Towards Plug-n-Play Numerical Control for Reconfigurable Manufacturing Systems

Vuk Lesi
Dept. of Electrical and Comp. Eng.
Duke University
Email: vuk.lesi@duke.edu

Zivana Jakovljevic
Faculty of Mechanical Engineering
University of Belgrade
Email: zjakovljevic@mas.bg.ac.rs

Miroslav Pajic
Dept. of Electrical and Comp. Eng.
Duke University
Email: miroslav.pajic@duke.edu

***Abstract*—Modern manufacturing systems require fast and effective adaptation to fluctuating market conditions and product diversification. This high level adaptability can be achieved through the utilization of Reconfigurable Manufacturing Systems (RMS), which should be based on modular equipment that is easily integrated, scalable, convertible in terms of functionality, and self diagnosable. RMS also necessitate the use of a dynamic controller architecture that is distributed, fully modular, and self configurable.**

In this paper, we present a control system design approach for reconfigurable machine tools through the use of modularized and decentralized CNC control. Specifically, we investigate design challenges for Plug-n-Play automation systems, where new system functionalities, such as adding new axes in existing CNC units, can be introduced without significant reconfiguration efforts and downtime costs. We propose a fully decentralized motion control architecture realized through a network of individual axis control modules. Reconfiguration of motion control systems based on this architecture can be achieved by only presenting the controller on each axis with information about machine configuration and the type of axis. This effectively enables modularity, reconfigurability, and interoperability of the machine control system. Finally, we present an implementation of the decentralized architecture based on the use of a real-time operating system, wireless networking, and low-cost ARM Cortex-M3 MCUs; we illustrate its effectiveness by considering machining of a standard test part defined in ISO 10791-7 using a software-in-the-loop testbed.

I. Introduction

As a result of globalization and fluctuating market conditions, production companies are facing the challenge of manufacturing individualized products according to customer demands [1]. Increase in product variety has been accompanied by a decrease in lot sizes, up to the level of one-off products. Thus, profitable manufacturing in such conditions requires fast and effective adaptation of manufacturing systems and their accommodation to quick changes of diversified products. This high level adaptability can be achieved through development and implementation of Reconfigurable Manufacturing Systems (RMS), which are based on modular equipment that is easily integrated, scalable in terms of production capacity, convertible in terms of functionality, and self diagnosable [2], [3], [4], [5].

Manufacturing resources modularity, changeability and interconnectivity has to be achieved at the mechanical (i.e., hardware), as well as at the machine control system level. The transition from traditional hierarchical automation pyramid to distributed/decentralized control systems [6] is crucial for RMS. The most important enabling technologies for ad-hoc configuration of manufacturing resources are Internet of Things (IoT) and Cyber-Physical Systems (CPS) within the Industrie 4.0 framework [7], [8], [9], [10].

The major components of RMS are Reconfigurable Machine Tools (RMTs). RMTs provide large possibilities for adaptation to manufacturing of diverse products through change of position and/or orientation of modules, modules swapping, and adding additional axes and/or spindles as needed [11]. Besides adaptability to changing needs, RMTs enable higher utilization of resources, and accordingly significant cost reductions. Nevertheless, the time needed for machine tool reconfiguring is a limiting factor in RMTs application. It includes not only the physical reconfiguration of the mechanical sub-system but also the reconfiguration of the machine's control system software.

Currently used control systems for machine tools are based on Computer Numerical Control (CNC) units that are not modular, and which usually include a large number of different configuration options that are rarely utilized to full extent. This introduces a design time tradeoff between reconfigurability and resources consumption; in order to support reconfigurability control units are significantly over-designed, leading to a low resource utilization for the most of the unit's operational lifetime. Furthermore, with the existing CNC units, reconfiguration requires extensive time intervals to adapt control system software to the new configuration of the machine, thus having negative influence on the responsiveness of the manufacturing system. Consequently, RMTs necessitate the use of a dynamic controller architecture [11] that is distributed, fully modular, and self-configurable.

Recently, open architecture CNC systems have been introduced to provide a desired level of reconfigurability, while offering advanced measurement and control functionality for RMTs [12]. These systems are implemented in a centralized manner, usually based on PCs running a Real-Time Operating System (RTOS). In these architectures, each machine axis has its own control module on the centralized platform and adding new axis implies control system hardware changes. In addition, implementation of a new machine configuration in open CNC requires control system software adaptation where programming skills and significant user intervention are

978-1-5090-1314-2/16/$31.00

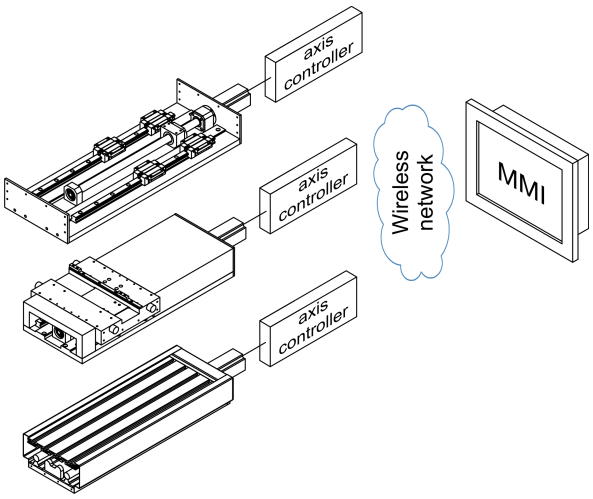


Fig. 1. Decentralized CNC system – Numerical Control Kernel is not executed on a centralized controller; rather all NCK functionalities are executed on individual axis controllers.

needed [13], [14]. Finally, self-configurability is not achieved at the motion control level [15], and changes in machine configurations, if not defined in advance, are very challenging, significantly increasing development time and costs.

In this paper, we investigate design challenges for Plug-n-Play (PnP) automation systems, where new system functionalities, such as adding new axes in existing CNC units, can be introduced without significant reconfiguration efforts and downtime costs. Specifically, we focus on CNC control units and propose a fully decentralized motion control architecture realized through a network of individual axis control modules (as shown in Fig. 1). With the proposed design, reconfiguration of motion control systems is carried out by only presenting the controller on each axis with information about machine configuration, and the type of axis (e.g., X, Y, Z) that the controller is running. This effectively enables modularity, reconfigurability, and interoperability of the machine control system. Furthermore, we present an implementation of the decentralized architecture on low-cost ARM Cortex-M3 based boards [16], and show that, unless highly accurate CNC machining is considered, low-power wireless communication can be used to coordinate axis controllers. Finally, as a proof of concept, we evaluate system performance within a Software-In-the-Loop (SIL) framework by considering machining of a standard test defined in ISO 10791-7 [17].

This paper is organized as follows. We start by presenting the conventional architecture used in CNC machines (Section II). Section III defines the architecture of the proposed decentralized solution for the Numerical Control Kernel. In Section IV, we describe design challenges and system implementation, while system evaluation is presented in Section V. Finally, in Section VI, we provide discussion and avenues for future work.

II. State-of-the-Art CNC Architecture

A conventional CNC system comprises of three functional components: Numerical Control Kernel (NCK), Programmable Logic Control (PLC), and Man Machine Interface (MMI) [18]. The functional decomposition is presented in Fig. 2. NCK is the functional component central to the machining process, used to translate formal specification of a physical part to be manufactured into motion of the axes. PLC unit controls other operational aspects of the machining process excluding the motion path calculation and motion control. Finally, MMI provides an interface through which the operator can input part specifications, alter machine settings, and monitor the status/progress of the machining process. Unless otherwise noted, in this paper we focus on the Numerical Control Kernel.

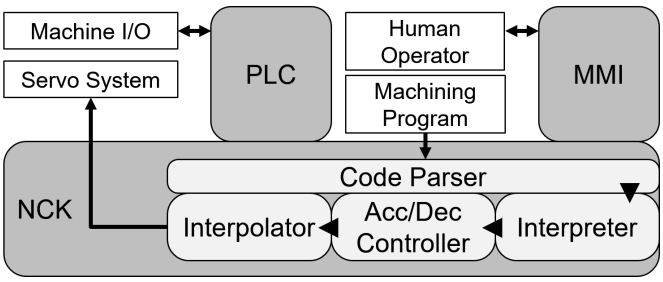


Fig. 2. Standard architecture of CNC systems.

Using the part specifications in ISO 6983 code (G-code) [19], which we will refer to as *part program*, the parser is responsible of checking code correctness and passing parsed commands to the interpreter.[1] Interpreter extracts geometric specifications such as end positions of linear segments, toolpath radius and commanded feed-rates and, assuming acceleration/deceleration control before interpolation (ADCBI) is used, conveys them to the acceleration/deceleration controller.[2] Given its input, the acceleration/deceleration controller constructs a velocity profile of every path segment, before feeding it to the interpolator. Based on the velocity profile, the interpolator generates incremental reference position values for each axis. Finally, reference position increments are integrated and the result is sent to the position controllers in charge of controlling the axes motors. Note that all stages except parsing are performed on a periodical basis; position control is performed within the motor drive for each axis.

The conventional architecture limits efficient system reconfigurations for several reasons. NCKs are usually designed and tuned to support a certain number of axes and the number of utilized axes depends on the machine configuration. Reconfiguration of a machine tool in terms of an axis pose change or installation of additional axes requires complicated reconfigurations of control and mechanical systems' hardware. Secondly, components of the system are usually custom built or utilize very specific settings, which severely limits interoperability of equipment coming from different manufacturers. In addition, the system architecture is centralized, meaning that on failure of the controller executing the NCK, the

[1]It is worth noting here that our approach does not specifically depend on the use of G-code; our results easily translate to newer programming and execution paradigms such as STEP-NC.

[2]More details about differences between acceleration/deceleration control before and after interpolation can be found in [18], Chapter 4.

machine is subject to an erroneous state that requests major overhaul, readjustments, etc. Modularity is not enforced on the axis level since the centralized execution scheme disables modularization on any level higher than the lowest actuation stage involving the motor drives and axes' motors. All of these properties embedded within the conventional architecture effectively disable design of Plug-n-Play automation systems.

On the other hand, in existing CNC systems, position controllers for different axes are independent due to the absence of feedback from the position control back to previous stages within the NCK, along with the absence of any real-time inter-axis coupling. Consequently, this enables implementation of a decentralized NCK as a distributed network of components, where each of the axis controllers locally executes all components necessary to specify its actions (Fig. 1). In the following section, we discuss an architecture where execution of all corresponding NCK functionality is done locally at the axis level, and show that it facilitates modularity, reconfigurability, and interoperability of CNC systems.

III. Decentralized CNC Architecture

To address limitations of existing CNC units when used in RMS, we propose a decentralized NCK architecture (shown in Fig. 1) that facilitates functionality expansion, such as adding more degrees of freedom in the machining process or swapping axes to obtain the desired workspace. Specifically, in the new architecture each axis controller contains *all functionalities of a standard NCK* – i.e., Interpreter, Acc/Dec controller, and Interpolator. The proposed decentralized architecture has been enabled by the fact that there is no tight inter-axis coupling or feedback between functional stages in the conventional architecture. This architecture allows partitioning of the traditional CNC system over individual axes into independent modules by distributing the NCK homogenously on the axis level. This implies that code parsing and interpretation, control of tool acceleration/deceleration, toolpath interpolation, and finally tool position control are performed independently for each axis, on behalf of the respective axis controller.

Note that in this design geometry specifications of the desired part remain the same on control units for all axes; this is also the same part program that would be input in a conventional (i.e., centralized) system. Additionally, we exploit wireless communication between axis modules, as is shown in Fig. 1. Due to the fact that only limited low-rate communication is required, we utilize low-power IEEE 802.15.4 compatible wireless networks. Reliability of synchronization over IEEE 802.15.4 can be guaranteed given that payloads for sync beacons are very small, and the periodicity of synchronization is low relative to periods of other important tasks (such as position control).

However, moving from a centralized control of all axes to decentralized axes control introduces the implementation challenge of providing synchronous execution of the part program on all axes controllers. To maintain machining accuracy, tight timing synchronization constraints have to be imposed on code execution between each of the axes' controllers

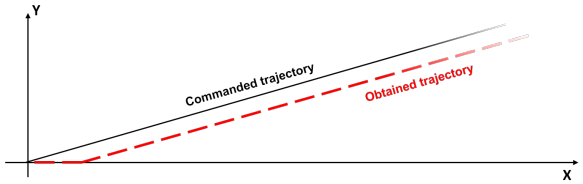


Fig. 3. Example showing trajectory deviation, from the commanded trajectory, when the Y axis controller lags in execution to the X axis controller by a constant delay.

as any synchronization error introduces additional accuracy impairment. To illustrate this, consider Fig. 3 that presents a simple 2D example of a linear segment. As shown in the figure, significant deviation occurs in the obtained tool trajectory compared to the commanded path if one of the axes lags after the other by a (constant) delay.

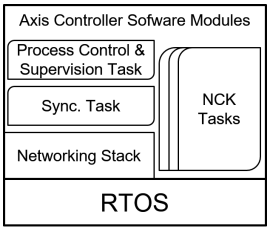
A. Software Design Requirements for PnP CNC

To support PnP operation of the proposed architecture as a network of axis controller modules, software design is based on a task set constructed to modularize the software aspect of the system. Identification of essential responsibilities of a single axis controller results in the software architecture depicted in Fig. 4(a).

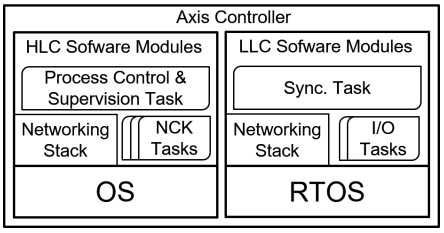
Interpretation of a part program, acceleration/deceleration control, and trajectory interpolation are performed within a set of tasks with a common goal to prepare reference position samples. Results of executions of one task are commuted to the next stage through intermediary buffers. The interpolating task feeds an I/O interface to forward reference position samples from the buffer to the position controller, which is integrated within the axis drive (e.g., servo drive). All axis drives are isolated from the CNC controller and they automatically close position, velocity and current loops with motors for their respective axes.

Note that trajectory interpolation must execute at the same rate at which the position control loop is closed. Thus, no deadlines of this task may be missed since this would imply staling of the reference value on the input of the position controller, and hence potentially incurring machining inaccuracy or physical damage to the mechanical components. This imposes hard real-time constraints on all tasks involved in the operation of the NCK. In other words, system design has to provide guarantees that the interpolator task would never starve, i.e., its input buffer should be charged in a timely manner relative to the discharge rate. Therefore, to ensure timeliness of the NCK tasks' executions and provide real-time guarantees for axes control, these tasks are executed on top of an RTOS.

Additionally, to ensure synchronous execution of the part program on individual axis controllers in the decentralized architecture, a synchronization mechanism needs to be implemented on top of the RTOS. Implemented as a high-priority periodical synchronization task, it provides periodical clock offset compensation with respect to the global notion of time present in the network, without effecting schedulability



(a) Unified single axis controller capable of scheduling all necessary tasks;



(b) Load balancing over a computationally capable Higher-Level Controller (HLC) and a microcontroller based Lower-Level Controller (LLC).

Fig. 4. Software Modules of Axis Controllers.

of other tasks. Finally, part program downloading, controller configuration, and process supervision are realized through an aperiodic task that communicates with a higher planning entity. As the timing of this task's execution is not as critical, we consider it as a soft real-time task, with soft guarantees for resource allocation.

Potential high complexity of the interpolation algorithm (e.g., NURBS interpolation) imposes hard computational requirements on the underlying hardware platform. Further on, the low period of the interpolating task (of the order of a millisecond corresponding to the position control loop sampling time) incurs significant overhead in task switching. Finally, the synchronization task introduces additional load on the resources by utilizing the networking stack. To support standard (and more complex) interpolation methods while ensuring that tasks with hard real-time constraints are executed in a timely manner, the task set of each axis controller can be shared among a Higher-Level Controller (HLC) and a Lower-Level Controller (LLC), as presented in Fig. 4(b).

Specifically, calculating reference position values can be segregated into the HLC for each axis, which does not have to run under real-time constraints, and need not communicate with HLCs of other axes. This is possible when no feedback from the position control exists back to the interpolator. Here, the HLC computes the trajectory by implementing code interpretation, acceleration/deceleration control and interpolation. Under these conditions, the LLCs remain responsible of maintaining inter-axis synchronization and timely adjusting reference position input of the position controller for their respective axis. Reference position values calculated by the HLC are charged into a buffer between the HLC and the LLC for each axis. The result of this load balancing is that the resource demanding NCK-related tasks, synchronization task and the task representing interface to the planning level are now shared among a low-cost microcontroller based LLC, and a HLC with timeliness requirements that can be relaxed due to a higher computation power.

The LLC's task set reduces to two computationally light tasks, besides the synchronization task. One task obtains reference position values from the higher-level controller for its respective axis through a low latency local link. The other task forwards these reference values to the position controller. Guaranteeing synchrony in controlling all axes of the machine remains embedded within the LLCs. To maintain features such as fast part code downloading, easy configuration changes and prompt process status supervision over the air, these aperiodic tasks are also mapped into the HLC. For these purposes, communication between HLCs to a planning level gateway can be realized through a wireless protocol with a higher bandwidth than 250 Kbit/s offered by IEEE 802.15.4 standard (e.g., IEEE 802.11). Nevertheless, a single hardware platform can still act as an autonomous controller with no need for an additional higher-level controller, as we described, if simple interpolation algorithms are to be used, or if a more computationally powerful platform with real-time guarantees is to be the base of an axis controller.

IV. System Implementation

In this section, we describe our design of the decentralized motion control architecture. We start by describing specific characteristics of the implemented task set on the utilized platform, before focusing on axes synchronization and control task implementation.

A. Task Set Specification

We implemented our axis controllers on ARM Cortex-M3 microcontroller unit (MCU) based development boards [16], with the MCUs running at 96 MHz. These modules communicate over IEEE 802.15.4-compliant wireless transceivers (Microchip MRF24J40MA 2.4 GHz [20]). On top of this platform, we run Nano-RK RTOS [21]. Due to the limitations of the hardware platform, we are unable to run the full software application model on a single CPU as described in Fig. 4(a). Our platform could support simple interpolation algorithms. Furthermore, the task switching overhead is too large to allow us to have the interpolating task execute with $1ms$ period, which is a typical sampling interval for position controllers, while ensuring that no deadlines are missed for all remaining tasks. Similarly, although executed with a low rate, long execution times of the synchronization task caused by lengthy network packet parsing, could potentially affect schedulability, and real-time execution guarantees, of other tasks.

These limitations lead us to the software architecture shown in Fig. 4(b). For purposes of evaluating the decentralized CNC architecture, our real-time application on the LLC comprises of a low-priority synchronization task and a periodic hardware

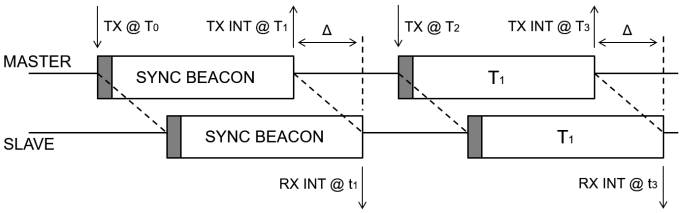


Fig. 5. In-band, one-way communication, master initiated synchronization sequence enables sub-$8\mu s$ timing synchronization without additional hardware or Start of Frame Delimiter interrupt feature of the radio (SFD shaded on the beginning of every packet).

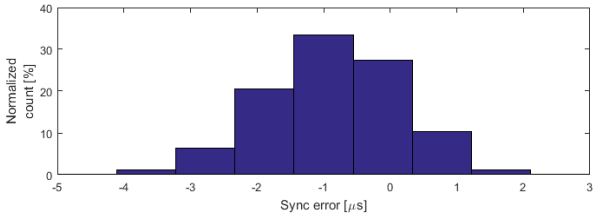


Fig. 6. Distribution of timing synchronization errors between $1ms$ periodic interrupt handler invocations between a master and a slave node over $200sec$ (sample mean is $-852.2ns$, standard deviation is $1.1483\mu s$). Over $12hrs$ of measurements, the synchronization error never exceeds $\pm 8\mu s$.

timer interrupt service routine of the highest priority. These tasks are described in the rest of this section.

B. Clock Synchronization Task

To implement timing synchronization on our platform, we faced several challenges. Our design strategy and focus on the use of off-the-shelf platforms, limits us from using additional hardware for out-of-band synchronization as in [22]. Further limited by a reduced multiplexed interrupt line on the utilized radio module, we were unable to implement any of the typical approaches to synchronization in wireless sensor networks (e.g., [23], [24], [25]). Since the radios we use cannot interrupt the MCU on decoding of the Start of Frame Delimiter (SFD) of the standard 802.15.4 frame, and have no timestamping capability on the physical layer, we construct a very simple alternate technique to synchronize the global clocks of our controllers, avoiding any additional hardware. We show that synchronization errors obtained with the use of our scheme are comparable to the aforementioned approaches considering factors such as the presence of additional hardware, access to physical layers of communication protocols, and implementation complexity. More importantly for the application at hand, synchronization errors that have been proven to be tolerable using simulation in our specific application domain are also well achievable by our synchronization algorithm.

Fig. 5 depicts the synchronization process initiated by one node in the network, which we refer to as the master node. We denote the global reference time (i.e., the master's local time) with capital letter T, and the slave's time with lowercase letter t, both with appropriate indices. At time T_0, the master node initiates the synchronization process by sending an empty beacon. After initiating the transmission sequence in its radio, it waits for an interrupt indicating that the last byte of the packet has been sent. At the occurrence of this interrupt, the master takes note of the current time T_1. At $t_1 = T_1 + \Delta$, the last byte of this packet is decoded on the slave's radio, which the slave's MCU is informed on via an interrupt. The slave takes note of this event by capturing its local time t_1.

Also, the master sends a second message, at time T_2, that includes the time when the last byte of the previous message was sent. The last byte of the packet containing this timestamp is in air at time T_3. The slave node decodes the last byte of this packet at $t_3 = T_3 + \Delta$. Given that, on the employed platforms, the distribution of link delays Δ between the occurrences of interrupts on the transmitting side and on the receiving side has a measured mean of $37.54\mu s$ and a very low standard deviation of $73.96ns$, the slave node can adjust its global time by compensating for the offset calculated as $\delta = T_1 - t_1 - \Delta$.

In our experiments, we showed that this simple synchronization scheme results in sub-$8\mu s$ master-slave synchronization error, which is in the worst-case half of the delay between periodical interrupt handler activations on controllers of different axes.[3] Fig. 6 presents the distribution of the synchronization errors for position control updates between the master and a slave axes, measured on a $200s$ timespan. In addition, on a 12-hour timespan, the observed synchronization error never exceeded $8\mu s$. In Section V, we show that this synchronization error satisfies machining accuracy requirements.

C. Control Task

For purposes of evaluating the decentralized CNC architecture, each axis controller stores a precomputed trajectory obtained from the interpolation routine. Long execution times of the synchronization task, extended by parsing of incoming time synchronization packets, and high overhead in frequent switching to and from a task with $1ms$ period may result in violation of hard real-time execution constraints. Thus, we implemented the control functionality of an axis controller on a lower level. The periodical timer interrupt triggers a handler every $1ms$. Within the timer interrupt handler, a reference position sample is forwarded from the top of the buffer with reference position samples to the position controller. This low-level implementation ensures that timeliness requirements are met, which would not be the case if forwarding reference position to the position controller remained a responsibility of a task under the government of the RTOS. To give maximum priority to the control task, the Nested Vectored Interrupt Controller (NVIC) of the Cortex-M3 core is configured to guarantee periodical activation that preempts execution in any other context in favor of the reference position forwarding.

V. Evaluation

Evaluation of the control system design on a real physical machining process is often difficult for several reasons. Safety

[3]Note that, since we do not have a multi-hop network, the synchronization delay does not increase with each hop in the network. Thus, the synchronization error between any two 'slave' axes is less than $2 \cdot 8\mu s$.

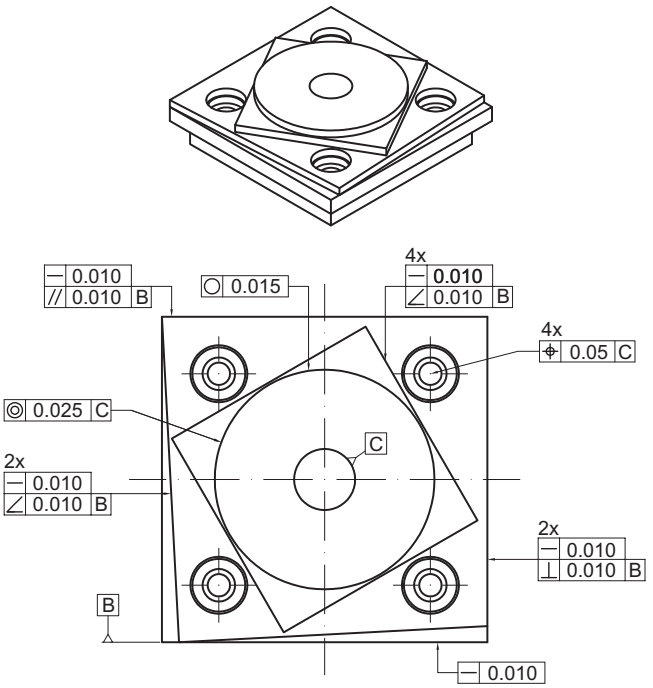


Fig. 7. ISO 10791-7 test part used as simulation input.

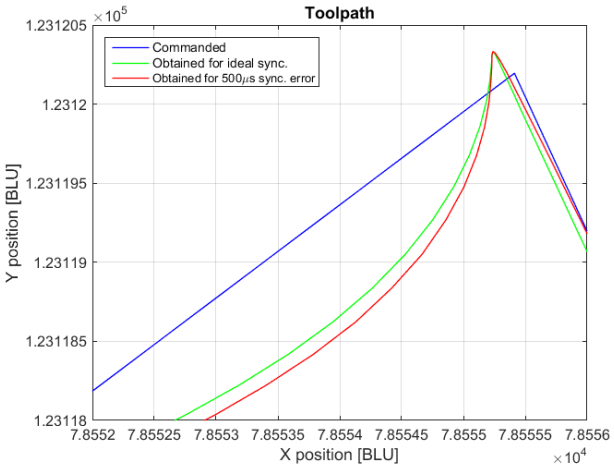


Fig. 8. Magnified excerpt of the tool trajectory near the top corner of the tilted square feature of the test part during simulation. Reference position tracking is effectively reduced when inter-axis synchronization is not ideal, but most of the inaccuracy is induced by non-ideal position control. (Axes scaled in Basic Length Units of the modeled servo system and axes: $1BLU = 1.27\mu m$)

conditions required during normal machining operations are hard to guarantee in both stages of low-level controller design and high-level trajectory features planning. Impacts of the proposed decentralization on machining accuracy is easier to estimate in simulation than on a physical object, since the latter involves complex measurements with specialized instrumentation. Thus, to evaluate system performance, we first used simulation with all components implemented either as simulation block diagrams in Simulink or as computation code in Matlab. Secondly, we designed a Software-In-the-Loop (SIL) testbed, at the point of the reference trajectory input to the position controller, and we employed our physical platforms, running software described in the previous section, as axis controllers. This allowed for a closer insight on the behavior of the system when real non-ideally synchronized hardware is involved. The two stages of evaluation are explained in the following subsections.

A. Simulation of the Decentralized Axes Control

We implemented a model of a two axis system in simulation and evaluated accuracy of a fully simulated system assuming both ideal and non-ideal inter-axis synchronization. Specifically, we designed position control and axis mechanics in Simulink and the trajectory calculation algorithm in Matlab. The toolpath calculation comprises the interpreter, acceleration/deceleration controller and interpolator. We adopted the DC servo position control loop model from [26] and extended it to capture measurement noises. We replaced the proportional velocity loop controller with a PI controller to enhance machining accuracy of the modeled servo system. In Simulink, we run two independent instances of the position control model to capture behavior of a two-axes machine. In this simplified model, all mechanical properties of axes are abstracted through static resistant torque.

The proposed decentralization adds temporal uncertainty in appearance of reference position values at the output of each axis controller, i.e., the input of the position controller. The input of the position control is a time-discrete signal updated every interpolation period. Despite the delay in the operation of one axis in reference to the other being introduced at the very input of the position controller, it is impractical and incorrect to model it at that point under the synchronous execution semantics of Simulink. We model the delay at the analog position output of the position control loop, which enables us to observe the behavior of the axes in simulation (as a whole) when there exists a constant or variable time delay due to synchronization error between the axes.

Reference trajectory is fed from the output of a simulation of the interpreter, acceleration/deceleration controller, and interpolator precoded in Matlab. Given a set of end points if linear segments are in question, and additionally the trajectory radius if a circular path is desired, our code implements a simple look-ahead acceleration/deceleration controller and reference-word interpolator using improved Tustin method for circular segments.[4]

We evaluate our approach by simulating the machining process of a standard test part from ISO 10791-7 [17], depicted in Fig. 7. We estimate straightness and circularity errors by fitting a regression curve (first order for linear segments) to the segment which we want to analyze, and calculating the distance between the farthest point of the obtained trajectory of the tool and the respective regression curve. This procedure provides us with the maximum single-sided tolerance. In a simple and safe overapproximation, which does not effect our

[4]Chapters 3 and 4 of [18] discuss in detail interpolation and acceleration/deceleration control respectively.

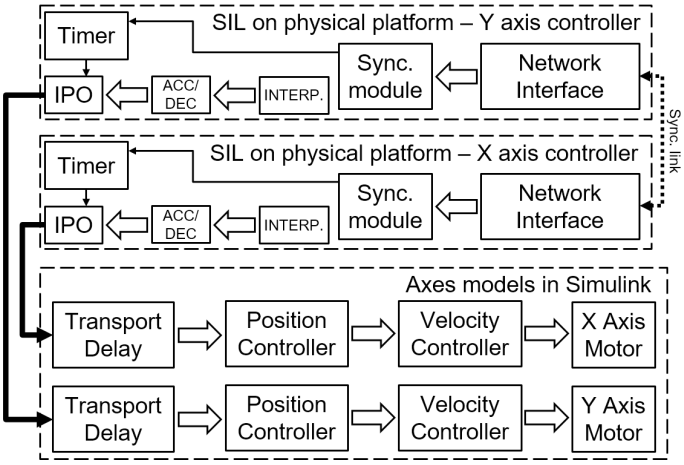


Fig. 9. Software-in-the-Loop Evaluation Framework. Two independent axis controllers execute NCK tasks for their respective axis and synchronize via a wireless network link. Two independent position controllers are simulated in Simulink.

conclusions, this tolerance is doubled to capture inaccuracies on both sides of the regression curve. Angularity, perpendicularity and parallelism are evaluated by estimating how well the obtained toolpath fits into tolerable limitations determined by the slope of commanded linear features relative to the slope of the datum defined by the standard. Concentricity is tested by comparing the location of centers of regression curves (second order) fitted to obtained trajectories with the center of the datum defined by the standard. Positioning accuracy for four holes along the corners of the test part is evaluated by calculating the deviation in distance between their centers obtained from the simulation of the toolpath, from the expected distance from the datum (center of middle hole).

Fig. 8 shows a magnified excerpt of the toolpath trajectory near the top corner of the tilted square feature of the test part. Reference tracking capability of the servo system is effectively reduced when inter-axis synchronization is not ideal. From the figure, it can be concluded that most of the inaccuracy is a result of non-ideal position control, rather than inter-axis sync error. Results quantify this occurrence.

Table I provides some of the tolerances defined in ISO 10791-7, measured in our simulation. The case with perfect inter-axis synchronization is compared to three simulations with different values for a constant delay between the two axes. It can be concluded that, under given conditions, standard tolerances can be well satisfied with the synchronization scheme described in Section IV. We thus expect that results obtained within our SIL framework will only show slight accuracy degradation compared to the ideal case of perfect parallel execution of NCK tasks for two axes. We prove so in the following subsection.

B. Evaluation Under SIL Framework

In the SIL simulation setup, we use two of our platform modules as axis controllers in the simulation loop. We record the moments of activation of the timer interrupt handler on both axis modules using a Digital Storage Oscilloscope (DSO). This enables us to precisely capture the jitter in the period under which the reference position values appear on the output of our axis controller for each axis. Our platform instances are placed in the simulation loop by initializing the transport delay blocks in Simulink with the exact synchronization errors directly obtained from the DSO measurements. The SIL evaluation framework is shown in Fig. 9.

	Ideal inter-axis sync.	Additional impairment at: $100\mu s$ sync. error	$500\mu s$ sync. error	$1ms$ sync. error
B datum straightness $[\mu m]$	1.15636	0.00001	0.00001	0.00001
bottom chamfer straightness $[\mu m]$	1.09598	0.20398	1.01352	2.02547
bottom chamfer angularity $[\mu m]$	1.31744	0.01020	0.79963	1.90338
$\oslash$108 cyl. circularity $[\mu m]$	4.12805	1.03771	5.02564	10.01034

TABLE I
SOFT SIMULATION RESULTS. NOTICE THAT ADDITIONAL ACCURACY IMPAIRMENT FOR SUB-$100\mu s$ SYNCHRONIZATION ERRORS IS VERY SMALL.

	Ideal inter-axis synchronization	Additional impairment in SIL simulation
B datum straightness $[\mu m]$	1.15636	0.00000
bottom chamfer straightness $[\mu m]$	1.09598	0.08535
bottom chamfer angularity $[\mu m]$	1.31744	0.06769
$\oslash$108 cyl. circularity $[\mu m]$	4.12805	0.01901

TABLE II
SOFTWARE-IN-THE-LOOP SIMULATION RESULTS. ADDITIONAL ACCURACY IMPAIRMENT IS VIRTUALLY NEGLIGIBLE GIVEN THE SYNCHRONIZATION ERRORS OBTAINED IN HARDWARE PLATFORMS.

Table II presents the same excerpt of measured tolerances for the test part from ISO 10791-7 as Table I, but shows results obtained from the SIL simulation. These results illustrate full feasibility of decentralization of a CNC system governed by previous discussions. Moreover, at the price of negligible accuracy impairment, introducing a wireless synchronization scheme reduces the physical interface of an axis controller to a power supplying connection. Wireless communication among axis controllers and towards a higher-level planning entity rises the level of reconfigurability and greatly broadens the self-configuration capabilities of a CNC system.

VI. DISCUSSION AND FUTURE WORK

In this paper, we have presented a step towards reconfigurable manufacturing through modularized and decentralized CNC control. We have identified central issues with the transformation from conventional, centralized to distributed and decentralized execution of the Numerical Control Kernel. We have proposed a decentralized system design that exploits available system resources based on the real-time execution

of suitable Numerical Control Kernel tasks. We have analyzed a real-time implementation of the architecture and presented an embedded solution on which we tested the effects of this transformation. Finally, we have shown decentralization to be feasible when a sufficiently performing synchronization technique is employed among the axis controllers.

The proposed design begins with a CNC controller that accepts its inputs specified in ISO 6983 (i.e., G-codes). Nevertheless, our results easily translate into newer machine tool control languages such as STEP-NC. The toolpath control level we targeted in our discussion is based on an architecture that can well support machine-independent and portable specifications, and is extendable to accept and provide additional information on the machining process such as on-machine simulation, feedback from on-machine sensors, etc.

Additionally, we have presented a new synchronization scheme for synchronizing global notions of time on embedded platforms in low-power wireless networks. This has enabled us to implement axis controllers with all widely available, off-the-shelf radio and MCU modules, of which the radio module does not offer the Start of Frame Delimiter interrupt. This timing synchronization scheme is applicable to a wide range of applications outside the scope of this paper.

As a part of our future work, we intend to formalize the transition from centralized execution of NCK tasks on a single CNC controller, to parallel execution on axis controllers. This transition has to formally specify semantics of parallel execution that guarantees system output preservation after decentralization, given that the input part program remains the same. Furthermore, the formal semantics needs to capture the effects of synchronization errors and support tradeoff analysis between the desired machining accuracy and requirements for axes synchronization. Finally, an avenue for future work is to deploy our axis controllers in a real physical system of axes, and evaluate its performance.

ACKNOWLEDGMENT

This work was supported by the NSF CNS-1505701 grant, and the Intel-NSF Partnership for Cyber-Physical Systems Security and Privacy. In addition, it was partially supported by the Serbian Ministry of Education, Science and Technological Development, research grant TR35020.

REFERENCES

[1] F. Jovane, Y. Koren, and C. Bor, "Present and future of flexible automation: Towards new paradigms," *CIRP Annals - Manufacturing Technology*, vol. 52, no. 2, pp. 543 – 560, 2003.

[2] H. A. ElMaraghy, "Flexible and reconfigurable manufacturing systems paradigms," *International Journal of Flexible Manufacturing Systems*, vol. 17, no. 4, pp. 261–276, 2006.

[3] H. ElMaraghy, G. Schuh, W. ElMaraghy, F. Piller, P. Schnsleben, M. Tseng, and A. Bernard, "Product variety management," *CIRP Annals - Manufacturing Technology*, vol. 62, no. 2, pp. 629 – 652, 2013.

[4] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, and H. V. Brussel, "Reconfigurable manufacturing systems," *CIRP Annals - Manufacturing Technology*, vol. 48, no. 2, pp. 527 – 540, 1999.

[5] Y. Koren and M. Shpitalni, "Design of reconfigurable manufacturing systems," *Journal of Manufacturing Systems*, vol. 29, no. 4, pp. 130 – 141, 2010.

[6] H. ElMaraghy and L. Monostori, "Variety management in manufacturing cyber-physical production systems: roots, expectations and R&D challenges," *Procedia CIRP*, vol. 17, pp. 9 – 13, 2014.

[7] H. Kagermann, W. Wahlster, and J. Helbig, "Recommendations for implementing the strategic initiative INDUSTRIE 4.0." 2013. [Online]. Available: http://www.acatech.de/fileadmin/user_upload/Baumstruktur_nach_Website/Acatech/root/de/Material_fuer_Sonderseiten/Industrie_4.0/Final_report__Industrie_4.0_accessible.pdf

[8] X. Yao and Y. Lin, "Emerging manufacturing paradigm shifts for the incoming industrial revolution," *The International Journal of Advanced Manufacturing Technology*, pp. 1–12, 2015.

[9] M. Pajic and R. Mangharam, "Embedded virtual machines for robust wireless control and actuation," in *16th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2010, pp. 79–88.

[10] M. Pajic, A. Chernoguzov, and R. Mangharam, "Robust architectures for embedded wireless network control and actuation," *ACM Transactions on Embedded Computing Systems*, vol. 11, no. 4, pp. 82:1–82:24, Jan. 2013.

[11] R. Landers, B.-K. Min, and Y. Koren, "Reconfigurable machine tools," *CIRP Annals - Manufacturing Technology*, vol. 50, no. 1, pp. 269 – 274, 2001.

[12] G. Pritschow, Y. Altintas, F. Jovane, Y. Koren, M. Mitsuishi, S. Takata, H. van Brussel, M. Weck, and K. Yamazaki, "Open controller architecture – past, present and future," *CIRP Annals - Manufacturing Technology*, vol. 50, no. 2, pp. 463 – 470, 2001.

[13] D. Milutinovic, M. Glavonjic, N. Slavkovic, Z. Dimic, S. Zivanovic, B. Kokotovic, and L. Tanovic, "Reconfigurable robotic machining system controlled and programmed in a machine tool manner," *The International Journal of Advanced Manufacturing Technology*, vol. 53, no. 9, pp. 1217–1229, 2010.

[14] H. Huang, G. Chi, and Z. Wang, "Development and application of software for open and soft multi-axis EDM CNC systems," *The International Journal of Advanced Manufacturing Technology*, pp. 1–12, 2016.

[15] L. Zhou, J. L. Yuan, P. Gao, and Y. H. Ren, "A new architecture of open CNC system based on compiling mode," *The International Journal of Advanced Manufacturing Technology*, vol. 73, no. 9, pp. 1597–1603, 2014.

[16] NXP Semiconductors N.V. (2009, Feb.), "LPC1768/66/65/64 32-bit ARM Cortex-M3 microcontroller," [Online]. Available: http://www.nxp.com/documents/data_sheet/LPC1768_66_65_64.pdf [Jan 07, 2016].

[17] *Test conditions for machining centres – Part 7: Accuracy of finished test pieces*, ISO 10 791-7:2014.

[18] S.-H. Suh, S. K. Kang, D.-H. Chung, and I. Stroud, *Theory and Design of CNC Systems*. London, UK: Springer-Verlag London, 2008.

[19] *Automation systems and integration – Numerical control of machines – Program format and definitions of address words – Part 1: Data format for positioning, line motion and contouring control systems*, ISO 6983-1:2009.

[20] Microchip Technology Inc. (2008, Jan.), "MRF24J40MA 2.4 GHz IEEE Std. 802.15.4™ RF Transceiver Module," [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70329b.pdf [Jan 08, 2016].

[21] A. Eswaran, A. Rowe, and R. Rajkumar, "Nano-RK: an energy-aware resource-centric RTOS for sensor networks," in *26th IEEE International Real-Time Systems Symposium (RTSS)*, Dec 2005, pp. 10 pp.–265.

[22] A. Rowe, R. Mangharam, and R. Rajkumar, "RT-Link: a time-synchronized link protocol for energy-constrained multi-hop wireless networks," in *3rd IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON)*, Sept 2006, pp. 402–411.

[23] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys)*. ACM, 2003, pp. 138–149.

[24] T. Schmid, P. Dutta, and M. B. Srivastava, "High-resolution, low-power time synchronization an oxymoron no more," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. ACM, 2010, pp. 151–161.

[25] F. Terraneo, L. Rinaldi, M. Maggio, A. V. Papadopoulos, and A. Leva, "FLOPSYNC-2: efficient monotonic clock synchronisation," in *IEEE Real-Time Systems Symposium (RTSS)*, Dec 2014, pp. 11–20.

[26] Y. Altintas, *Manufacturing Automation: Metal Cutting Mechanics, Machine Tool Vibrations, and CNC Design*, 2nd ed. New York, NY: Cambridge University Press, 2012, ch. 6, pp. 250–312.