2_3_Pipeline Model

1. Pipeline Timing Analysis

For pipelined model, we will still use the example IBC system used in last part. But now, we assume we may use multiple core or we can build an pipeline structure.

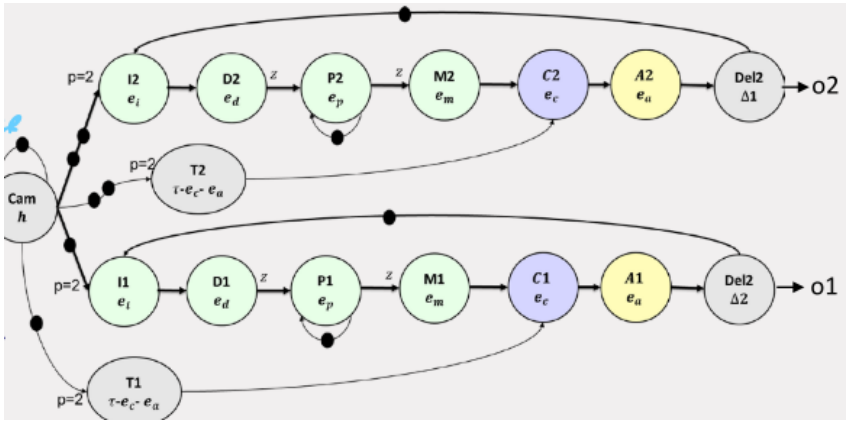
- Long sensor-to-actuator delay can degrade the control performance (e.g., longer settling time) if the controller is implemented sequentially
- Pipelining the sensing operation is a way tackle this design issue

Two Core Cases

- If frames can be processed **independently**, we can increase the rate of sensor-data processing by **processing frames in parallel on multiple cores**
- Pipelining reduces the sampling period
- Another requirement is: **we need to guarantee the output interval are the same**

Normal Model

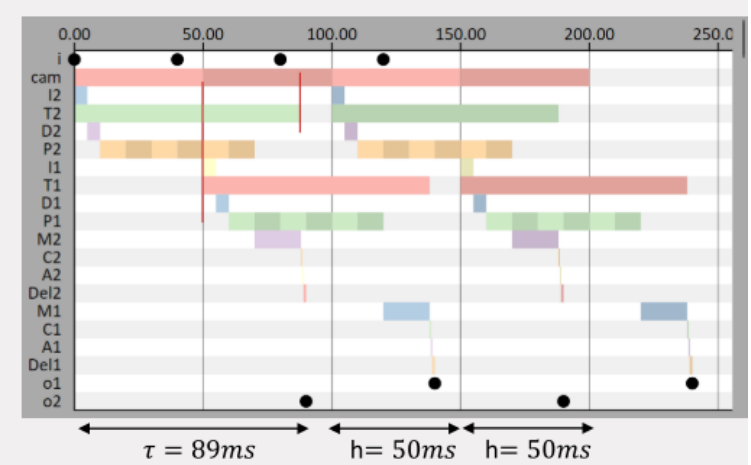
For two pipeline case, based on the last above requirement, we can first draw the SDF model:



and we can calculate:

- sampling period $h = \lceil \frac{\tau}{p * f_h} \rceil f_h = 50$ ms
- $\Delta_1 = \Delta_2 = p \cdot h - \tau = 11$ ms

The execution schedule is shown in the following Figure



Enforcing $\tau = p \times h$

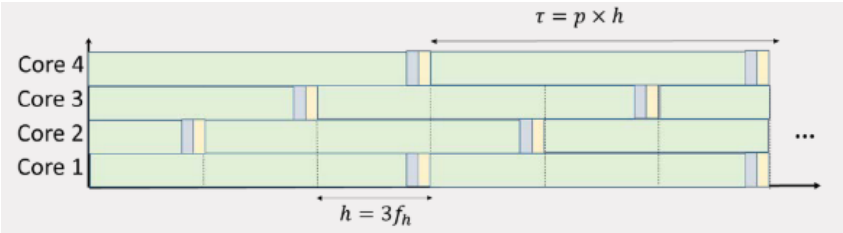
Because the analysis of controller design is quite complex (not in logic, but in augmented state dimension), we can enforce $\tau = p \times h$ to simplify the problem. Which means, the sensor-actuator delay is multiply of sampling time, so in a given timestep of sampling time, the input u will keep the same

- We can change the value of T_1 and T_2 to enforce $\tau = p \times h$

$$\mathrm{T}_1 = \mathrm{T}_2 = \mathrm{p} \times h - e_a - e_c = \quad (100 - 0.5 - 0.5)ms = 99ms$$

More Pipeline Case

- Three pipeline is almost the same
- For the 4 pipeline case, we will find that the Core 4 **does not help**. It repeats the activities in Core 1.



Discussion: Intuition of Calculating h

τ/p means if we want to arrange p execution (average time so we use divide), how long is the interval between two execution.

And then the intuition is the same as sequential model.

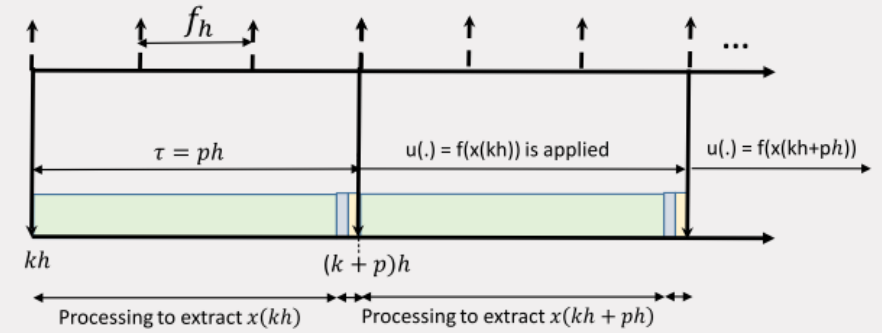
2. Controller Design

Here, we consider the model with enforcing $\tau = p \times h$

- sampling time $h = \lceil \frac{\tau}{p * f_h} \rceil f_h$
- $\tau_{eff} = h \cdot p$

Basic Model

Because we use enforcing model, hen the input-sensing relation is shown in the graph



That means we have

- Continuous-Time model: $\dot{x} = Ax(t) + Bu(t - \tau_{eff})$
- Discrete-Time model: $x(kh + h) = \phi x(kh) + \Gamma u(kh - ph)$, where:

$$\phi = e^{Ah}$$
$$\Gamma = \int_0^h e^{As} B ds$$

Augmented Model

In order to solve the control problem and use control theory, we need to transform the delayed model into non-delayed model

We define augmented state z:

$$z(kh) = \begin{bmatrix} x(kh) \\ u(kh - ph) \\ u(kh - ph + h) \\ u(kh - ph + 2h) \\ \cdots \\ u(kh - h) \end{bmatrix}$$

Then we will have:

$$z(kh + h) = \begin{bmatrix} x(kh + h) \\ u(kh - ph + h) \\ u(kh - ph + 2h) \\ u(kh - ph + 3h) \\ \cdots \\ u(kh) \end{bmatrix} = \begin{bmatrix} \phi x(kh) + \Gamma u(kh - ph) \\ u(kh - ph + h) \\ u(kh - ph + 2h) \\ u(kh - ph + 3h) \\ \cdots \\ u(kh) \end{bmatrix} = \begin{bmatrix} \phi & \Gamma & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} z(kh) + \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ 1 \end{bmatrix} u(kh)$$

Summary

- Can use pipeline to **pipelining the whole task:**
  - always used when the details of **processing step is not known**, that is we cannot divide the task into several steps
  - $\tau > h$, for augmented state model, we need more previous input
  - **marginal effect**: after arriving at a given number of pipeline, more processor will have no improvements