Delay-based Design of Feedforward Tracking Control for Predictable Embedded Platforms

Mojtaba Haghi[1], Feng Wenguang[1], Dip Goswami[1] and Kees Goossens[1]

Abstract— **This paper presents a design technique for feedforward tracking control targeting *predictable* embedded platforms. An embedded control implementation experiences sensor-to-actuator delay which in turn changes the location of the system *zeros*. In this work, we show that such delay changes the number of unstable zeros which influences the tracking performance. We propose a *zero loci* analysis with respect to the delay and identify delay regions which potentially improve tracking performance. We utilize the analysis results to improve tracking performance of implementations targeting modern predictable embedded architectures where the delay can be precisely regulated. We validate our results by simulation and hardware-in-the-loop (HIL) implementation considering a real-life motion system.**

I. Introduction

High precision motion control systems play an important role in many domains like robotics [1], lithography [2], and many more [3]. They perform fast and accurate tracking of a predefined *reference* signal [1], e.g., motion path. Feedforward tracking controllers are commonly used to achieve this high precision [4]. In such control structures, a feedback component ensures the stability of the closed loop system, and the feedforward component enables the reference tracking [5].

The current trend in many domains is to use embedded platforms to realize a cost and energy effective control implementation [6]. Such platforms usually run dedicated applications and are a part of a larger mechanical or electrical system [7]. They usually have limited computation resources and deal with critical timing constraints [8]. It is important to analyze these constraints and their implication on applications. In the context of control applications, the most important artifacts are sensor-to-actuator delay and sampling period [9]. In the commercial-of-the-shelf (COTS) platforms, an embedded implementation often experiences timing-varying nature of execution which implies time-varying delay and sampling period [10].

For the control perspective, a possible solution to deal with the above time-variation is designing a robust controller. These approaches are either H_∞ based [11] or LMI (linear matrix inequality) based [12] in which a controller is designed to stabilize the system for a given range of sampling periods and delays. Such robustness comes at the cost of degraded performance which is often not acceptable in high precision motion applications. In this paper, we propose an alternative approach where we consider a predictable implementation platform with limited/no time-variation in execution time. Such platforms offer extensive timer mechanism to extract the precise knowledge of delay and sampling period at the design time which can be exploited in the controller design for an improved performance [13].

One of the timing constraints of embedded platforms is sensor-to-actuator delay. It is a common conclusion in the literature that a higher amount of delay negatively influences the control performance [14],[15]. Consequently, the practice is to reduce the delay for a better performance at a higher hardware cost (e.g., higher priorities in the scheduling or a higher processing resource usage [16]). On the contrary, we observe that a higher delay may potentially improve the control performance. We show that the sensor-to-actuator delay influences the systems dynamics and changes the location of system *zeros*. In particular, we show that the sensor-to-actuator delay may change the number of unstable zeros (zeros which are outside the unit circle in Z-plane) which in turn influences the feedforward tracking performance. The performance of model-inversion based feedforward controllers is highly dependent on the number of unstable zeros of the system [5]. There are different approaches to design feedforward controllers that try to deal with unstable zeros in different teachniques, e.g., NPZ-Ignore [17], ZPETC [18] and ZMETC [19]. Regardless of the approach, a change in the number of unstable zeros impacts the feedforward performance significantly. A higher number of unstable zeros degrades the performance. Therefore, if an increase in the delay decreases the number of unstable zeros, it improves the performance of feedforward control. In view of this observation, we treat sensor-to-actuator delay as a design parameter in the controller design to find an optimal delay value with the lowest possible number of unstable zeros.

Our contributions: Our main contributions are the following:

- We characterize the impact of sensor-to-actuator delay on the number of unstable zeros (and hence, the system performance). We show that the sensor-to-actuator delay can be used as a design parameter in the feedforward tracking control.
- Based on the above characterization, we propose a design method for feedforward tracking controllers targeting predictable embedded platforms for an improved performance.
- We validate our design method considering a predictable embedded platform with an hardware-in-the-loop (HIL) implementation.

[1]Mojtaba Haghi, Dip Goswami, Kees Goossens and Feng Wenguang are with electronic systems group, Eindhoven University of Technology. {s.m.haghi, d.goswami, k.g.w.goossens}@tue.nl, fwgsbc@live.com

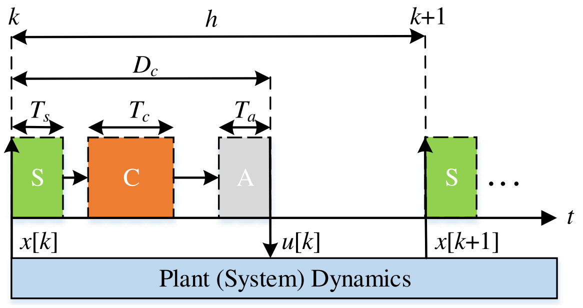

Fig. 1. Timing diagram of an embedded control system

II. Embedded control with delay

This section illustrates impact of the sensor-to-actuator delay on control systems implemented on a embedded platform. We use a motivating case study for illustration.

A. Sensor-to-actuator delay

Let us consider a controllable linear time-invariant (LTI) continuous system. The state-space of the system is given by,

$$\begin{aligned} \dot{X}(t) &= AX(t) + BU(t), \\ Y(t) &= CX(t), \end{aligned} \tag{1}$$

where $X(t) \in R^n$ are the states of the system, $U(t)$ is the input, $Y(t)$ is the output of the system. A control loop is implemented by sequentially and periodically performing three main operations – sensing, computation and actuation. In the sensing operation, the states of the system are read by the sensor at the time instances t_k which are defined as:

$$x[k] := X(t_k), \quad k \in N_{\geq 1} \tag{2}$$

In the computation operation, the controller calculates the next control value $u[k]$. In the actuation operation, the actuator updates the control value $u[k]$ of the system. The time between two sensing operations is called sampling period h.

As illustrated in Fig. 1, execution of the three operations requires finite time on an embedded platform. We denote execution times of the sensing, computation and actuation operations by T_s, T_c and T_a respectively. T_o is the summation of communication overheads of *sensor-to-controller* and *controller-to-actuator*. The time from the start of the sensing operation to the end of the actuation operation is called sensor-to-actuator delay D_c which is given by,

$$D_c = T_s + T_c + T_a + T_o. \tag{3}$$

We focus on the case where delay D_c is shorter than sampling period h,

$$0 \leq D_c < h. \tag{4}$$

B. Delay modeling

Considering h and D_c, the discrete-time equivalent of system (1) is [14],

$$\begin{aligned} x[k+1] &= \phi x[k] + \Gamma_0 u[k] + \Gamma_1 u[k-1], \\ y[k] &= Cx[k], \end{aligned} \tag{5}$$

where $\phi = e^{Ah}$, and

$$\Gamma_0 = \int_0^{h-D_c} e^{As} B ds,$$

$$\Gamma_1 = \int_{h-D_c}^{h} e^{As} B ds.$$

To rewrite the equations in the standard state-space representation, we augment the state with the delayed actuation by defining $\xi[k] = \left[x^T[k], u^T[k-1]\right]^T$. With this new augmented states we have [20],

$$\begin{aligned} \xi[k+1] &= \phi_{aug}\xi[k] + \Gamma_{aug}u[k], \\ y[k] &= C_{aug}\xi[k], \end{aligned} \tag{6}$$

where

$$\phi_{aug} = \begin{bmatrix} e^{Ah} & \Gamma_1 \\ 0 & 0 \end{bmatrix}, \quad \Gamma_{aug} = \begin{bmatrix} \Gamma_0 \\ I \end{bmatrix}, \quad C_{aug} = [C \quad 0]$$

From the definition of Γ_0 and Γ_1 it is easy to note that both ϕ_{aug} and Γ_{aug} are dependent on D_c.

C. Zeros of delayed system

We find the system zero polynomial by transforming the state space model to transfer function representation [21]. The transfer function of the discrete-time augmented state-space (6) is given by,

$$G(z) = \frac{\Delta(z)}{P(z)} = C_{aug}(zI_n - \phi_{aug})^{-1}\Gamma_{aug}, \tag{7}$$

where $G(z)$ is the transfer function, $\Delta(z)$ is the zero polynomial and $P(z)$ is the characteristic (or pole) polynomial. From the definition of matrix inverse we have:

$$(zI_n - \phi_{aug})^{-1} = \frac{adj(zI_n - \phi_{aug})}{det(zI_n - \phi_{aug})},$$

where $adj(.)$ is the adjugate of the matrix and $det(.)$ is its determinant. $det(.)$ is the characteristic polynomial $P(z)$. Therefore from (7), the zero polynomial is

$$\Delta(z) = C_{aug}adj(zI_n - \phi_{aug})\Gamma_{aug}. \tag{8}$$

System discrete-time zeros are the roots of $\Delta(z) = 0$. Since ϕ_{aug} and Γ_{aug} are functions of D_c, we compute discrete-time system zeros by solving $\Delta(z) = 0$ for various delay values.

D. Motivating case study: PATO system

In our work, we used a physical system called PATO as a motivating case study [22]. This system has a fourth-order state-space. The states of the system are the position of the two masses θ_1 and θ_2 and their respective rotary speeds of ω_1 and ω_2. The controller input $u(t)$ is the current i_m of the motor speed controller. The output of the system $y(t)$ is the position of the first mass θ_1. Here, the control task is to make θ_1 follow a desired reference $r(t)$. The identified state-space of the system is adopted from experiments reported in [23] and is as follows:

$$\dot{X}(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -7.08\times10^4 & 7.08\times10^4 & -1.1\times10^6 & 1.1\times10^6 \\ 7.08\times10^4 & -7.08\times10^4 & 1.1\times10^6 & -1.1\times10^6 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ 0 \\ 1.173\times10^4 \\ 1 \end{bmatrix} i_m$$
$$y(t) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} X(t) \tag{9}$$

E. The Impact of the delay on system zeros

We illustrate the impact of delay on the system zeros with the example of system (9). We consider sampling period $h = 8ms$ and varied delay in the range $0 \le D_c < h$. For each delay choice, we first obtain discrete-time augmented system (6) and next, we obtain the zero polynomial (8) for the given h and D_c.

Fig. 2 shows the number of unstable zeros for each value of D_c. It shows that the system has different numbers of unstable zeros for different delays. In particular, there are four delay regions with different number of unstable zeros.

- For $\mathbf{0 \le D_c < 0.1ms}$, the system has no unstable zeros. For example, for $D_c = 0.05ms$ the zero polynomial (8) of the system is:

$$\Delta(z) = z^4 + 1.7z^3 + 1.71z^2 + 0.93z + 7\times10^{-4},$$

from which we obtain the zeros of the system as:

$$z_1 \approx 0, \quad z_2 = -0.9391,$$
$$z_3 = -0.3843 + 0.9186i, \quad z_4 = -0.3843 - 0.9186i,$$

where all of them are inside the unit circle and hence stable.

- For $\mathbf{0.1ms \le D_c < 1.1ms}$, the number of unstable zeros increases to 2. For example, for $D_c = 1ms$ the zeros of the system are:

$$z_1 = -0.0189, \quad z_2 = -0.9834$$
$$z_3 = -0.4539 + 1i, \quad z_4 = -0.4539 - 1i$$

where z_3 and z_4 are unstable.

- For $\mathbf{1.1ms \le D_c < 3.8ms}$, the number of unstable zeros increases to 3.
- For $\mathbf{3.8ms \le D_c < 8ms}$, the number of unstable zeros goes down to 1. For $D_c = 5ms$ the zeros of the system are:

$$z_1 = -5.6, \quad z_2 = -0.6875$$

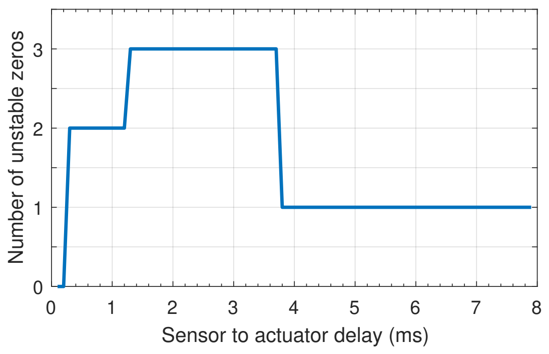

Fig. 2. Number of unstable zeros of PATO system for $h = 8ms$ and $D_c \in [0, h)$

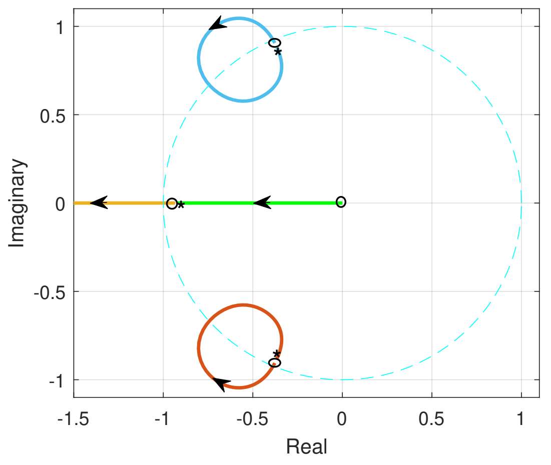

Fig. 3. Zero locus of system (9) for $h = 8ms$ and $D_c \in [0, h)$

$$z_3 = -0.4573 + 0.6i, \quad z_4 = -0.4573 - 0.6i$$

where z_1 is the only unstable zero of the system.

Therefore, the number of unstable zeros of system (9) may change with the value of sensor-to-actuator delay. This further motivates us to investigate how the location of system zeros changes with sensor-to-actuator delay D_c.

III. Zero Loci analysis

In this section, we obtain *zero loci* by computing system zeros from zero polynomial (as described in Subsection II-E) and their transition for $0 \le D_c < h$ for a given h. Next, we provide a number of observations about the zero loci as well as its behavior for first order and second order systems.

Fig. 3 and Fig. 4 show the zero loci for the system (9) with $h = 8ms$ and $h = 10ms$ respectively. A zero starts from the *circle* point for $D_c = 0$ and moves toward the *star* point for $D_c = h - \epsilon$ where ϵ is an infinitesimal quantity. The arrows shows the direction of transition of a zero location. The dotted line represents the unit circle.

- For $h = 8ms$, the analysis is the same as what is described in Section II-E. System starts with 4 stable zeros for $D_c = 0$ but two of them (red and blue lines) immediately go out of the unit circle and become unstable for a small amount of delay $(0.1ms \le$

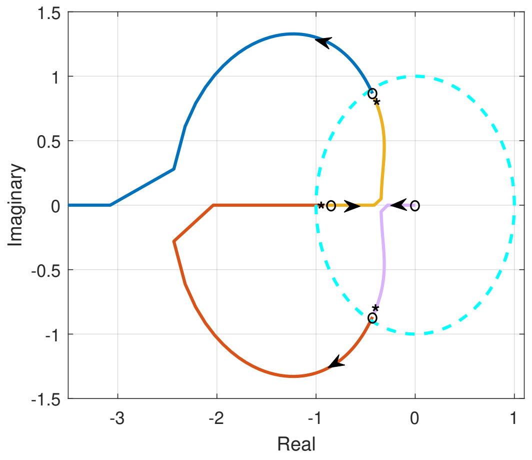

Fig. 4. Zero locus of system (9) for $h = 10ms$ and $D_c \in [0, h)$

$D_c < 1.1ms$). By increasing the delay another zero (yellow line) also becomes unstable. For a higher delay ($3.8ms \leq D_c < 8ms$), two of unstable zeros (red and blue lines) come back to unit circle and become stable.

- For $h = 10ms$, the system starts with 4 stable zeros for $D_c = 0$ but two of them (red and blue lines) go out of the unit circle and become unstable for a small delay ($0 \leq D_c < 0.3ms$). The number of unstable zeros does not change for $0.3ms \leq D_c < 9ms$. For delays close to the sampling period ($9.0ms \leq D_c < 10ms$), one of the unstable zeros (red line) comes back to the unit circle and becomes stable.

It can be noted, in the above examples, there is a zero at the origin for $D_c = 0$ which grows with the increase in delay. The existence of this zero can be proven by the following lemma.

Lemma 3.1. *For any controllable LTI system* (1)*, the delayed model of* (6) *has a zero at the origin of z-plane for* D_c *=0.*

Proof. Since $D_c = 0$, from (5) we observe that Γ_1 is equal to zero. This makes ϕ_{aug} to be a block diagonal matrix:

$$\phi_{aug} = \begin{bmatrix} e^{Ah} & 0 \\ 0 & 0 \end{bmatrix}$$

Since ϕ_{aug} is block diagonal, its eigenvalues are the list of eigenvalues of each block which are e^{Ah} and 0. Since the eigenvalues of ϕ_{aug} are the poles of discrete-time transfer function (7), the system has a pole at the origin and poles at eigenvalues of e^{Ah}. For $D_c = 0$ the transfer function should be identical to the dynamic system without delay and therefore the pole at the origin should be canceled. This implies that there should be a zero at the origin to cancel the existing pole there. Therefore, there is always a zero at the origin for $D_c = 0$. □

In summary:

- Zero loci show that the location of system unstable zeros may change with sensor-to-actuator delay.
- The behavior of zero loci with delay is dependent on sampling period.
- At $D_c = 0$, the system has a zero at the origin of z-plane and it grows with the delay. Whether this zero passes the unit circle and becomes unstable depends on system dynamics.

We further characterize the zero loci for some specific classes of dynamic systems – first order and second order systems.

A. First order systems

We consider a *strictly proper* first order system. The system has one state and the state vector X in (1) is a scalar. The continuous state-space of the system is:

$$\begin{aligned} \dot{x}(t) &= ax(t) + bu(t), \\ y(t) &= cx(t), \end{aligned} \tag{10}$$

where a, b and c are scalars. The augmented state-space based on (6) is given by:

$$\begin{aligned} \xi[k+1] &= \begin{bmatrix} e^{ah} & \Gamma_1 \\ 0 & 0 \end{bmatrix} \xi[k] + \begin{bmatrix} \Gamma_0 \\ I \end{bmatrix} u[k], \\ y[k] &= \begin{bmatrix} c & 0 \end{bmatrix} \xi[k]. \end{aligned} \tag{11}$$

Using (8), the zero polynomial for the system (11) is given by,

$$\Delta(z) = c\Gamma_0 z + c\Gamma_1. \tag{12}$$

Solving $\Delta(z) = 0$, we obtain the system zero at $z = -\frac{\Gamma_1}{\Gamma_0}$. Based on the definition of Γ_0 and Γ_1, the following can be observed:

- $\Gamma_0 = \int_0^{h-D_c} e^{as} b ds = \frac{b}{a}(e^{ah-aD_c} - 1)$ monotonically decreases with the increase in D_c.
- $\Gamma_1 = \int_{h-D_c}^{h} e^{as} b ds = \frac{b}{a}(e^{ah} - e^{ah-aD_c})$ monotonically increases with the increase in D_c.
- $|z| = \left| -\frac{\Gamma_1}{\Gamma_0} \right|$ monotonically increases with the increase in D_c.
- The system zero will start at the origin of z-plane for $D_c = 0$ and will grow with increase of delay. For a value of D_c (near half of sampling period) it goes outside the unit circle and becomes unstable when the sampling period h is *sufficiently* small.

B. Second order systems

The general form of a second order system in continuous-time is given by:

$$G(s) = \frac{b_1 s + b_2}{s^2 + a_1 s + a_2}. \tag{13}$$

where b_1, b_2, a_1 and a_2 are the system parameters. The controllable canonical form equivalent of (13) is:

$$\begin{aligned} \dot{X}(t) &= \begin{bmatrix} 0 & 1 \\ -a_2 & -a_1 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u[t], \\ y(t) &= \begin{bmatrix} b_2 & b_1 \end{bmatrix} X(t). \end{aligned} \tag{14}$$

The augmented discrete-time equivalent of (14) is:

$$
\begin{aligned}
\xi[k+1] &= \begin{bmatrix} e^{Ah} & \Gamma_1 \\ 0 & 0 \end{bmatrix} \xi[k] + \Gamma_0 u[k], \\
y[k] &= \begin{bmatrix} b_2 & b_1 \end{bmatrix} \xi[k].
\end{aligned} \tag{15}
$$

Using the Taylor series expansion, we have:

$$e^{Ah} = I + Ah + A^2h^2. \tag{16}$$

Here, we omitted the higher order elements of the series, since they are negligible with sufficiently small sampling periods. Using (16), Γ_0 and Γ_1 are given by:

$$
\begin{aligned}
\Gamma_0 &= \int_0^{h-D_c} e^{As} B ds = [\frac{1}{2}A(h-D_c)^2 + h - D_c]B, \\
\Gamma_1 &= \int_{h-D_c}^{h} e^{As} B ds = [\frac{1}{2}A(2h-D_c) + 1]D_c B,
\end{aligned} \tag{17}
$$

Using (16) and (17) in (15), we obtain the zero polynomial from (8). Next, we analyze two cases:
Case 1 with $b_1 = 0$: The system has no internal zero. From the zero polynomial in (8) and omitting the negligible parts, we have:

$$\Delta_0(z) = (h-D_c)^2 z^2 + [(h+D_c)^2 - 3{D_c}^2]z + {D_c}^2. \tag{18}$$

We observe the following:

- System has two zeros and their locations depend on D_c and h.
- With $D_c = 0$, we have:
$$\Delta_0(z) = h^2 z^2 + h^2 z,$$
which means the system has one zero on the unit circle and one zero at the origin.
- The zero on the unit circle moves outside (becomes unstable) with the increase in the delay. The zero at the origin moves from the origin towards the unit circle.
- With $D_c = h - \epsilon$ we have:
$$\Delta_0(z) = \epsilon z^2 + h^2 z + h^2,$$
where ϵ is an infinitesimal quantity. It means the stable zero reaches close to unit circle but does not pass it and get unstable.

Fig. 5 shows the zero locus of an example second order system. The number of unstable zeros is one for $D_c \in [0, h)$.

Case 2 with $b_1 \neq 0$: The system has one internal zero. From the zero polynomial in (8) and omitting the negligible parts, we have:

$$
\begin{aligned}
\Delta_1(z) =& \Delta_0(z) + b_1[(h - D_c - \frac{1}{2}a_1(h-D_c)^2)z^2 + \\
& [(h+Dc)^2 - 3{D_c}^2]z + {D_c}^2,
\end{aligned} \tag{19}
$$

which means that the zero polynomial $\Delta_1(z)$ constitutes of $\Delta_0(z)$ (the zero polynomial in Case 1) and more elements with b_1. We observe the following:

- System has two zeros and their locations depend on delay D_c, sampling period h, and system dynamics.

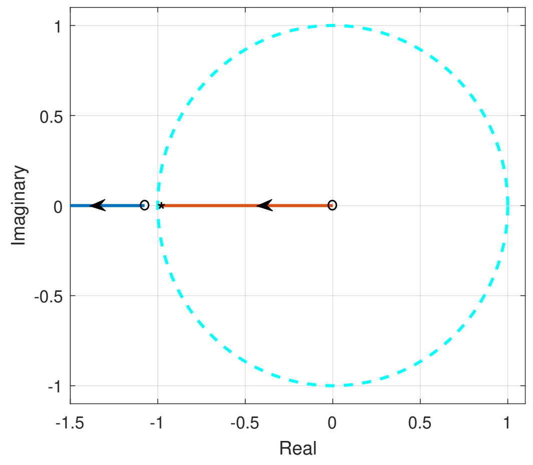

Fig. 5. Zero locus of a second degree system with $a_1 = a_2 = b_2 = 1$, $b_1 = 0$, $h = 10ms$ and $D_c \in [0, h)$.

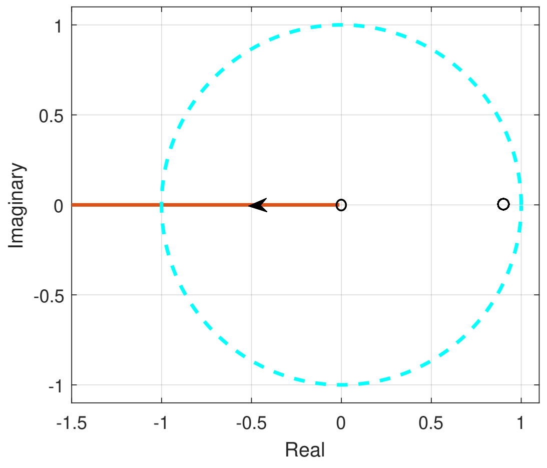

Fig. 6. Zero loci of a second degree system with $a_1 = a_2 = b_2 = 1$, $b_1 = 1$, $h = 10ms$ and $D_c \in [0, h)$.

- With $D_c = 0$, we have:
$$\Delta_1(z) = (h^2 + b_1 h - \frac{b_1 a_1}{2})z^2 + (h^2 - 2b_1h^2)z,$$
which implies that the system has a zero at the origin and one zero at the discrete-time equivalent of internal zero.
- The zero at the origin grows with the increase in delay. For a value of D_c (dependent on system dynamics), it goes outside the unit circle and becomes unstable.
- The other zero which is at the discrete-time equivalent of internal zero is not affected by the delay and stays at the same point for all delay values.

Clearly, the number of unstable zeros changes depending on D_c. The number of unstable zeros will be either 0 or 1. Fig. 6 shows the zero locus of an example second order system.

In this section we showed that the zero placement and the number of unstable zeros can change by varying the sensor-to-actuator delay. This means that if the amount of delay be known and adjustable it could be treated as a control

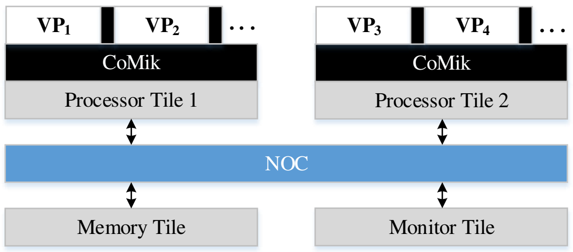

Fig. 7. Predictable embedded platform under consideration

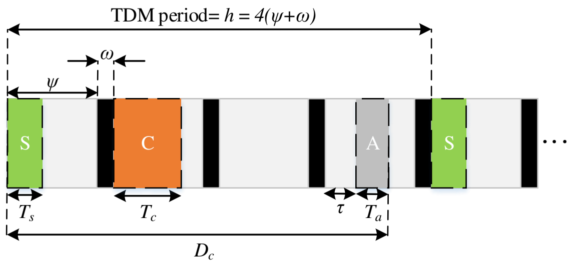

Fig. 8. TDM frame with $N = 4$ and three tasks of S:$sensing$, C:$computation$, A:$actuation$ with execution times of T_s, T_c, and T_a respectively. The black boxes are CoMik slots while white blocks are partition slots. τ is the added delay to execution of actuation in its dedicated slot. Sensing and computation tasks always start execution at the beginning of their allocated slot.

parameter to change the location of the system zeros. Such precise regulation of delay is possible to be implemented on predictable embedded execution platforms explained in the next section.

IV. Embedded Platform Properties

For an embedded control implementation, we consider a tile-based architecture that offers con?guration with multi-processors (processor tiles), interconnections through a Network-on-Chip (NoC), and memories (memory tiles) within the same platform. An example architecture is shown in Fig. 7. Each processor tile is mainly composed of a MicroBlaze soft-core processor. The monitor tile is for debugging purposes. The memory tile contains the external memory interface and controller, and the NoC provides interconnection between the tiles. To enable independent implementation, verification and execution of multiple applications, the platform offers composability by virtualizing all processors, interconnections, and memory resources [8]. First, we illustrate the platform properties and scheduling. We then define sampling period and other timing properties based on the platform scheduling. Finally, we define the sensor-to-actuator as a design parameter.

A. Application scheduling

The platform uses CoMik (Composable and Predictable Micro-kernel) to create virtual processors (VPs) that can be used as dedicated resources [8]. Each VP's utilization of the underlying physical processors and their interconnections (i.e., NoC communication) are allocated in a time-division multiplexing (TDM) manner. Using perfectly periodic TDM policies both in the processors and their interconnections, the platform achieves global synchronization with a very fine (i.e., cycle accurate) time granularity.

To achieve cycle-accurate temporal behavior, we split the TDM frame into N CoMik slots of length ω clock cycles and N partition slots (or VPs) of length ψ clock cycles. The CoMik slots enable jitter-free context switching between VPs, and the applications only execute on the VPs. An application is executed in an allocated partition slot (or VP) and is paused every time a new CoMik slot starts. Its execution is only resumed in the next partition slot assigned for the same application. The TDM frame is defined at design time with desired execution order of the VPs. The TDM period is $N \times (\omega + \psi)$ clock cycles. In Fig. 8, we show a TDM table with 4 partition slots that run sequentially and periodically.

B. Control application scheduling and sampling period

We are interested in scheduling of the control application. As described in Section II, a control application consists sensing, computation and actuation operations. Each operation is implemented as a task. We schedule these tasks in the partition slots in TDM table. To make sure that the execution of each task fits in its respective partition slot, ψ is chosen as follows:

$$\psi \geq Max\{T_s, T_c, T_a\}, \tag{20}$$

where T_s, T_c, and T_a are execution times of sensing, computation and actuation tasks respectively[1]. The calculated T_s, T_c, and T_a on the platform are about 1000, 2000 and 1000 clock cycles respectively. ω is chosen to be as short as possible. In the current implementation for the control application, it is set to 4096 clock cycles.

C. Sampling period

Sampling period is defined as the time period between two consecutive sensing tasks. With sensing task scheduled once in the TDM table, sampling period is given by:

$$h = \frac{N \times (\omega + \psi)}{F_p}, \tag{21}$$

where F_p is the frequency of the platform which is $100Mhz$. Fig. 8 illustrate an example of TDM table in which the sampling period is equal to $4 \times (\omega + \psi)/F_p$. For this example by choosing $\psi = 245904$ clock cycles, the sampling period based on this TDM table is $10ms$.

D. Sensor-to-actuator delay as a parameter

The sensor to actuator delay is computed as (3). The minimum of D_c in our platform is achieved when control tasks scheduled in consecutive slots and their execution start

[1]Although the execution of control tasks are not fixed, they are always very close to their worst case execution time (with an error in order of clock cycles), by the virtue of the platform predictability.

at the beginning of their allocated partition slots. In this case the minimum possible value for overhead would be:

$$T_{o,min} = (\psi - T_s) + (\psi - T_c) + 2\omega. \tag{22}$$

Using (22) in (3), the minimum D_c is given by:

$$D_{c,min} = 2(\psi + \omega) + T_a. \tag{23}$$

Considering a TDM table with 4 slots, $h = 10ms$, ψ, ω, and T_a are about $0.04ms$, $2.46ms$, $0.01ms$ respectively. Therefore, $D_{c,min}$ is $5.01ms$. As described in Section III, the optimal delay (denoted as as D_c^*) is defined as the delay which results in the minimum number of unstable zeros and is implementable on the platform. Since $D_c < D_{c,min}$ is not implementable in the platform, $D_c^* \geq D_{c,min}$. With $D_c^* > D_{c,min}$, D_c can be increased either by adding partition slots between control tasks or by adding a delay in execution of actuation in its partition slot[2]. In this case D_c can be defined as:

$$D_c = D_{c,min} + m(\psi + \omega) + \tau, \tag{24}$$

where m is the number of added slots and τ is the added delay to the execution of actuation. Fig. 8 is an example where $m = 1$ with an added delay τ. Therefore, by choosing the right m and τ, the delay can be adjusted to $D_c = D_c^*$. Thus, this mechanism allows us to use the senor-to-actuator as a design parameter.

V. Delay-based feedforward design

In this section, we present the proposed delay-based embedded feedforward tracking controller. We use the example of system described in Section II-D.

A. Controller architecture

The designed controller is a feedback-feedforward controller. The feedback part stabilizes the system and the feedforward part is a closed-loop model-inversion to achieve accurate reference tracking. The feedback part is a state-feedback with all desired poles set to 0.9 (it can be done using any other state-of-the-art design technique). Calling the feedback part as $C(z)$ and plant transfer function as $G(z)$, The closed-loop transfer function is:

$$G_{CL} = \frac{C(z)G(z)}{1 + C(z)G(z)}.$$

To design the feedforward part F_{CL}, we set F_{CL} equals (or approximately equals) to G_{CL}^{-1}. Ideally, F_{CL} is equal to G_{CL}^{-1} which makes the transfer function between reference and output y equals to 1. If the closed-loop system has unstable zeros direct inversion is not possible and it yields to unstable inversion in F_{CL}. In this case an stable approximation inverse is used. Here, we use *Zero-Phase-Error Tracking controller* (ZPETC) stable approximation [18]. To design ZPETC, we first rewrite G_{CL} as:

$$G_{CL} = \frac{\Delta(z)}{P(z)} = \frac{\Delta_s(z)\Delta_u(z)}{P(z)}, \tag{25}$$

[2]The execution of sensing and computation always starts at the beginning of their respective slots.

where $\Delta_s(z)$ is the stable zero polynomial and $\Delta_s(z)$ is the unstable zero polynomial. Now, the ZPETC controller is:

$$F_{CL} = \tilde{G}_{CL}^{-1}(z) = \frac{P(z)}{\Delta_s(z)\Delta_u^*(z)}, \tag{26}$$

where $\tilde{G}_{CL}^{-1}(z)$ means *approximate* inverse of the closed loop. $\Delta_u^*(z)$ which is a zero-phase approximation of the unstable zeros is defined as:

$$\Delta_u^*(z) = \frac{(\Delta_u(z)|_{z=1})^2 z^n}{\Delta_u^f(z)} = \frac{(\Delta_u(1))^2 z^n}{\Delta_u^f(z)}, \tag{27}$$

where $\Delta_u^f(z)$ is defined based on $\Delta_u(z)$. It means that if:

$$\Delta_u(z) = b_{un} z^n + b_{u(n-1)} z^{n-1} + ... + b_{u0},$$

then

$$\Delta_u^f(z) = b_{u0} z^n + b_{u1} z^{n-1} + ... + b_{un}.$$

$\Delta_u^f(z)$ is at the same degree of $\Delta_u(z)$ with its *flipped* coefficients.

B. Implementation

To design the controller, we first choose the sampling period $h = 10ms$. As described in previous section, the minimum delay $D_{c,min}$ on the platform for this $h = 10ms$ is $5.01ms$. The first step is to choose the optimal delay D_c^* which gives the minimum number of unstable zeros and is implementable on our platform. As described in Section III, for $h = 10ms$, $D_c < 0.3ms$ region does not have any unstable zero. However, $D_c < D_{c,min} = 5.01$ is not implementable on the platform. On the other hand, $D_c = D_{c,min}$ results in two unstable zero while the system have only one unstable zero in the region $9ms \leq D_c < 10ms$. Therefore, we choose the optimal delay which is also achievable in the platform as $D_c^* = 9.5ms$. To achieve this delay, TDM frame is configured as Fig. 8, and we set $m = 1$ and $\tau = 4.59ms$.

C. Results

Validation of the design is done through an HIL (hardware-in-the-loop) implemnentation. Considering the platform in Fig. 7, we implemented the discrete-time model of the system (9) with sampling period of $100\mu s$ on processor tile 1 and the designed controller on processor tile 2. The performance metric is defined as P^{-1} where:

$$P = \sum_{k=1}^{N} (r[k] - y[k])^2$$

The reference signal is:

$$r(t) = sin(4\pi t) + sin(8\pi t)$$

Fig. 9 and Fig. 10 illustrate the results of MATLAB and HIL simulations for $D_c = D_{c,min} = 5.01ms$ and $D_c = 9.6ms$ respectively. Table. I presents the performance results. HIL implementation provides results very close to the simulation which validates the proposed method. Although for the design with $D_c = 9.6ms$ gives a higher transient error, it achieves a better performance in the steady-state compared

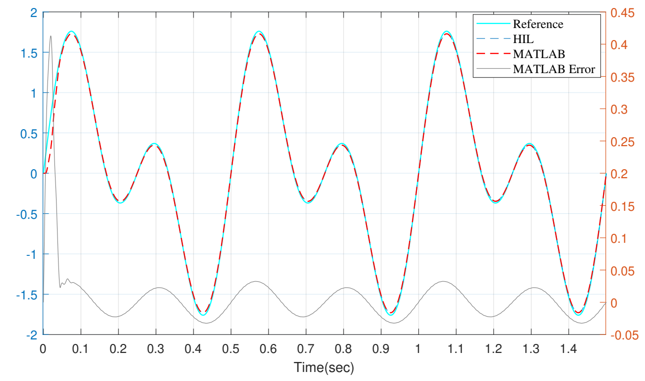

Fig. 9. output response and Error for $D_c = 5.01ms$

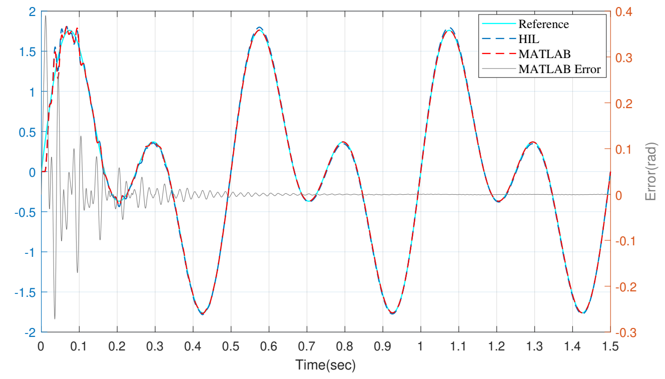

Fig. 10. output response and Error for $D_c = 9.6ms$

TABLE I
PERFORMANCE MEASUREMENT WITH DIFFERENT DELAYS

Delay(ms)	P^{-1}**(MATLAB)**	P^{-1}**(HIL)**
5.01	0.027	0.026
9.6	0.040	0.032

to the design with $D_c = 5.01$. This is because the system has one less unstable zero with $D_c = 9.6ms$.

Discussion: We demonstrated that sensor-to-actuator delay could be treated as a design parameter. In some design scenarios, an extra delay improves the performance of the controller when the number of unstable zeros decreases. More importantly, a higher delay tolerance in the control loop relaxes the requirement on the embedded platform reducing the implementation cost at the end. In embedded implementations, it is common to share platform among multiple applications. Such relaxed timing requirement implies more processing resource for other applications (e.g. the third partition slot in Fig. 8 can be assigned other applications).

VI. Conclusion

In this paper, we proposed a delay-based embedded feedforward tracking controller with improved tracking performance. We demonstrated that a higher sensor-to-actuator delay does not necessarily implies a lower control performance. We have shown that increasing delay may decrease the number of unstable zeros. This in turn improves the performance for controllers using the model-inversion feedforward component. Hence, we used sensor-to-actuator delay as a design parameter. Considering a platform with predictable timing behavior, we validated our approach by HIL simulations. A possible extension of our work can be a general design flow with delay as a design parameter.

Acknowledgments

This work was partially supported by the H2020 project I-MECH (GA no.737453).

References

[1] K. Ohnishi *et al.*, "Motion control for advanced mechatronics," *IEEE/ASME Transactions on Mechatronics*, pp. 56–67, March 1996.
[2] L. Hong *et al.*, "Motion control for wafer stage of 0.1 μm lithography," in *Integration Technology, 2007. ICIT'07. IEEE International Conference on*. IEEE, 2007, pp. 338–342.
[3] Purtojo and Wahyudi, "Integral anti-windup scheme of full-state feedback control for point-to-point (ptp) positioning system," in *2008 International Conference on Electronic Design*, Dec 2008, pp. 1–6.
[4] A. De Luca, "Feedforward/feedback laws for the control of flexible robots," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 1. IEEE, 2000, pp. 233–240.
[5] J. Butterworth *et al.*, "Analysis and comparison of three discrete-time feedforward model-inverse control techniques for nonminimum-phase systems," *Mechatronics*, vol. 22, no. 5, pp. 577–587, 2012.
[6] R. Alur *et al.*, *Handbook of networked and embedded control systems*. Springer Science & Business Media, 2007.
[7] M. Barr, "Embedded systems glossary," *Neutrino Technical Lib.*, 2007.
[8] K. Goossens *et al.*, "Noc-based multiprocessor architecture for mixed-time-criticality applications," *Handbook of Hardware/Software Codesign*, pp. 491–530, 2017.
[9] P. Marti *et al.*, "Jitter compensation for real-time control systems," in *Real-Time Systems Symposium, 2001.(RTSS 2001). Proceedings. 22nd IEEE*. IEEE, 2001, pp. 39–48.
[10] W. Heemels *et al.*, "An introduction to event-triggered and self-triggered control," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 3270–3285.
[11] H. Yan *et al.*, "H output tracking control for networked systems with adaptively adjusted event-triggered scheme," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–9, 2018.
[12] M. Izák *et al.*, "Stability and control of systems with uncertain time-varying sampling period and time delay," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 11 514–11 519, 2008.
[13] J. Valencia *et al.*, "Composable platform-aware embedded control systems on a multi-core architecture," in *Digital System Design (DSD), 2015 Euromicro Conference on*. IEEE, 2015, pp. 502–509.
[14] K. J. Åström and B. Wittenmark, *Computer-controlled systems: theory and design*. Courier Corporation, 2013.
[15] P. Marti *et al.*, "On real-time control tasks schedulability," in *Control Conference (ECC), 2001 European*. IEEE, 2001, pp. 2227–2232.
[16] J. Valencia *et al.*, "Resource utilization and quality-of-control trade-off for a composable platform," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*. IEEE, 2016, pp. 654–659.
[17] B. P. Rigney *et al.*, "Nonminimum phase dynamic inversion for settle time applications," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 989–1005, 2009.
[18] M. Tomizuka, "Zero phase error tracking algorithm for digital control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 109, no. 1, pp. 65–68, 1987.
[19] J. A. Butterworth *et al.*, "Architectures for tracking control in atomic force microscopes," *IFAC Proceedings Volumes*, pp. 8236–8250, 2008.
[20] M. B. Cloosterman *et al.*, "Stability of networked control systems with uncertain time-varying delays," *IEEE Transactions on Automatic Control*, vol. 54, no. 7, pp. 1575–1580, 2009.
[21] K. Ogata, *Discrete-time control systems*. Prentice Hall Englewood Cliffs, NJ, 1995, vol. 2.
[22] W. Geelen *et al.*, "The impact of deadline misses on the control performance of high-end motion control systems," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 2, pp. 1218–1229, 2016.
[23] J. Boot, *Frequency response measurement in closed loop: brushing up our knowledge*, ser. DCT rapporten. TU/e, 2003, dCT 2003.059.