Synchronization of Distributed Controllers in Cyber-Physical Systems

Vuk Lesi
Dept. of Electrical and Comp. Eng.
Duke University
Durham, North Carolina, USA
vuk.lesi@duke.edu

Zivana Jakovljevic
Faculty of Mechanical Engineering
University of Belgrade
Belgrade, Serbia
zjakovljevic@mas.bg.ac.rs

Miroslav Pajic
Dept. of Electrical and Comp. Eng.
Duke University
Durham, North Carolina, USA
miroslav.pajic@duke.edu

***Abstract*—Due to misaligned clock sources, distributed control in Cyber-Physical Systems (CPS) requires not only synchronous execution of control algorithms on distributed system components, which we refer to as *cyber-synchronization*, but also appropriate generation of actuation signals—we refer to this as *physical-synchronization*. In this paper, we define general requirements for cyber-physical synchronization, as well as show their use on a specific real-world application—distributed motion control for reconfigurable manufacturing systems. We present synchronization challenges in such systems and investigate effects of synchronization errors on the overall system functionality (i.e., machining accuracy). Furthermore, we introduce a low-cost synchronization scheme that can be implemented with of-the-shelf components and validate it on standardized accuracy tests with 2D configurations of industry-grade single-axis robots. We show that our cyber-physical synchronization techniques ensure minimal accuracy impairment of distributed motion control without introducing significant cost/overhead to system design.**

***Index Terms*—cyber-physical synchronization, distributed position control, reconfigurable manufacturing systems**

I. Introduction

Due to the tight interaction between the *cyber* and *physical* components in Cyber-Physical Systems (CPS), synchronization of distributed CPS controllers imposes challenges that span beyond conventional views on synchronous software execution—i.e., synchronicity of actuation signals directly affects the behavior of controlled physical elements. For example, commonly used pulse train-based actuation signals are generated by hardware components clocked by potentially unsynchronized clock sources; this affects their frequency or pulse width and thus may introduce additional misalignment of actuation signals on distributed control nodes. Consequently, to achieve the desired quality of control in distributed CPS, not only does invocation of deployed controllers need to be synchronized (we refer to this as *cyber-synchronization*), but distributed generation of actuation signals must be performed in a way that ensures their *physical-synchronization*.

The problem of synchronous software execution (cyber-synchronization) has been widely addressed, with plethora of existing synchronization protocols. For example, in the wireless domain, the method from [1] employs out-of-band beacon signal to achieve sub-20 μs synchronization in a power-conserving fashion. Control-theoretic approaches (e.g., [2], [3]) are used to perform synchronization solely based on synchronization packet arrival times, rather than payloads containing transmission timestamps, achieving sub-μs synchronization errors. However, this protocol relies on the use of contention delay-free media access control. Dual clock-based, virtual high-resolution time, based on special-purpose hardware, is presented in [4]. Also, very accurate (and expensive) chip-scale atomic clocks can be used to achieve ns-level synchronization of clock sources even over wireless (e.g., [5]). All these methods focus on providing ever-precise software timestamping or clock synchronization capabilities (typically for sensing tasks), while understanding of their effects on actuation and the underlying controlled physical process remains rather limited.

In general, challenges for *cyber-physical synchronization* are platform- and application-specific. For example, the underlying platform determines whether fine-grained tuning of the oscillator circuitry to provide clock-source frequency synchronization (e.g., as in [5]) is available, or if software timestamping-based methods have to be used (e.g., as in [6]); similarly, the actuation inputs are typically controlled by analog voltage (from DA convertors) or timer-generated pulse frequency/width, which on the other hand imposes different requirements for physical-synchronization. Thus, in this work we focus on the synchronization challenges for commonly used low-cost platforms without clock-source frequency tuning—i.e., where timestamping-based synchronization should be performed, and where pulse frequency/width-controlled actuator drives are employed. To show their effects on performance of distributed controllers in CPS, we focus on synchronization problems for a specific industrial application, with emphasis on architecture-based trade-offs and challenges. Specifically, we consider distributed motion control in reconfigurable manufacturing systems; however, the results presented in this work can be easily generalized to a wide range of CPS applications and domains that employ similar actuation stages.

By synchronicity of CPS controllers, we specifically refer not only to synchronous invocation of control code on distributed controllers, but also to (1) compensation of actuation signals that must be implemented due to clock-source

This work is sponsored in part by the ONR under agreements N00014-17-1-2012 and N00014-17-1-2504, and the NSF CNS-1652544 grant, as well as Serbian Ministry of Education, Science, and Technology grant TR35004.

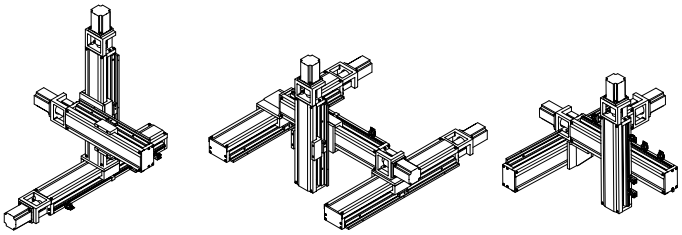


Fig. 1. Reconfigurable manufacturing systems composed of single-axis robots controlled in a distributed manner—tight timing synchronization of distributed controllers is needed to ensure the desired trajectory accuracy.

skews, and (2) timely administering updates of the actuation signals in a manner that minimizes physical disturbances in the controlled process. We start by capturing challenges for cyber-physical synchronization and present their effects on the performance of CPS controllers. We provide a low-cost cyber- and physical-synchronization scheme that minimizes quality-of-control degradation due to control distribution (i.e., moving from centralized to distributed control architecture). Finally, we implement an instance of distributed position control on low-cost ARM Cortex-M4F controllers and validate our implementation on a series of standardized accuracy tests.

This paper is organized as follows. Sec. II introduces the distributed motion control problem and its specific synchronization requirements. Sec. III defines synchronization challenges for CPS controllers, and captures effects of synchronization errors on the overall system performance. Sec. IV introduces a method to address these challenges and minimize cyber- and physical-synchronization errors. Sec. V describes our implementation and experimental validation, before concluding remarks are provided in Sec. VI.

II. Motivating Application: Reconfigurable Manufacturing

We consider the problem of distributed position control of multi-axis machines (see Fig. 1) whose design should facilitate system reconfigurability, based on modular components that are easily integrable and convertible in terms of functionality [7]–[10]. For example, their two-dimensional configurations can be used in conjunction with laser cutters, while three- and higher-dimensional configurations can be used for machining or complex assembly tasks; Fig. 1 illustrates three such configurations obtained with a suitable number of single-axis robots.

These machines traditionally employ centralized control [11], which significantly limits their reconfigurability [6]. Recently, in [6] we have shown feasibility of distributed motion control where each single-axis robot is controlled by its own networked microcontroller (MCU)-based platform, which is referred to as the *low-level controller* (LLC). Specifically, all low-level control and hard real-time tasks critical for operation of a single-axis robot (e.g., limit switch monitoring, position control) are performed by the LLC, while high-level trajectory planning and scheduling of manufacturing tasks is executed on an off-machine (potentially an edge- or cloud-based) *high-level controller* (HLC). Such LLCs have hard real-time requirements, with position control (for both centralized and distributed architectures) implemented as a periodical interpolation (IPO) routine with a period on the order of $1 - 10\ ms$ [11]. As the typical working cycle for every axis consists of an acceleration phase, a constant velocity phase, and a deceleration phase, from the commanded velocity profile, the IPO routine interpolates the next reference position and computes the actuation signal for the next period.

Finally, the actuation stage commonly consists of a stepper motor or variable frequency drive (e.g., [12]) with a corresponding motor; the actuation signal is typically a pulse train whose frequency or pulse width is controlled by the MCU. For example, in a stepper-based system, unit displacement of an axis corresponds to an actuation pulse, and thus the specific output pulse frequency depends on the desired motion speed.

Synchronization Requirements for Distributed Motion Control: Any synchronization error between motion controller executions on different axes (i.e., the IPO routines) introduces additional machining accuracy impairment. As we show in Section V, due to the system design, execution times of the IPO routines do not vary significantly. Thus, instead of synchronous control updates (i.e., actuation), it is sufficient to minimize the synchronization error between the IPO invocations on all axes in the system. For example, Fig. 2(a) illustrates machining accuracy impairment due to a constant synchronization error between IPO routine invocations on X and Y axes. We refer to such requirements for synchronous IPO invocations as ***cyber-synchronization***.

However, distributed position control not only requires synchronization of the IPO routines on all axes. Specifically, actuation signals (their frequency or pulse width) are directly affected by the unsynchronized clocks that drive dedicated hardware timers used to generate these signals. For instance, with pulse frequency-controlled actuation, the pulse train frequency is directly proportional to the desired motion speed; thus, any variation in clock frequency on LLCs implies inconsistencies between commanded and obtained velocities, which in return directly affects trajectory accuracy. This is illustrated in Fig. 2(b), where a fixed frequency offset in pulse train generation on X and Y axis controllers introduces trajectory error. Additionally, regardless of clock-source synchronicity, in the scenario of the change in motion speed, conducting an update of the pulse frequency-controlled actuation signal at an arbitrary point in time may cause undefined behavior of the motor drive circuitry (e.g., due to a shorter-than-expected high or low signal level), as illustrated in Fig. 4. Thus, in addition to ***cyber-synchronization***, it is also necessary to consider ***physical-synchronization*** between distributed controllers (i.e., their actuation stages) to limit temporal differences between the actuation signals used for control.

Therefore, we focus on design challenges related to timing synchronization for distributed CPS control with emphasis on:

- Control scenarios where ensuring synchronicity of controller invocations is insufficient (i.e., pulse width/frequency-controlled actuation stages),
- Effects of utilizing realistic actuation modules (i.e., motor drives), and

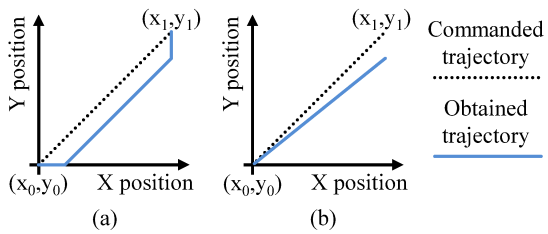


Fig. 2. Illustration of synchronization error-induced trajectory tracking inaccuracies in distributed motion control in a 2-D space: (a) scenario with a fixed synchronization error between interpolation activations on X and Y position controllers; (b) scenario with synchronization error between pulse width or pulse frequency-based actuation signals on X and Y position controllers.

- Application of most common timestamping-based synchronization schemes (e.g., IEEE 1588).

While this work is mainly focused on distributed motion control, the results can be generalized to a range of CPS applications in which similar types of actuation signals are used.

III. Synchronization Challenges for Distributed Control of CPS

Existing synchronization protocols commonly employ the underlying network to communicate information on clock skew among nodes. In general, the CPS controller platform (in conjunction with the sync. protocol) may offer fine-grained adjustments to the on-board digital clock-source (which we will refer to as the *clock-source sync.* architecture), or alternatively, only the local notion of synchronized wall-time (referred to as the *timestamping-based sync.* architecture). In the following we consider challenges for cyber- and physical-synchronization for both architectures.

A. Cyber-Physical Synchronization Challenges with Timestamping-Based Synchronization

As illustrated in Fig. 3(a, b), as part of the synchronization (SYNC) mechanism one free-running hardware timer (referred to as the `SYNC timer` within the SYNC mechanism) is commonly dedicated to be synchronized to the corresponding timers on other nodes (e.g., as in [6]). To achieve this, the timer's counter register is periodically adjusted based on the offset calculated from the received synchronization protocol messages that contain timestamping information. Note that the `SYNC timer` may be a special-purpose timer, different than general-purpose on-board timers; e.g., with IEEE 1588 Precision Time Protocol, a dedicated on-board timer is required, with direct hardware connection to the Ethernet peripheral.

Since the local `SYNC timer` is the only peripheral whose operation is synchronized with other nodes, it is necessary to use software to propagate synchronization to the remaining peripherals. This can be achieved through a *SYNC routine* that is periodically invoked by the `SYNC timer` to adjust other on-board timers that require synchronization; this results in synchronized invocations of control routines on different nodes by adjusting the timer (`IPO timer` in Fig. 3) used to generate control (i.e., IPO) activations. Such cyber-synchronization requirements are sufficient when the motor drive is controlled by analog signal generated by a D/A converter (Fig. 3(a)), as the desired (analog) actuation signal can be directly generated. On

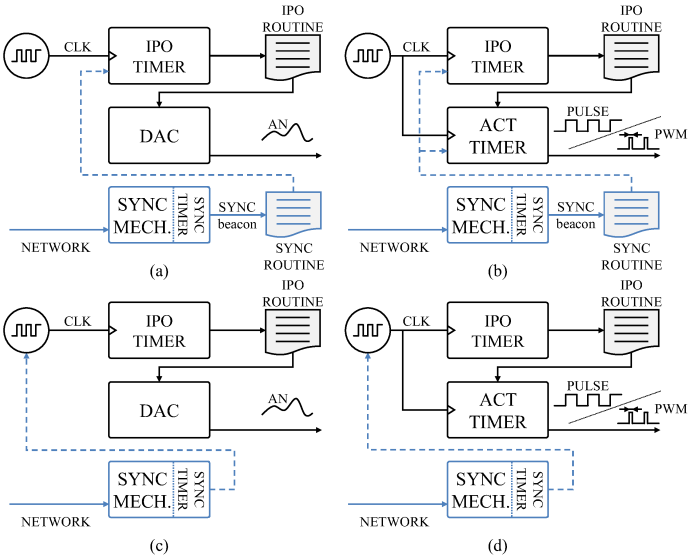


Fig. 3. Synchronization-aware system architecture for distributed motion control: (a, c) when D/A converter is used to generate actuation signals, and (b, d) with timer-based generation of actuation signals; (a, b) when timestamping-based synchronization is available, and (c, d) when clock-source synchronization is available.

the other hand, the common use of pulse-frequency or pulse-width controlled motor drives (Fig. 3(b)) imposes physical-synchronization constraints on the hardware components (i.e., timers) used to generate these signals.

1) Challenges for Cyber-Synchronization: The time base of the SYNC routine typically differs by orders of magnitude from the periods of the IPO routine and actuation signals. For example, with IEEE 1588-based synchronization, a software interrupt with 1 sec period is provided—i.e., 1 pulse per second (*pps*) signal, which is referred to as the *SYNC beacon*. The beacon is used to periodically inspect the state of other on-board timers, and establish the difference between the elapsed local-time for those timers and the global time interval since the last SYNC beacon (i.e., during the latest SYNC interval). For instance, consider the case where the SYNC beacon arrives every 1 sec and the IPO routine should activate every 1 ms. Due to synchronization errors, the number of IPO activations per SYNC interval may not be exactly 1000.

The naive approach to propagate synchronization to the timer activating the IPO routine would be to make the IPO period longer or shorter during the final (e.g., 1000^{th}) IPO period based on the computed offset; this would ensure that the required number of IPO routine invocations occurred during the SYNC interval. However, this could cause disturbance in the physical components of the system, as during the final of the 1000 IPO periods, the axis would potentially have to travel the same displacement significantly faster, due to the abrupt correction of the IPO period duration. This problem is exacerbated when due to errors in the estimation of the timing mismatch, the SYNC routine overcompensates the synchronization error that needs to be accounted for.

2) Challenges for Physical-Synchronization: Physical-synchronization of timers used to generate actuation signals (`ACT timer` in Fig. 3(b)) introduces additional design

challenges. Specifically, the number of pulses (i.e., reload frequency) generated by such a timer directly determines the actuation input delivered to the physical plant. Thus, as the local IPO time intervals are adapted as part of the cyber-synchronization, it is necessary to ensure that the actuation signals generated by actuation timers are properly adapted not only as the function of desired tool/end effector motion (i.e., trajectory/speed) but also as the function of used cyber-synchronization scheme; effectively, the mismatch between the durations of IPO intervals on different nodes can be mapped into a frequency skew between actuation timers on the nodes.

In addition, some modern platforms (e.g., NXP Kinetis K64 MCU family) may not maintain the phase of signals generated by different on-board timers, even when the timers use the same clock-source; we will refer to this phenomenon as the *inter-timer phase synchronization error*. For instance, experiments from [13] showed that the phase shifting between such two timer signals at the output of the chip is approximately one full period of the higher frequency signal over the course of $1\ sec$. When used for control, the phase drift between the two timers does not affect the number of generated pulses, but only alignments of IPO routines' activations and the first/last actuation pulse. This can cause problems when the IPO routine updates the axis velocity as illustrated in scenario from Fig. 4(a), since the input logic of motor drives typically requires pulses with a minimum prescribed pulse width—any pulse violating these requirements is considered as zero commanded velocity.[1] Thus, such short pulses will cause inconsistent motion impairing machining accuracy and potentially increasing machine wear and the chance of tool breakage.

B. Cyber-Physical Synchronization Challenges with Synchronized Clock Sources

A synchronization protocol can be setup on top of hardware offering fine-grained tuning of the oscillator circuitry to provide clock-source frequency synchronization as illustrated in Fig. 3(c, d) (e.g., [5]).

1) Challenges for Cyber-Synchronization: Essentially, clock-source synchronization provides synchronous-rate execution of all functionalities on the distributed platforms. Consequently, synchronization is inherently propagated to IPO invocations and thus, cyber-synchronization is guaranteed (down to the maximum synchronization error).

2) Challenges for Physical-Synchronization: In the case of synchronized clock sources, the actuation pulse train frequencies across controllers are also synchronized; therefore, providing synchronized clock sources ensures desired temporal properties of actuation signals. However, recall *inter-timer phase synchronization* errors illustrated in Fig. 4 (introduced in Sec. III-A2); due to phase drifting between IPO invocations and the actuation signal, unexpectedly short pulses may occur, as in the case of timestamping-based synchronization.

While synchronizing clock sources does alleviate the problem of cyber-physical synchronization, clock-source frequency synchronization usually requires Oven-Controlled Crystal Oscillators (OCXOs), or atomic clock sources which can significantly contribute to the price of an axis controllers for distributed position control. Therefore, in our implementation and evaluation, we consider timestamping-based synchronization.

[1]Undefined motor drive behavior is also possible in the case of non-conforming input pulse-trains.

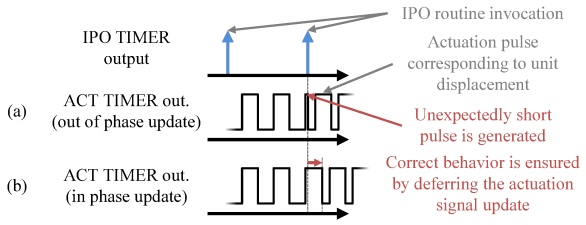


Fig. 4. Example of a frequency-controlled actuation signal update in presence of inter-timer phase synchronization errors: (a) an unexpectedly short pulse may occur, potentially causing a vibration in the axis position if frequency updates are administered immediately as they are computed, (b) correct behavior is ensured by deferring the frequency update.

C. Effects of synchronization errors

To analyze effects of synchronization errors on accuracy in distributed control, we simulated the architectures from Fig. 3(b, d). Specifically, in addition to the hardware timers used for invocation of the IPO routine and generation of actuation signals, we modeled the input stage of a motor drive (widely used Geckodrive G203V that is also used in our implementation described in Section V). The motor drive requires a pulse to have a minimum high level duration of $1\ \mu s$ and a minimum low level duration of $2\ \mu s$ to be considered valid, resulting in a unit displacement (i.e., Basic Length Unit—BLU) of the axis typically in the $[1, 20]\mu m$ range, depending on the employed mechanical system.

We compared the obtained pulse trains that propagate through the amplification and output stages of the motor drives for two axes. We considered three scenarios:

(a) Fixed invocation delay between controllers on two axes (i.e., as illustrated in Fig. 2(a)),
(b) Fixed skew of the pulse frequency between the controllers (i.e., as illustrated in Fig. 2(b), and
(c) Fixed error in actuation signal update (i.e., as in Fig. 4));

where (a) captures cyber-synchronization errors, while (b), and (c) capture effects of physical-synchronization errors. Fig. 5 shows results for the sample acceleration phase of a standard axis working cycle, where a nominal motion speed is achieved in 10 steps by increasing commanded velocity (i.e., the frequency of pulses) every $10\ ms$. Fig. 5(a) shows position errors induced by synchronization errors in IPO routine invocations—with sufficiently good cyber-synchronization, minor accuracy impairment can be guaranteed under distributed axes control.

Fig. 5(b) shows position errors induced by clock skews between two controllers, i.e., non-equal frequencies of the generated actuation pulse trains. Here, we assume that IPO routines are perfectly synchronized. Thus, due to the critical (i.e., last in the IPO period) pulses generated by a skewed clock being intercepted by an IPO routine updating the signal frequency, this error is not linear in the actuation frequency skew. In essence, some skewed frequencies have closer harmonic

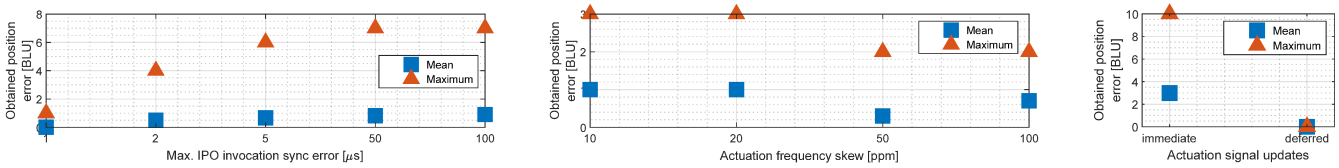


Fig. 5. Axis position errors (in Basic Length Units—BLU) due to (a) cyber-synchronization errors (i.e., IPO routine invocation misalignments), and (b)–(c) physical-synchronization errors affecting the exact number of valid pulses issued by an axis controller over a period of time (i.e, due to frequency clock skews in (b), and improper administering of actuation signal updates in (c)).

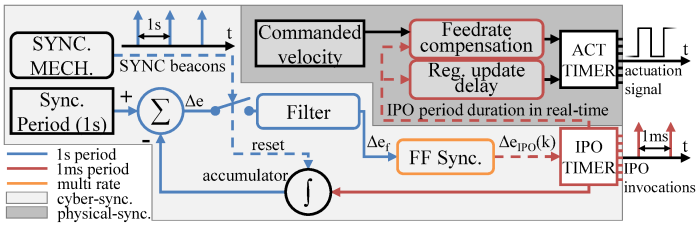


Fig. 6. Synchronization scheme for distributed control of CPS.

relation to the nominal pulse frequency, requiring tightly controlled physical synchronization in the general case (e.g., arbitrary commanded velocities). It is important to note that simulations of our actuation system stage become intractable for skews lower than 10 ppm, as the required simulation step approaches 1 ps.

Fig. 5(c) compares position errors induced by *immediate* and *deferred* updates of the actuation signal at the 10 acceleration points (i.e., during the same acceleration stage). As the critical pulses are intercepted by the (synchronous) IPO update, a number of pulses is *lost*; on the other hand, if the actuation signal updates are deferred, no positioning error is introduced. Notice that in this case we assume IPO invocations are perfectly synchronized, as well as that there is no frequency skew between pulse trains across axes. Therefore, errors induced due to synchronization shown in Fig. 5 pertain to individual effects of non-ideal cyber- and physical-synchronization on the actuation signals, and do not take into account mechanical effects such as loads.

IV. Synchronization of Distributed Controllers

To achieve both cyber- and physical-synchronization of distributed controllers, we introduce the synchronization scheme presented in Fig. 6. We start by noting that cyber-synchronization performance is not affected by the employed physical-synchronization scheme, unlike the other way. In addition, the following synchronization scheme applies to platforms providing timestamping-based synchronization (i.e., architectures from Fig. 3(a, b)).

Our cyber-synchronization scheme employs a *feed-forward* (FF) approach using the estimate of the error between local time duration of the last SYNC beacon and the predefined (i.e., global) SYNC beacon interval—the error is denoted as Δe in Fig. 6. Since the clock skew between nodes does not change very fast, the synchronization error can be expressed as a slowly-changing bias term and a random (measurement) noise; e.g., the computed offset may depend on the (varying) network utilization. To avoid over-reacting to the noise, the error signal is filtered over time and the filtered beacon synchronization error Δe_f is uniformly accounted for over the entire 1000 IPO invocations within the SYNC beacon interval; this is done by periodically introducing very small corrections. For example, if the `IPO timer` is clocked at 60 MHz, then 60,000 timer $ticks$ need to elapse between two $1ms$ IPO routine invocations. Given a typical clock skew of a few ppm, this amounts to offset correction of e.g., $\Delta e_f \approx 3\ \mu s$ every $1\ s$, which is around 180 $ticks$ of the `IPO timer`. Instead of correcting this offset once, which could affect the physical part of the system, we 'spread' corrections of 1 $tick$ uniformly (or nearly uniformly) over 180 out of the 1000 IPO periods in the SYNC beacon interval.

The methodology used for cyber-synchronization is not applicable for physical-synchronization, as the timer generating actuation signals (i.e., `ACT timer` from Fig. 6) operates on a much smaller time base—the actuation signal can change state hundreds of times during one IPO period. In addition, the `ACT timer` adapts parameters of its output signals according to the desired trajectory. Thus, correcting for the offset the same way as previously discussed could have severe consequences to the physical part of the system. For instance, in a pulse-width controlled actuation stage, a correction of just a few $ticks$ may be nearly equal to the pulse width of a unit actuation pulse, potentially causing an abrupt vibration in the mechanical system. We thus, as shown in Fig. 6, propose to synchronize the actuation signal to the `IPO timer`, rather than the originally synchronized free-running timer, part of the employed SYNC mechanism.

Like previously described, as part of the `IPO timer` synchronization, corrections in the `IPO timer` counter are introduced throughout the SYNC period. Thus, duration of some IPO periods will slightly deviate from the nominal duration (1 ms for our running example), and it is important to consider feedrate compensation when configuring `ACT timer`. Since the duration of the next IPO period is always known at every IPO routine activation (as the timer reload value has to be set for the next IPO activation), that duration should be used for velocity calculations instead of the nominal duration (of 1 ms). Let us consider a stepper-based actuation stage where pulse width is kept constant and the pulse frequency determines velocity. For instance, if the commanded velocity is 20 $\frac{\mu m}{ms}$, and the unit displacement (BLU) is 5 μm, the nominal actuation signal is 4 $\frac{pulses}{ms}$. Instead of configuring the `ACT timer` to this pulse frequency, the IPO routine should set the frequency of the actuation signal

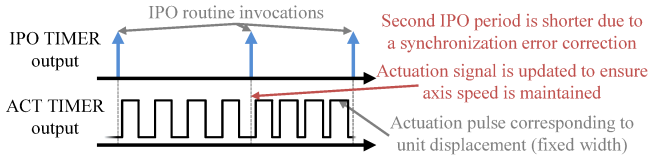


Fig. 7. Example of the synchronization procedure for the `ACT timer` based on the period of the IPO routine. Adjusting the pulse frequency to the synchronized `IPO timer` period ensures suitable velocity is obtained.

to 4 $\frac{pulses}{next\ IPO\ period\ lenght}$, to account for the `IPO timer` synchronization. This is depicted in Fig. 7 with a drastic difference in the IPO periods for illustrative purposes.

Finally, to prevent the scenario from Fig. 4(a), modern timer peripherals in microcontrollers offer the functionality of deferring updates of the `ACT timer` parameters through the use of buffered registers. Specifically, the IPO routine updates the `ACT timer` registers as soon as it computes the corresponding actuation parameters, while the hardware internally updates the timer registers with the buffered values only at predefined loading points. In this case, a suitable loading point to update the signal frequency would be at the end of pulse currently being generated by the timer, and after the minimum low-level duration has elapsed, as shown in Fig. 4(b). Note that in the case platforms with synchronized clock sources, only this compensation is required to maintain physical-synchronization.

V. System Implementation and Validation

To evaluate the presented synchronization techniques, we implemented distributed motion control for HIWIN KK86 single-axis robots driven by TRINAMIC QSH5718-76-28-189 stepper motors and Geckodrive G203V stepper-motor drives. Axis controllers (i.e., LLCs) execute on ARM Cortex-M4F-based, NXP FRDM-K64F development boards running MQX RTOS with full Ethernet communication stack. Also, IEEE 1588-based synchronization is available through an IXXAT library [14]; however, this specific platform supports only timestamping-based synchronization. Furthermore, as the synchronization library encapsulates the configuration of the IEEE 1588 hardware timestamping timer, only synchronized 1 pps (pulse per second) events are exposed, thus requiring additional software synchronization routine—the IPO and actuation timers are synchronized using the methodology described in Section IV. The physical setup is shown in Fig. 9(c)

Fig. 8 shows the LLC software architecture. A Transmission Control Protocol (TCP) client is used for communication with a HLC that constructs trajectories for individual axes from the (to-be-machined) workpiece specification. A sporadic task with 400 ms period checks the state of the local double-trajectory buffer, and requests a buffer refresh if any of the buffers is exhausted. We employ a double-buffer strategy to eliminate starvation and minimize resource access conflicts; i.e., position control consumes one buffer and actuates the physical system with tight timing requirements, while the second buffer can be reloaded by the TCP communication task. The IPO routine is triggered by a periodic 1 ms interrupt from the `IPO timer`. To minimize control latency and jitter,

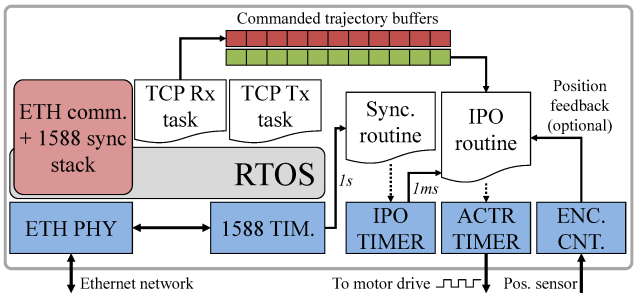


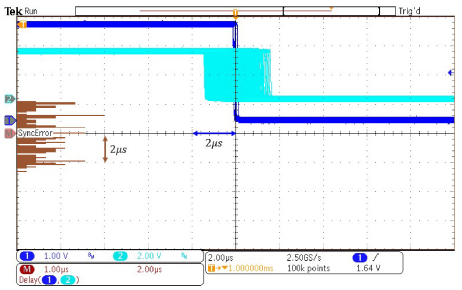
Fig. 8. Hardware/software architecture of the low-level controllers (LLCs).

this interrupt routine is configured as a *kernel interrupt*[2] and assigned highest-priority, resulting in non-preemptive IPO executions. Yet, since the IPO routine only computes commanded speed using the reference velocity profile, and thus has a very short execution time, its non-preemptivity does not significantly impact other (e.g., TCP communication or IEEE 1588 synchronization) tasks.

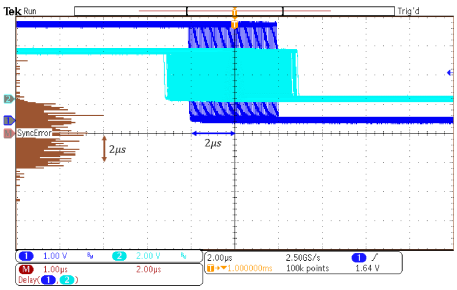
SYNC routine is invoked every 1 sec, based on the IEEE 1588 SYNC beacon, which in our experiments were synchronized down to ± 200 ns on distributed axes. The SYNC routine implements the offset noise term correction within the `IPO timer` counter register and updates the error filter for the feed-forward synchronization, as described in Section IV. The FF synchronization is performed regularly by the IPO routine itself. Fig. 9 shows the obtained synchronization errors between IPO routine invocations on two axis controllers for regular axes operation (i.e., with all tasks executed) as well as under reduced MCU utilization; histograms of synchronization errors are shown vertically on the left side of the oscilloscope screens, where both horizontal (for the time domain signals) and vertical division (for the histogram) are 2 $\frac{\mu s}{div}$.

As shown in Fig. 9, using the presented synchronization techniques we propagated IEEE 1588-based time-stamping synchronization (with error $\sim$ ± 200 ns), to synchronized execution of the main application functionality (distributed position control) with the synchronization error bounded by ± 3.4 μs when the entire application stack was executed on the MCU-based axis controllers. Synchronization error is ± 1.6 μs, when position control and communication are inhibited and axis controllers only maintain synchronization (i.e., TCP server is offline and there is no reference trajectory to execute)—therefore, the software SYNC routine reduces sync performance by only $\sim \pm 1.4$ μs. Note that these results only ensure that the synchronization errors for IPO invocations are within the desired limits (i.e., cyber-synchronization). In the following subsection we experimentally evaluate the impact of the deployed synchronization techniques on the main application performance metrics—position control (i.e., machining) accuracy, using a standardized accuracy test.

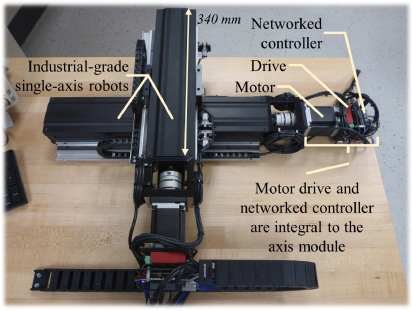
[2]Kernel interrupts in MQX are allowed to bypass the standard bookkeeping of an *interrupt manager* in an RTOS; this ensures minimal interrupt service latency, but precludes system calls (e.g., mutex locking/unlocking). This is not a limitation as the IPO routine only performs geometrical interpolation calculations and configures hardware used to generate actuation signals.



(a) IPO routine synchronization error under reduced MCU utilization



(b) IPO routine synchronization error under normal MCU utilization



(c) Distributed single-axis robot setup used for evaluation in Sec. V-A

Fig. 9. Synchronization errors between IPO routines (with 1 ms period); falling edges of two signals (dark- and light-blue) represent IPO invocation on two LLCs: (a) synchronization error is ± 1.6 μs when the TCP server is offline, and thus the axis control is inhibited as no reference trajectory is available; (b) synchronization error is ± 3.4 μs under normal operation, during execution of a reference trajectory supplied from the TCP server. Brown histograms show the synchronization error distributions. Both the horizontal division (i.e., for the time domain signals) and vertical division (i.e., for the histogram) are 2 $\frac{\mu s}{div}$. (c) shows the physical 2D setup of single-axis robots distributed used for evaluation of proposed cyber-physical synchronization.

A. Validation on Machining Accuracy Tests

Our validation is based on tests defined by the ISO 10791-7 standard [15]; we compare system's performance in terms of geometric accuracy of pieces machined by the centralized and distributed control architectures. Influence of different systematic and random errors inherent from the underlying mechanical system can be isolated if both centralized and distributed control modes are tested using the same components under the same conditions. Thus, to evaluate centralized control architecture, we implemented a *synchronous* mode of operation where the IEEE 1588-based synchronization is bypassed and axes are controlled synchronously. Essentially, system's performance in synchronous mode sets a baseline benchmark for our physical process-aware sync. evaluation.

Fig. 10 shows the considered test piece along with tolerances on geometric features. We perform tests using the feedback signal (i.e., encoder measurements) to acquire the machining trajectory, as described in Annex D of ISO 230 Part 4 [16]; this procedure is alternative to using equipment such as a laser interferometer. [3] Since errors are not obtained from an independent measuring instrument, we compute the *straightness* error by finding maximal deviations (in both perpendicular directions) from a straight line fitted to the first and the last sample corresponding to the respective feature. *Angularity* is determined by computing the distance between two lines sloped according to the feature angle that enclose the obtained trajectory. *Position* error is determined by computing the distance between the commanded and the obtained bore positions.

Table I compares an excerpt of obtained straightness, angularity and position errors, as defined in ISO 10791-7, for the synchronous and distributed modes of operation. The first column shows worst-run machining errors in the fully synchronous mode of operation. The remaining columns summarize results for distributed motion control with cyber-synchronization only (middle), and with both cyber- and

[3]BALLUFF S1F series magnetically-coded sensing system is used to acquire axis position.

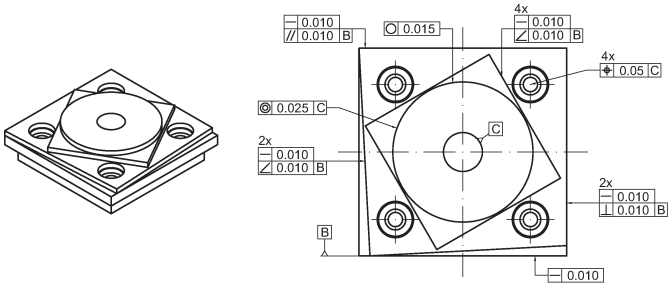


Fig. 10. ISO 10791-7 test piece; machining trajectory of the piece in the X-Y plane is used as a reference input during testing. Tolerances are captured as $(type, tolerance, datum)$, where $type$ specifies how the tolerance is measured, $tolerance$ is the allowed deviation, and the optional $datum$ specifies the reference features relative to which the respective tolerance is defined.

physical-synchronization implemented (right); specifically, we show the worst-run additional accuracy impairments relative to the synchronous mode. As can be seen, without physical-synchronization, straightness of linear segments remains practically unaffected as the clock skew between axes is roughly constant. Yet, tilted segments and positioning are heavily affected due to the accumulation effect of the different number of actuation pulses among axes. On the other hand, when physical-synchronization is employed, practically insignificant accuracy impairments are introduced when position control is distributed (recall that the BLU of our axes is 1 $BLU = 5$ μm). This demonstrates the effectiveness of our cyber-physical synchronization techniques in realistic scenarios.

B. Discussion

Notice that implementing closed-loop position control (i.e., by utilizing position measurements to adjust the actuation signal at real-time) inherently compensates for some physical-synchronization issues. Specifically, feed-rate compensation (see Fig. 7) is naturally performed by the closed-loop controller, and does not need to be handled separately. Additionally, actuation pulse-train frequency skews are also inherently masked by the closed-loop controller that "*skews*" the frequencies to compensate towards minimizing error to target position. However, timely administering of actuation

TABLE I

EXCERPT OF MEASURED ERRORS: SYNCHRONOUS MODE ERRORS ARE SPECIFIED FOR THE WORST RUN, WHILE THE DISTRIBUTED-MODE ERRORS ARE GIVEN AS THE WORST-RUN ADDITIONAL IMPAIRMENTS WITH CYBER-, BUT WITH AND WITHOUT PHYSICAL-SYNCHRONIZATION.

	Feature and tolerance	Synchronous mode	Distr. w/cyber w/o phys.	Distr. w/cyber w/ phys.
outer edges	B datum straightness [μm]	< 1.03	$+0.00$	$+0.00$
	left vertical edge straightness [μm]	< 1.00	$+0.00$	$+0.00$
diamond edges	bottom-left edge straightness [μm]	< 22.81	$+0.00$	$+0.00$
	bottom-left edge angularity [μm]	< 23.33	$+40.31$ max	$+0.71$ max
corner bores	bottom-left bore position [μm]	< 3.16	$+39.22$ max	$+0.00$
	bottom-right bore position [μm]	< 5.00	$+63.88$ max	$+0.00$

signal updates remains a challenge (see Fig. 4) as undesired pulse width may cause undefined drive behavior and/or present additional unwanted disturbance for the closed-loop algorithm.

On the other hand, while closing the loop (i.e., using feedback in controls) eliminates the need for some of our methods, it significantly increases the cost of a single-axis module. For instance, in our implementation the cost of the magnetically-coded sensing system with resolution $1\ \mu m$ and overall measurement system accuracy of $\pm 10\ \mu m$ (equal to $\pm 2\ BLU$) amounts to approximately 50% of the price of the single-axis robot itself. Thus, cost reduction is a natural choice for application where dynamic cutting forces do not exist and adaptability to a wide range of dynamic loads is not necessary, such as in contactless workpiece machining, assembly tasks, etc.

Fig. 11 qualitatively summarizes architectural design tradeoffs based on the employed synchronization technique, for different actuation (i.e., pulse/frequency- or analog-based) and position control types (i.e., open or closed loop). In essence, synchronized clock sources help reduce the complexity of synchronization software compared to timestamping-based synchronization schemes, but at the expense of generally costlier hardware platform. On the other hand, while potentially more accurate, closed-loop systems are significantly more expensive due to the cost of feedback sensing components. Conversely, open-loop systems may require more complex software, due to additional synchronization techniques described in this paper.

VI. CONCLUSION

In this paper we have presented a design approach that allows platform- and application-specific use of cyber-synchronization schemes for distributed CPS controllers. Additionally, we have introduced a method for physical-synchronization of distributed actuation signal generation. We have quantified the effects of cyber- and physical-synchronization errors through simulation of the actual actuation stage we later deployed in our implementation. We have evaluated our synchronization scheme by implementing an instance of distributed motion control with low-cost MCUs

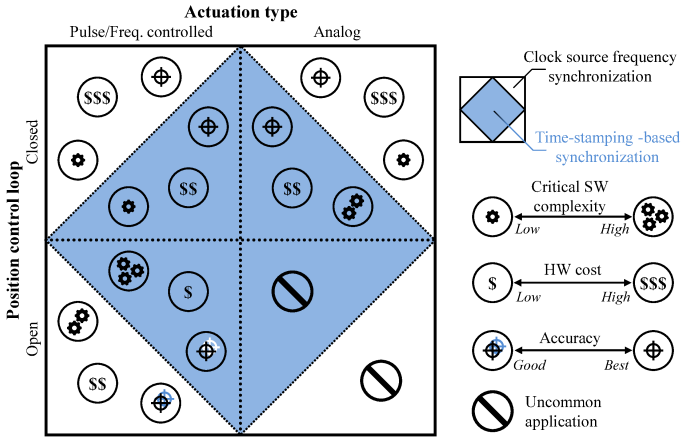


Fig. 11. Architectural design tradeoffs: comparison of software complexity, hardware cost and the overall system accuracy for distributed CPS controllers.

and a 2D configuration of industry-grade single-axis robots. Finally, we have demonstrated that cyber-physical synchronization ensures maintenance of the system's main functionality (i.e., machining accuracy) on standardized accuracy tests. As avenue for future work, we propose to address architectural issues during distribution of conventional numerical control kernels used for centralized multi-axis motion control over networked CPS controllers by quantifying real-time bandwidth and computation requirements.

REFERENCES

[1] A. Rowe, R. Mangharam, and R. Rajkumar, "Rt-link: A time-synchronized link protocol for energy- constrained multi-hop wireless networks," in *2006 3rd Annual IEEE SECON*.

[2] A. Leva and F. Terraneo, "Low power synchronisation in wireless sensor networks via simple feedback controllers: The flopsync scheme," in *2013 American Control Conference*, June 2013, pp. 5017–5022.

[3] F. Terraneo, L. Rinaldi, M. Maggio, A. V. Papadopoulos, and A. Leva, "Flopsync-2: Efficient monotonic clock synchronisation," in *2014 IEEE Real-Time Systems Symposium*, Dec 2014, pp. 11–20.

[4] T. Schmid, P. Dutta, and M. B. Srivastava, "High-resolution, low-power time synchronization an oxymoron no more," in *IPSN 2010*.

[5] A. Dongare, P. Lazik, N. Rajagopal, and A. Rowe, "Pulsar: A wireless propagation-aware clock synchronization platform," in *2017 IEEE RTAS*.

[6] V. Lesi, Z. Jakovljevic, and M. Pajic, "Towards Plug-n-Play Numerical Control for Reconfigurable Manufacturing Systems," in *ETFA 2016*.

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

[8] 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.

[9] 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.

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

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

[12] Parker Motion & Control Technologies, *Motion Control Products, Drives, Motors and Controller Products*, 2013, 192-490123N5.

[13] H. Ma, "Using FTM Global Time Base Feature with KSDK and TWR-K65F120M," *NXP Community*, July 2015. [Online]. Available: https://community.nxp.com/docs/DOC-106216

[14] IXXAT, *IEEE 1588 PTP Protocol Software*, 2018.

[15] International Standardization Org., *ISO 10791 Test conditions for machining centres – Part 7: Accuracy of finished test pieces*, Std., 1998.

[16] International Standardization Organization, *ISO 230 Test code for machine tools*, Std., 2012.