Modeling Multicore Programmable Logic Controllers in Networked Automation Systems

M. Hashemi Farzaneh, S. Feldmann, *Student Member IEEE*,
C. Legat, J. Folmer, B. Vogel-Heuser, *Senior Member IEEE*
Institute of Automation and Information Systems (AIS), Technische Universität München (TUM)
85748 Garching near Munich, Germany
Email: {hashemi, feldmann, legat, folmer, vogel-heuser}@ais.mw.tum.de

***Abstract*—Integrating Multicore Programmable Logic Controllers (PLC) in Networked Automation Systems (NAS) promises a higher computing performance and PLC manufactures are launching their first Multicore PLCs (MPLC). However, analyzing the MPLCs time behavior considering distributing automation tasks on different cores is more complex than analyzing the time behavior of single-core PLCs. For increasing analyzability of NAS a modeling notation is needed. However, MPLCs are not considered sufficiently in modeling NAS nowadays. In this paper, properties and requirements of MPLCs regarding the distribution of automation tasks on multiple cores are explored. These properties and requirements affect real-time constraints of automation tasks of a Multicore-Capable NAS (MCNAS). Based on the explored properties and requirements a modeling notation for modeling MPLCs is proposed. This modeling notation offers the automation engineer higher analyzability of the time behavior of MCNAS.**

***Keywords*—automation, networked automation systems, modeling, multicore, plc.**

I. Introduction

Multicore technology is developing rapidly and multicore processors offer a higher computing performance in comparison to single-core processors because of the ability of parallel code execution on different cores. This trend increases for Networked Automation Systems (NAS) [1]. NAS in industrial automation describes an automation system consisting of sensors and actuators as well as interconnected controllers (e.g. Programmable Logic Controllers (PLCs)) [2]. Automation tasks that have to satisfy real-time requirements need more computing capabilities. The modest price of multicore CPUs and significant increasing of computing performance motivate NAS operators to integrate multicore processors in PLCs [9]. Moreover, integrating Multicore Programmable Logic Controllers (MPLC) in NAS offers the possibility to increase system reliability, e.g. to compensate failures of hardware components (MPLC or cores of MPLC). In case of a MPLC or core failure, software components can be reassigned to other hardware with sufficient remaining computing capacity. Because of distributing automation tasks on different cores, analyzing the time behavior of MPLC is more complex than analyzing the time behavior of single-core PLCs. Due to increasing complexity of the NAS, a modeling notation for modeling NAS architecture and time behavior is needed [3]. The objective of such a modeling notation is increasing analyzability of the NAS architectures on the one hand and offering a basis for verification of the time behavior of a modeled NAS [2] on the other hand.

However, modeling MPLCs has not yet been sufficiently integrated into modeling NAS although NAS and MPLCs are increasing the system's complexity drastically. Furthermore, scheduling of MPLCs influences the timing behavior of NAS and vice versa. The effect of these influences on the system's overall timing behavior is too complex to be determined manually and based on expert knowledge solely. This may lead to misconfigured systems being discovered during the plant's commissioning phase and, thus, real-time constraints are not satisfied or the overall system is not working properly making cost-intensive re-engineering necessary. Up to now, there is no holistic modeling approach to model MPLCs within NAS and, hence, no verification process for estimating the timing behavior of NAS is applied to support engineers during the design phase.

In this paper, analyzing and modeling MPLC is focused. Analyzing MPLC, properties and requirements are derived which affect the time behavior of MPLC in a Multicore-Capable Networked Automation System (MCNAS). These properties and requirements are discussed and utilized for developing a modeling notation for MPLCs. To demonstrate the developed modeling notation a real example from industry, a hydraulic press, is utilized and excerpts of its model are presented.

The paper is structured as follows: In the next section, the state of the art related to multicore challenges and modeling automation systems is discussed. In section III, software and hardware properties and requirements for modeling MPLC are classified. A notation for modeling MPLC is presented in section IV. An application example from industry demonstrating the modeling notation for MPLC is introduced in section V. The paper is summarized and an outlook on future work is given in section VI.

II. State of the Art

In the following, the state of the art with focus on modeling notation for automation systems, general challenges of multicore processors and challenges of multicore processors in real-time automation systems is presented. Gomaa [4] describes a method for designing real-time and distributed applications, which integrates object-oriented, and concurrency concepts by using the Unified Modeling Language (UML). Katzke et al. [5] introduce an implementation oriented approach for object oriented software development of heterogeneous distributed systems. In this approach, UML is extended by model elements for representing control code as well as small-scale patterns

978-1-4799-0224-8/13/$31.00 ©2013 IEEE

for plant automation. Huber et al. [6] introduce a graphical notation for characterizing the software architecture and to distribute processes and tasks on processors in the automotive domain. Witsch et al. [7] present a modeling notation for designing communication networks in automation systems. It includes elements of NAS such as sensors, actuators, PLCs, switches, etc. This modeling notation offers automation engineers a design instrument for earlier phases of system design. Vogel-Heuser et al. [3] consider time requirements for modeling NAS and consider time-related information separated into time requirements and properties. Frey et al. [8] introduce a Network-Controller- using Modelica to model distributed automation systems. These models can be applied for analyzing response time in open and closed-loop control systems. None of the previously described approaches consider specific characteristics of MPLC.

Hansen [9] discusses the emerging trend of using MPLCs in industrial informatics and factory automation and addresses the existing trends, the future possibilities and challenges such as accessing shared hardware. Byna et al. [10] address the problem of accessing shared memory in multicore CPUs and introduce simple analytical models for prediction of the occurrence of data access contention and provide a guideline for choosing the optimal number of CPU cores to run an application without causing data access contention. Chen et al. [11] discuss the problem of the high complexity of internal behavior of multiprocessor system-on-chip (MPSoC) and present a general framework for profiling MPSoC embedded systems. They suggest a framework helping designers to identify performance problems and to improve the architecture of embedded systems.

Paolieri et al. [12] claim that current multicore CPUs are less analyzable than single-core CPUs and propose a multicore architecture with shared resources that allows the execution of applications with hard real-time and non-hard real-time constraints at the same time. It provides time analyzability for hard real-time tasks. Rosen et al. [13] suggest that cache misses play a significant role estimating Worst Case Execution Time and analyzes four Bus Scheduling Approaches and optimizes two of them to enhance the predictability of real-time applications on multicore CPUs. Saifullah et al. [14] suggest that multicore processors have potential to enable real-time applications with timing constraints that cannot be fulfilled on a single-core processor. Furthermore, the approach addresses the problem that hardware technology regarding multicore processors is moving at a rapid pace but developing software and programming models is slow. For exploiting multicore resources, a new task decomposition algorithms is presented. This algorithm considers the intra-task parallelism. Oriol et al. [15] present FASA as a scalable component framework for distributed control systems. FASA computes an optimal schedule for automation tasks considering real-time constraints. Unfortunately, none of the related works which deal with challenges of multicore processors considers the modeling of MPLC.

The reviewed literature deals either with modeling notation for NAS without considering issues of multicore processors or address challenges of multicore processors in non-real-time and real-time systems without discussing modeling. Modeling MPLCs in automation systems has not been considered sufficiently. For this reason, modeling notations of automation systems and specific characteristics of MPLCs are brought together in this paper for modeling MPLCs in NAS.

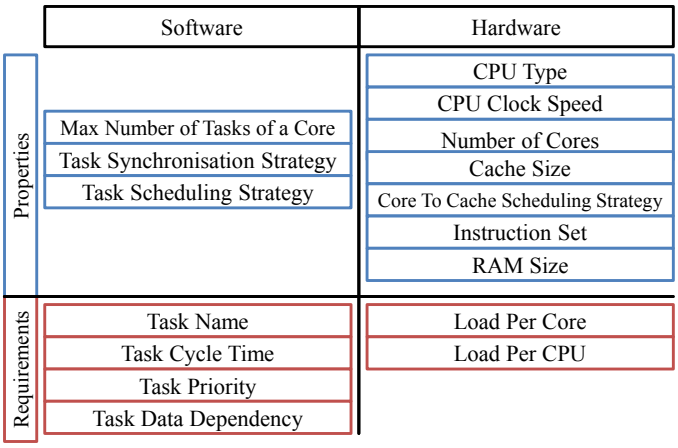


Fig. 1. Explored properties and requirements for modeling MPLC

III. Analyzing Properties and Requirements Concerning MPLCs

In this section, software and hardware properties and requirements are investigated which are related to MPLC. For this purpose information from two sources are collected. The first source is based on data sheets of MPLCs which describe the hardware and software properties of MPLCs (e.g. CPU Type, Task Scheduling Strategy). The second source is based on multicore processor challenges discussed in literature. The challenges of multicore processors are crucial for real-time automation systems for making a statement whether automation tasks deadlines can be met. By not meeting these deadlines in hard real-time systems, catastrophic sequences may occur.

Analyzing MPLCs, two classes of parameters can be determined. The first class deals with hardware-related parameters and the second one deals with software-related parameters. For each class, properties and requirements are defined as sub classes. Properties class includes parameters which come with a MPLC as a product and are constant (e.g. CPU Type, Engineering Environment installed on a MPLC, etc.). Moreover, the class of requirements consists of parameters which are set by designers of a MCNAS and have to be fulfilled in order to have a correct functioning MCNAS (e.g. Load per Core). Fig. 1 depicts an overview of investigated parameters. From the explored hardware and software properties and requirements, we derive MPLC time requirements. These time requirements have to be fulfilled in order to meet the task cycle time can be met. The meeting of the task cycle time depends on meeting these time requirements. In the following subsections the modeling of these properties and requirements are discussed.

A. Hardware properties

CPU Type includes information about CPU producer and CPU model of the multicore CPU in a MPLC (e.g. Intel i7 2715QE). CPU Clock Speed is a significant factor for estimating the performance of a CPU. High Number Of Cores offers

high computing performance regarding parallel computing of automation tasks. Cache is a rapid buffer memory in the CPU that stores process data. Using cache, repeated access on Random Access Memory (RAM) can be avoided. In this way, data access delays are reduced. Cache plays an important role in a multicore CPU [10] and increases the computing performance. Having a big Cache Size, more data can be stored for a rapid access by CPU. The time behavior of a multicore-CPU also depends on the *Core To Cache Scheduling Strategy* for competing cores while accessing the shared cache [16]. *Core To Cache Scheduling Strategy* deals with strategies such as *two-phase locking strategy* that control the concurrent cores, accessing a shared cache. The concurrency control mechanisms lead to data access delays which affect the performance of the MPLC. *Instruction Set* specifies the set of instructions (programming instructions) of a CPU. *Instruction Set* affects the performance of CPU allocating RAM. For instance, a CPU with a 32-bit instruction set can allocate only 4 Gigabyte of the RAM but a CPU with a 64-bit instruction set can allocate around 16 Exabyte $(18.4 * 10^{18} bytes)$ of the RAM. The bigger the RAM Size, the higher the performance of PLC. Having big *RAM Size* more data can be stored at the run-time for a rapid access by CPU.

B. Hardware requirements

Load per Core specifies the load on a core, which will be set by the designer of MCNAS. This parameter must be set if a core must not exceed a specific temperature regarding to security issues. High CPU temperatures can lead to CPU or core failures. Failures in automation systems in critical environments such as nuclear power plants can endanger people in the neighborhood. *Load per CPU* limits the whole load on a multicore CPU. Here is the previously mentioned reason also valid.

C. Software properties

A task in automation systems, describes a program which runs on a PLC in a cyclic manner. *Max Number of Tasks of a Core* limits the number of running tasks on a core. Depending on *Task Synchronization Strategy* different task execution times are expected and fulfilling real-time constraints can be guaranteed [18]. For parallel running tasks a *Task Synchronization Strategy* is needed to avoid that two tasks access a shared data at the same time. Without task synchronization, tasks can calculate with not-updated data. Working with not-updated data, leads to wrong calculations and results. Task Scheduling Strategy is needed when two or more competing tasks run on a CPU. Depending on parameters such as task priority, a running order for tasks is specified. Task Scheduling Strategy plays a significant role estimating execution time of a task [14].

D. Software requirements

Task requirements come from the automation engineers programming MCNAS. Each automation task has a unique *Task Name* and a *Task Cycle Time* (task deadline). *Task Cycle Time* must be met in order to have a correctly running closed-loop automation system. Each task has a *Task Priority* that has to be considered depending on the scheduling strategy which is specified as *Task Scheduling Strategy*. Software in automation systems is modularized and consists of so-called Program Organization Units (POU) [19]. An automation engineer assigns POUs to tasks while designing an automation system. Depending on the complexity of the POUs, the execution time of a task is affected. Two automation tasks T1 and T2 (consisting of POUs) are defined as dependent tasks (*Task Data Dependency*) if an output variable of T1 is input variable of T2 or vice versa. T1 and T2 are also dependent tasks if they share global variables. High data dependencies lead to longer execution times. The reason is the task synchronization overhead which was discussed in subsection C.

TABLE I. DERIVED TIME REQUIREMENTS

Time Requirements	ID	Derived from
Task Synchronization Delay	Task_Sync_Delay	Task Data Dependency, Task Synchronization Strategy Per Core
Task Memory Access Delay = Core To Cache Delay or Core To Cache Delay + Core To RAM Delay	Task_Mem_Delay	Cache Size, Core To Cache Scheduling Strategy, RAM Size
Priority Dependent Task Delay	Task_Prio_Delay	Task Priority
Scheduling Dependent Task Delay	Task_Sched_Delay	Task Scheduling Strategy Per Core
Computing Performance Delay	Com_Delay	CPU Clock Speed, CPU Instruction Set, Load Per Core, Load Per CPU

E. Deriving time requirements

From the previously discussed requirements and properties, time requirements are derived. These time requirements are both hardware-related and software-related. *Task Synchronization Delay* (Task_Sync_Delay) is caused by software requirement Task Data Dependency and property Task Synchronization Strategy. The more shared variables between two parallel tasks, the higher the synchronization delay. *Task Memory Access Delay* (Task_Mem_Delay) deals with the delay accessing a data from memory and is affected by hardware properties Cache Size, Core To Cache Scheduling Strategy, RAM Size. Task_Mem_Delay is equal to *Core To Cache Delay* if data exists already in Cache. If the data does not exist in Cache, it is equal to *Core To Cache Delay + Core To RAM Delay*. *Priority Dependent Task Delay* (Task_Prio_Delay) is affected by software requirement Task Priority: the higher the priority of a task, the shorter the Task_Prio_Delay. *Scheduling Dependent Task Delay* (Task_Sched_Delay) is the delay caused by software property *Task Scheduling Strategy*: the more efficient is the scheduling strategy, the shorter Task_Sched_Delay. An efficient scheduling strategy guarantees that all tasks are done as fast as possible. The *Computing Performance Delay* (Com_Delay) is the delay depending on CPU properties CPU Clock Speed, Instruction Set, Load per Core, and Load per CPU. Table IDerived time requirementstable.1 shows an overview of the derived task time requirements. The sum of the derived time requirements of a task has to be less than its cycle time in order to fulfill real-time constraints of the task.

IV. MODELING NOTATION FOR MCNAS

In this section, the investigated properties and requirements from the previous section are transferred into a modeling

TABLE II. MODELING NOTATION ELEMENTS FROM [3]

Communication-related time behavior	
Application-related time behavior	
Host and network line	Host
PLC with CAN bus adapter and I/O-modules, control task running on PLC	P L C, CAN, S, S, A, A, 8x, 8x, 8x, 8x
Digital/analog sensor and digital/analog actuator	S, S, A, A

notation for MCNAS. Firstly, a modeling notation of NAS regarding NAS architecture and time behavior is introduced. Secondly, new elements for modeling MPLC are proposed. These modeling elements include the investigated properties and requirements of MPLC.

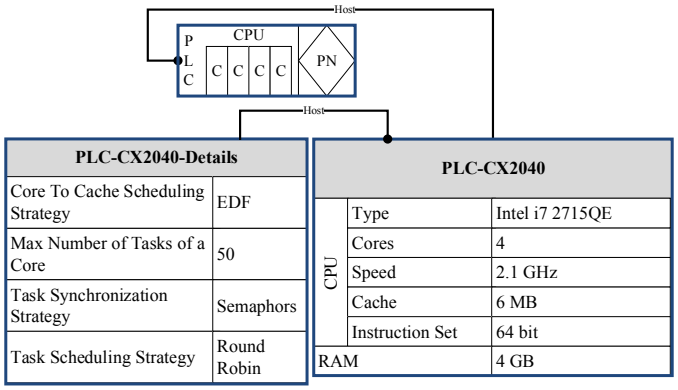
A. Modeling notation for NAS as basis

The modeling notation developed by Vogel-Heuser et al. [3] covers the basic components of a NAS such as sensors and actuators and PLCs. In this notation, time requirements of a NAS are integrated providing the ability to compare NAS architectures and their time behavior. Table IIModeling notation elements from [3]table.2 shows an excerpt of the basic components of the modeling notation and the components of the modeling notation regarding hardware and time requirements and properties. Modeling elements related to the time behavior are divided into communication-related and application-related elements. They are either a property or a requirement modeling element. The PLC modeling element does not contain any details. These basic components are extended to develop a modeling notation for MPLC.

B. Modeling MPLCs in MCNAS

For modeling MPLC first a PLC modeling element is needed which indicates the number of cores. Looking at this symbol, an automation engineer can recognize that the used PLC is a MPLC or a single-core PLC. Fig. 2MPLC modeling elementsfigure.2 depicts a sample of the three needed elements for modeling MPLC properties. For modeling the properties of MPLC a modeling element is proposed which includes the main properties of MPLC. This element contains information about CPU and RAM. For a more fine modeling the MPLC details such as the Core To Cache Scheduling Strategy, Max Number of Tasks of a Core, *Task Synchronization Strategy* and *Task Scheduling Strategy* has to be embedded into another modeling element includes these details. Probably, an automation engineer is not interested in all details of MPLC in the earlier phases of modeling a MCNAS (e.g. the exact type of MPLC will be specified in later phases of MCNAS development). Thus, details can be added optionally for latterly expanding them in a modeling tool. Modeling elements for a MPLC are connected using a line.

For modeling requirements on MPLC two modeling elements are proposed. The first modeling element deals with task requirements of MPLC and includes requirements such as Name, Task Cycle Time, Priority, Task dependency and



PLC-CX2040-Details	
Core To Cache Scheduling Strategy	EDF
Max Number of Tasks of a Core	50
Task Synchronization Strategy	Semaphors
Task Scheduling Strategy	Round Robin

PLC-CX2040		
CPU	Type	Intel i7 2715QE
	Cores	4
	Speed	2.1 GHz
	Cache	6 MB
	Instruction Set	64 bit
RAM		4 GB

Fig. 2. MPLC modeling elements

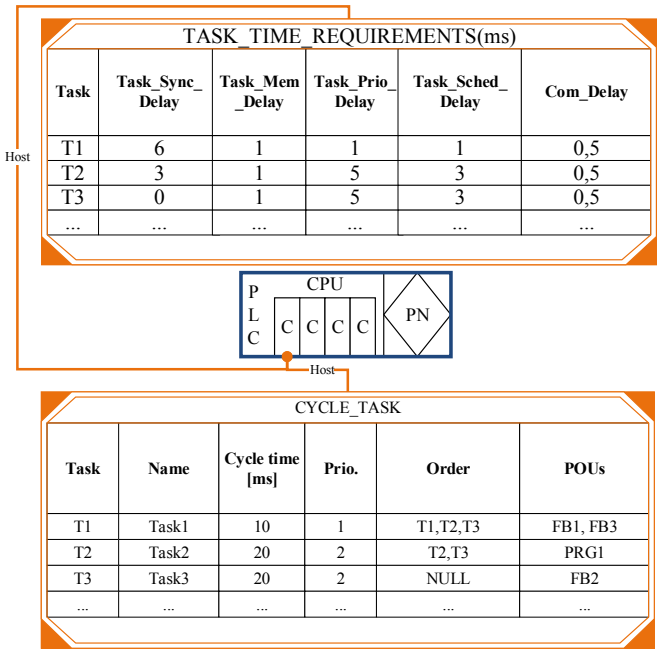


Fig. 3. Automation task requirements

assigned POUs. The other modeling element takes over the derived time requirements from the previous section. The sum of time requirements must be less than the Task Cycle Time of each task (Task Cycle Time has to be met). Both modeling elements are connected to one of the cores of the MPLC modeling element. Fig. 3Automation task requirementsfigure.3 shows the elements for modeling requirements of MPLC.

V. APPLICATION EXAMPLE

In this section, a hydraulic press as an application example from industry is introduced first to demonstrate the proposed modeling notation for MPLC subsequently.

A. Introduction of hydraulic press

In order to demonstrate the presented modeling notation, a hydraulic press used for the production of fiber boards is used as an application example, cf. Fig. 4Using MPLC modeling notation for modeling a hydraulic pressfigure.4. A full description of the whole plant is further detailed in [20].

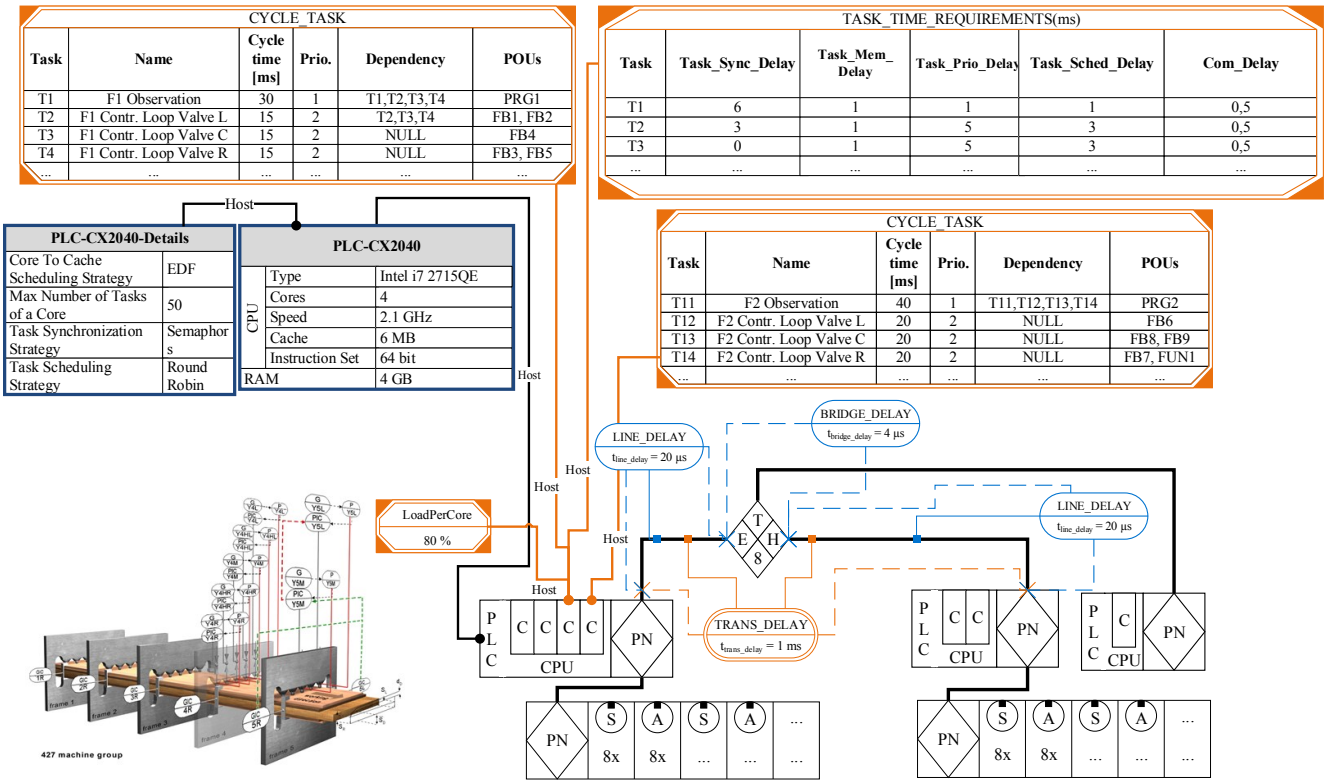


Fig. 4. Using MPLC modeling notation for modeling a hydraulic press

In wood industry, plants can consist of 3000 up to 6000 analogue or digital sensors or actuators connected via field-buses. Often, thousands of control loops with data delays of less than 10 to 20 ms need to be controlled by multiple PLCs. The hydraulic press is the most costly piece of equipment in wood industry. The press principle is simplified as follows: For the finished fiber board, material (already mixed with glue) needs to be pressed with a specific pressure in order to obtain a certain thickness related to a desired set value. In order to control both the distance and pressure on the material, certain hydraulic systems within so-called *frames* are distributed along the hydraulic press. Each frame consists of five hydraulic systems that – in turn – may consist of a proportional valve, a distance sensor and a pressure sensor. Controllers are mostly realized using proportional-integral-differential controllers used to control pressure and distance on the material. Furthermore, as the material gets thinner along the press' longitudinal direction, the controllers' distance and pressure set values not only depend on the sensors and actuators within the frame, but also on preceding and succeeding frames or frame groups. To make things worse, a press may consist of up to 70 frames and, thus, up to 500 sensors, making the engineering process even more complex. In order to master the system's complexity, appropriate support in developing and modeling the NAS needs to be provided. For the sake of clarity and simplicity, the exemplary hydraulic press consists of 16 frames.

B. Modeling the example of hydraulic press

A possible architecture for fulfilling the presented control task is shown in Fig. 4Using MPLC modeling notation for modeling a hydraulic pressfigure.4 The architecture consists of two MPLCs and one single-core PLC. The PLCs are connected via PROFINET. Beckhoff MPLCs CX 2040, CX 2030, CX 2020 have been used for the hydraulic press. The information has been derived from the data sheets available in [17]. The PLCs are switched via an Ethernet switch.

I/O-modules for 16 press frames containing the appropriate sensor and actuator interfaces are connected to the PLCs via PROFINET field bus interface. PLC-CX 2040 takes over the tasks of frames 1-8. PLC-CX 2030 (dual core) process tasks of frames 9–13 and tasks of frames 14-16 run on PLC-CX 2020 (single-core). The hardware properties of the PLC-CX 2040 are described on the left side. The descriptive modeling elements detailing PLC-CX 2030 and PLC-CX 2020 are not included in Fig. 4Using MPLC modeling notation for modeling a hydraulic pressfigure.4 because of space saving. Software requirements (CYCLE_TASK and TASK_TIME_REQUIREMENT) are shown on the top of the Figure (PLC-CX2040). The modeling elements LINE_DELAY and BRIDGE_DELAY are introduced in [3] and describe an excerpt of the time requirements of the whole MCNAS of the hydraulic press.

Using the presented modeling example, it has been shown that characteristics of MPLCs including hardware properties, hardware requirements, software properties, software requirements and time requirements on MPLC can be modeled. Providing an appropriate tool support, these characteristics can be analyzed, e.g. the analysis of the task distribution: using a given scheduling process and taking the CYCLE_TASK properties into account, the timing behavior of a software implementation executed on a MPLC can be determined.

Furthermore, effects between timing behaviors of MPLCs and, hence, the complete MCNAS can be analyzed. Using an appropriate tool being developed in future research work, the information presented above will be verified using methods from automatic verification, e.g. simulation and stochastic model-checking (cf. [2]). This enables to support engineers while dealing with the complexity of the overall automation system during engineering. Hence, the engineer is able to point out time-critical scenarios, e.g. task distributions of MPLCs or response times of NAS and their effects on the overall timing behavior. Therefore, the presented modeling approach will provide a significant step towards identifying design mistakes and violations of time requirements at an early design stage.

VI. Conclusion and future work

Multicore Programmable Logic Controllers (MPLCs) offer a higher performance because of their capability of parallel code execution in Multicore-Capable Networked Automation Systems (MCNAS). In this paper, we focused on analyzing and modeling MPLC. The result of analyzing MPLC was a set of software and hardware properties and requirements. Based on these properties and requirements a notation for modeling MPLC has been proposed. This modeling notation was demonstrated by means of an application from industry.

Future work will focus on evaluating the proposed modeling notation for MPLC and deeper investigation of MPLC considering operating system and engineering environment. Another focus will be the verification of the time behavior of MCNAS concerning software distribution on MPLC cores in case of hardware failures providing higher system reliability. Additionally, a tool support will be implemented covering the presented characteristics and requirements in order to enable automatic verification of MCNAS. The prototypical tool will be evaluated empirically both during university courses and by experts from industry to verify that both the system's failure rate and design time decrease using the presented approach in comparison to a traditional one.

References

[1] ABB's Protection Library Eliminates Need for Safety PLC(2013). [Online]. Available: http://www.automationworld.com.

[2] B. Vogel-Heuser , G. Frey, H. Hermanns, J. Folmer, L. Liu and A. Hartmanns, "Modeling of Networked Automation Systems for Simulation and Model Checking of Time Behavior", in Systems, Signals and Devices (SSD), 2012 9th International Multi-Conference, 2012, pp. 1-5.

[3] B. Vogel-Heuser, S. Feldmann, T. Werner, and C. Diedrich, "Modeling network architecture and time behavior of Distributed Control Systems in industrial plant automation", in 37th Ann. Conference of the IEEE Industrial Electronics Society. Nov. 2011, pp. 2232-2237.

[4] H. Gomaa, "Designing concurrent, distributed, and real-time applications with UML", in 23rd International Conference on Software Engineering, 2001, pp. 737-738.

[5] U. Katzke and B. Vogel-Heuser, "Design and application of an engineering model for distributed process automation", in American Control Conference, 2005, pp. 2960-2965.

[6] F. Huber, B. Schlitz, A. Schmidt and K. Spies, "AutoFocus A tool for distributed systems specification", B. Jonsson and J. Parrow, Eds. Springer Berlin Heidelberg, 1996.

[7] D. Witsch and B. Vogel-Heuser, "Modellierungsansatz fr Zeitanforderungen und Kommunikationsnetze", atp – Automation Technology in Practice, vol. 50, no. 6, pp. 44-53, 2008.

[8] G. Frey and L. Liu, "Modellierung und Simulation vernetzter Automatisierungs- und Regelungssysteme in Modelica Modeling and Simulation of Networked Automation and Control Systems in Modelica", at – Automatisierungstechnik, vol. 57, no. 9, pp. 466-476, 2009.

[9] K. T. Hansen, "Usage of multicore in automation", in IEEE International Symposium on Industrial Electronics. pp. 3784–3786, 2010.

[10] S. Byna, X.-H. Sun and D. Holmgren, "Modeling Data Access Contention in Multicore Architectures", in 2009 15th International Conference on Parallel and Distributed Systems. 2009, pp. 213–219.

[11] P.-H. Chen, C.-T. King, Y.-Y. Chang and S.-Y. Tseng, "Multiprocessor System-on-Chip Profiling Architecture: Design and Implementation", in 15th International Conference on Parallel and Distributed Systems. 2009, pp. 519-526.

[12] M. Paolieri, E. Quinones, F. J. Cazorla, G. Bernat and M. Valero, "Hardware support for WCET analysis of hard realtime multicore systems", ACM SIGARCH Computer Architecture News, vol. 37, no. 3, p. 57, Jun. 2009.

[13] J. Rosen, A. Andrei, P. Eles and Z. Peng, "Bus Access Optimization for Predictable Implementation of Real-Time Applications on Multiprocessor Systems-on-Chip", in 28th IEEE International Real-Time Systems Symposium (RTSS 2007). Dec. 2007, pp. 49-60.

[14] A. Saifullah, K. Agrawal, C. Lu and C. Gill, "Multicore Real-Time Scheduling for Generalized Parallel Task Models", in IEEE 32nd Real-Time Systems Symposium. Nov. 2011, pp. 217-226.

[15] M. Oriol, M. Wahler, R. Steiger, S. Stoeter, E. Vardar, H. Koziolek and A. Kumar, "FASA: a scalable software framework for distributed control systems", in 3rd international ACM SIGSOFT symposium on Architecting Critical Systems, ser. ISARCS12. New York, NY, USA: ACM, 2012, pp. 51-60.

[16] Y. Cui, W. Zhang, Y. Chen and Y. Shi, "A Scheduling Method for Avoiding Kernel Lock Thrashing on Multi-cores", in IEEE International Conference on Parallel and Distributed Systems. pp. 17-26, 2010.

[17] Beckhoff (2012). [Online]. Available: http://www.beckhoff.de/default.asp?embedded_pc/.

[18] G. Lipari, G. Lamastra and L. Abeni, "Task synchronization in reservation-based real-time systems", IEEE Transactions on Computers, vol. 53, no. 12, pp. 1591-1601.

[19] International Electrotechnical Comission. Programmable controllers - Part 3: Programming languages,Reference: IEC 61131-3 ed. 3.0 .

[20] B. Vogel-Heuser, Automation in the Wood and Paper Industry. Berlin Heidelberg: Springer, 2009, pp. 1015-1026.