved for the Linux OS, 64MiB for the bare-metal OS, the remaining 704 MiB are reserved as shared memory (under direct control of the bare-metal OS), mostly for trace data. All operations relating to program and data contained in the first-level-cache associated to a core is fully independent of the activity of the other core. Unfortunately it is essential for the control to access the FPGA-Bridge for control purposes and the SD RAM for trace functionality in every control cycle (typical from 20 µs to 500µs–depending on the type of control). This requires interaction with the second level cache which introduces interference with the service core. In our experience the CPU load of the service core remains below 1% and causes negligible access to the second level cache.

However, executing code requiring much access to the SD RAM may cause relevant interference between the control and the service tasks. Figure 3 gives an example for such interference. Optimization of SRAM array structure for energy efficiency improvement in advanced CMOS technology [6].

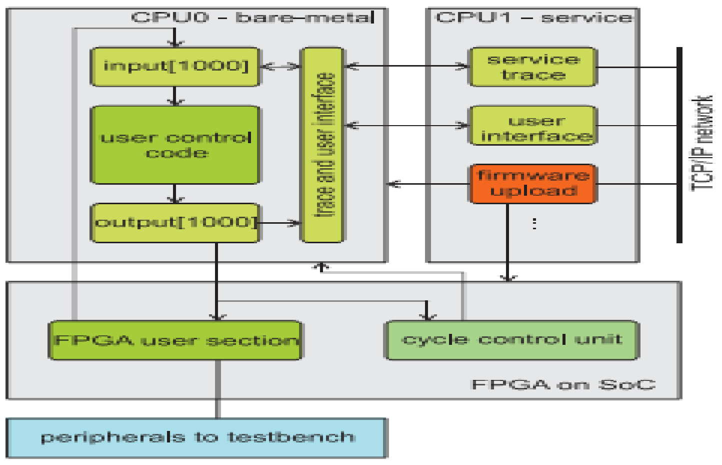

Figure 3. Example for such interference

The execution time of the selected control code, including acquisition of measured data and trace functionality, takes between 28µs and 34µs in normal operation (cp, t < 0 s). In this state the control code deposits measured data in a circular buffer for later storage in a data file (which is then provided by the service core). It is important to notice that the system layout allows for flexible cycle time (e.g. for random PWM or PLL-controlled cycle-time adjustment in normal PWM mode). Consequently, each stored measurement contains its own (and possibly unique) cycle time. Upon a trigger event a pre-defined pre-trigger sequence and a post-trigger sequence are to be stored in the data file.

With the trigger event at t=0 s a service function on Linux prepares the data for storing in a data file. For this it has to find the starting point of the pre-trigger sequence. This cannot be derived from the cycle time, because the cycle time may vary. The service function has to sum up all cycles backwards into the past until it reaches the desired pre-trigger duration. A large amount of data is read from the SD RAM at very high speed (the service core has nothing else to do and provides high calculation power). A lot of cache misses result, loading the second-level cache from the side of the service core. This increases the time which the bare-metal core needs for memory access-the execution time of the control code reaches a short peak of up to 68 µs. This is more than twice the mean value–and could very well violate the allowed control cycle duration, causing the system to trip. After this initialization of the trace service the data access rate becomes moderate. Here the write access to slower storage devices or network interfaces helps. The cache interference remains, but is much more tolerable. In consequence service tasks have to be slowed down deliberately in processes with high data access. This is no significant drawback in performance: The service core has no strict real time requirements or priorities. Furthermore, adding pauses to self-made service routines is simple. However, it is clear that software on the Linux core can interfere with the real-time control–software components should be tested carefully before being used: Even a short load peak of few µs could violate the real-time requirements of the real-time control code.

4. The Peripheral Concept

In context of power electronics in regions of several kilowatts two signal standards are common. Commonly used standards for analogue quantities are current signals in regions ±20mA to ±200mA. Current-based signals offer high noise immunity, consequently allowing for long-distance connections in EMI challenging environments. Many types of sensors from isolating current and voltage probes to temperature sensors are available. For digital signals like communication, digital signal transmission (e.g. sigma-delta-modulated signals) and PWM signals fibre optics present a fast and simple option. They are nearly unaffected by EMI and their transmission delay is practically independent from the cable length allowing for precise timing. Also, they provide nearly perfect isolation making additional galvanic isolations needless.

To handle multiple tasks a modular platform is developed. Each of the two GPIO slots can be equipped with an adapter- board for four of the following peripheral modules:

- 8 channel current input analogue to digital converter
  - Several signal standards up to ± 200 mA (directly compatible with common current sensors)
  - Up to 1 MSPS-1µs sampling time
  - 12 bit resolution
  - Real parallel sampling of all 8 values
- 8 channel current output digital to analogue converter
  - Several signal standards up to ± 100 mA
  - Up to 1 MSPS-1µs sampling time
  - 12 bit resolution
  - Real parallel output of all 8 values
- 8 channel versatile link fibre optic
  - Input/output direction arbitrary during assembly
  - 50 MSPS
  - Highly isolated and robust converter coupling.

  The adapter board itself provides the following features:
- Fast FPGA-Watchdog functionality providing save switching signals during boot and programming phase of the FPGA
- Sufficient ground connection for all connections
- Supply of 3.3 V, 5.0 V and ±15 V
- Four Module Slots
- TWI-interface option for timing uncritical port expansions (e.g. contactor, temperatures, hardware-user-interface: switches, LED’s, As all these components connect to the FPGA, flexible adaption by suitable PCB and adapted VHDL code is straightforward.

5. User Code Integration and Base Functions

For an efficient development chain a structure with appropriate interfaces is implemented, Figure 3. All standard tasks for cycle based control are encapsulated in a precompiled bare- metal framework which can be linked with a pre-compiled user code using a standard gcc compiler. The user code is executed iteratively–the timing can be set by the user code and will be realized precisely by the FPGA. The FPGA samples measurements synchronized to the control cycle–it can provide up to 1000 float values to the user code per control cycle. These are collected in an input array residing in memory. The direct integration of HPS and FPGA allows a memory swap between FPGA and HPS within one FPGA cycle (10 ns)–the HPS can then access the newly available values directly, limited only by the speed of memory access (see above). During the control cycle output values to the FPGA (and the peripherals attached to the FPGA) are collected in an output array accommodating space for 1000 float values. At the end of the control cycle the memory section containing the output values is also swapped and in this way made available to the FPGA. The FPGA user section may contain VHDL code providing measurements, pre- calculations (like mean values), margin checks and pulse pattern generation suitable for the control of the test bench. Individual features can be integrated easily. The trace functionality can be configured by the service core during run-time. It is possible to trace a selection of input and output values . Furthermore the service core can upload new firmware to the FPGA and bare-metal. The whole system is fully programmable and controllable over a TCP/IP network.

6. Results and Discussion

The control platform is driving a 200kW 3-phase-two-level- converter connected to an induction machine. To make misbehaviour obvious an open loop control is implemented in the user control code section. A closed-loop control algorithm would suppress most abnormalities. Cycle time is set to 200µs–matching the maximum valve switching frequency of the converter.

The average calculation time including measurement acquisition and transfer of commands to the FPGA is about 13µs for this application. A PWM implemented in the FPGA user section generates the valve control signals which are transmitted via fiber optics to the converter. The converter currents are transformed by current transducers provide +/-200 mA for +/- nominal current, they directly connect to the ADC-module of the control system. The ADC-Module samples the current synchronized to the PWM reference–in consequence the measuring result show only the fundamental frequency. The results show the expected behavior, additional scope measurements prove the correct functionality of the trace.

7. Conclusion

The findings presented in this paper characterize the chosen concept and the selected SoC device as a cost-effective, efficient and highly performant solution for the control and emulation of power electronic systems. The combination of two ARM 9 cores and an FPGA section gives flexibility and adaptability. The distribution of tasks to the two cores, using asymmetrical multiprocessing, provides a reliable basis for real- time systems. Service core (Linux OS, for communication and ancillary services) and control core (bare-metal OS, for the real- time control code, down to 20 µs cycle time) run independently, each is optimized for the tasks allotted. However, care has to be taken on the side of the Linux part of the system: The inherent and unavoidable interference caused by a second level cache and SD RAM supporting both cores is not negligible. Fast and frequent SD RAM access initiated by the Linux core may severely increase the calculation time needed by the real-time core. The suggested countermeasure is to force mandatory pauses (with no or low SD RAM access) in the software running on the Linux system.While this is simple for self-made software components care has to be taken when integrating third-party functions into the Linux system.

References

[1] Dong C, Zeng H. *Minimizing Stack Memory for Hard Real-time Applications on Multicore Platforms.* Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden. 2014; 1-6.

[2] Paun V, Monsuez B, Baufreton P. *On the Determinism of Multi- core Processors.* French Singaporean Workshop on Formal Methods and Applications (FSFMA), Singapore. 2013; 32-46.

[3] Abourida S, Belanger J, Dufour C. *Real-time HIL Simulation of a Complete PMSM Drive at 10 µs Time Step.* European Conference on Power Electronics and Applications, Dresden. 2005; 9.

[4] Vincke R, Messiaen A, Boydens J. Hybrid FPGA/Multi-core CPUs for Industrial Applications. *Annual Journal of Electronics*. 2013. ISSN 1314-0078.

[5] Vardhan H, Akin B, Jin H. A Low-Cost, High-Fidelity Processor-in-the-Loop Platform. *IEEE Power Electronics Magazine,* 2016; 18- 41.

[6] Ashwin JS, Praveen JS, Manoharan N. Optimization of SRAM Array Structure for Energy Efficiency Improvement in Advanced CMOS Technology. *Indian Journal of Science and Technology,* 7(S6): 35-39.

[7] Jyothi B, M.Venugopala Rao. Carrier-Based PWM Technique for Inverter-Fed Multiphase Induction Motor. *International Journal of Electrical and Computer Engineering (IJECE)*. October 2016; 6(5): 1967-1984.

[8] Trio Adiono, Aditya F. Ardyanto, Nur Ahmadi, Idham Hafizh, Septian G. P. Putra. An SoC Architecture for Real-Time Noise Cancellation System Using Variable Speech PDF Method. *International Journal of Electrical and Computer Engineering (IJECE)*. December 2015; 5(6): 1336-1346.

[9] Suman S, Sharma KG, Ghosh PK. 250 MHz Multiphase Delay Locked Loop for Low Power Applications. *International Journal of Electrical and Computer Engineering (IJECE)*. 2017; 7(6): 3323-3331.

[10] Rajalakshmi C, ArGunasekaran IK. Low-power High-speed Amplification Using Filterbank for Digital Hearing Aids. *International Journal of MC square Scientific Research (IJMSR)*. 2016; 8(1):60-73.