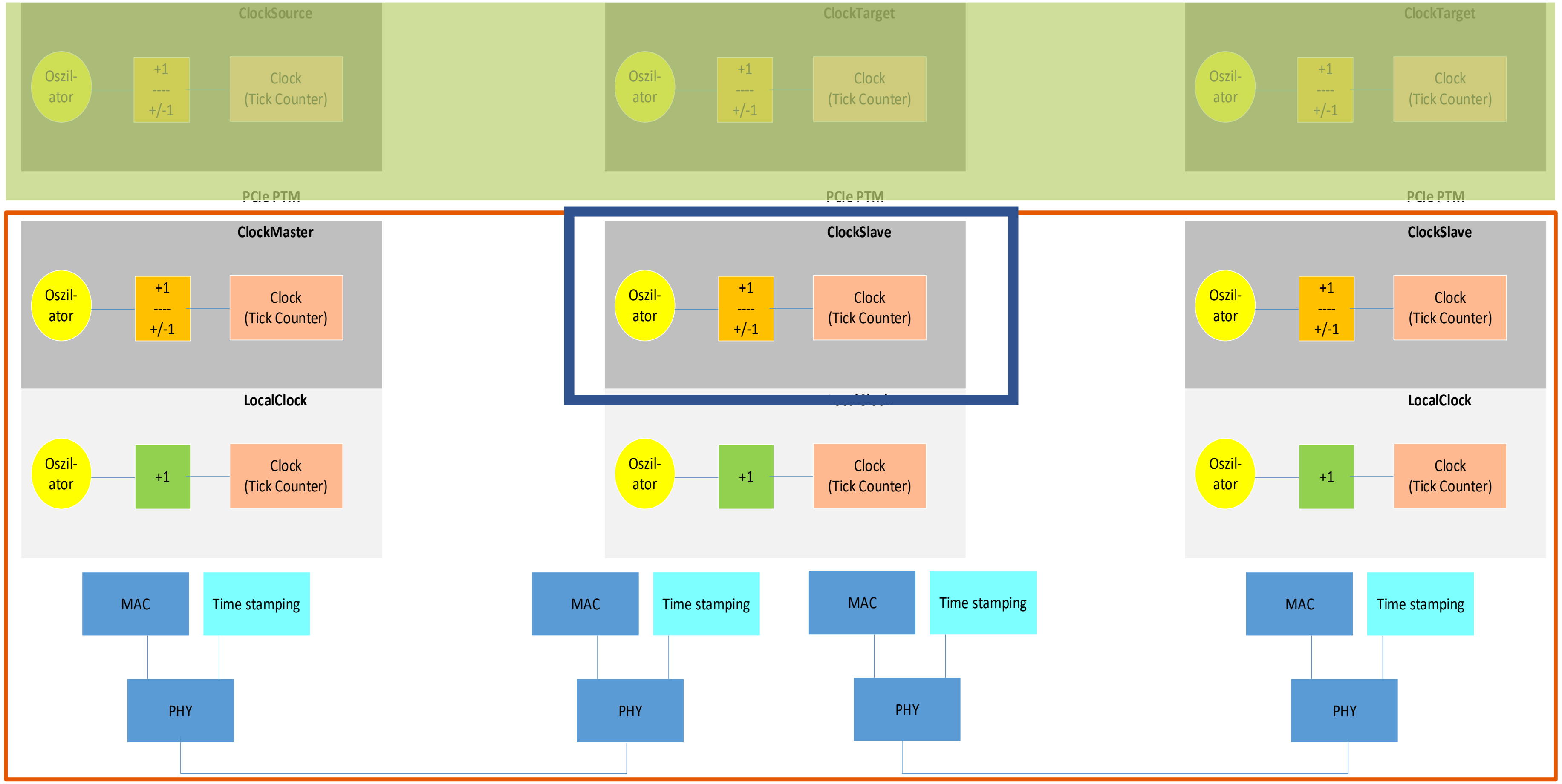
ClockSlave PI Controller: Definition and Implementation

Dragan Obradovic, Siemens AG

ClockSlave


ClockSource
Oszil-
ator
+1

+/-1
Clock
(Tick Counter)
PCIe PTM
ClockTarget
Oszil-
ator
+1

+/-1
Clock
(Tick Counter)
PCIe PTM
ClockTarget
Oszil-
ator
+1

+/-1
Clock
(Tick Counter)
PCIe PTM
ClockMaster
Oszil-
ator
+1

+/-1
Clock
(Tick Counter)
LocalClock
Oszil-
ator
+1
Clock
(Tick Counter)
ClockSlave
Oszil-
ator
+1

+/-1
Clock
(Tick Counter)
Oszil-
ator
+1
Clock
(Tick Counter)
ClockSlave
Oszil-
ator
+1

+/-1
Clock
(Tick Counter)
LocalClock
Oszil-
ator
+1
Clock
(Tick Counter)
MAC
Time stamping
PHY
MAC
Time stamping
PHY
MAC
Time stamping
PHY
MAC
Time stamping
PHY

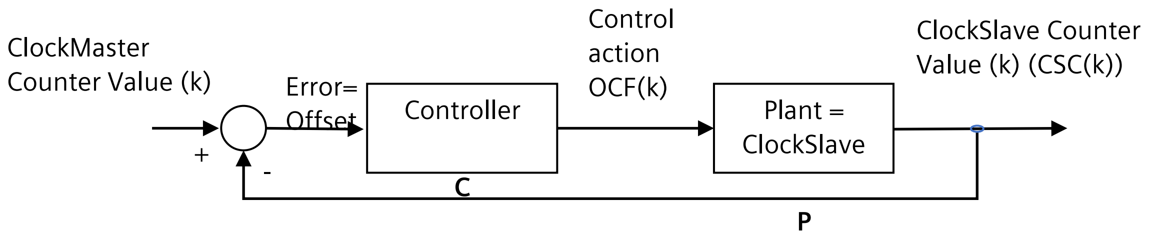
ClockSlave Control Loop

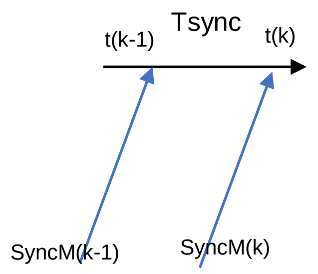
- Control loop necessary to enable „**tracking**“ of the ClockMastertime (“MasterTime”) by ClockSlave

- **Offset**: difference between the MasterTime the slave “n” becomes via Sync Messages (corrected by the pDelay) and the ClockMaster time

- **OCF**: controller output, which scales the frequency of the free running clock (e.g. LocalClock) in order to minimize Offset

- Frequency scaling achieved by changing the number of ticks of the free running clock by ± 1 in appropriate time intervals





- Controller is active only at the arrival of Sync Message
- Sync Messages arrive periodically with the period of ~Tsync (Sync interval at the GM)

➔ The controller is practically time-discrete

ClockSlave Control Loop: Time Discrete Representation

- **Controller**: → time discrete PI controller

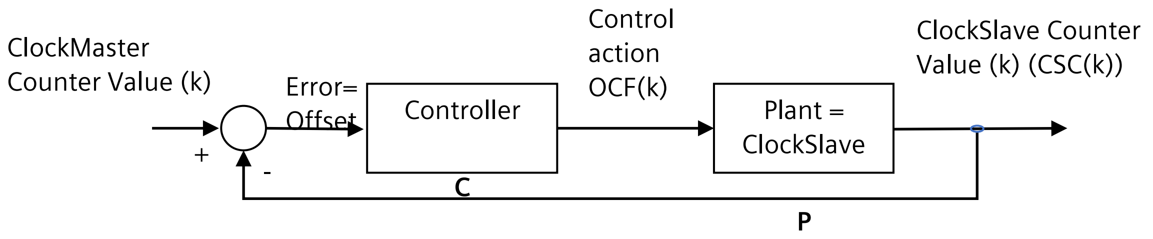
$$OCF(k) = OCF(k-1) + K_p \cdot (Offset(k) - Offset(k-1)) + K_I \cdot Offset(k-1) \cdot Tsync$$

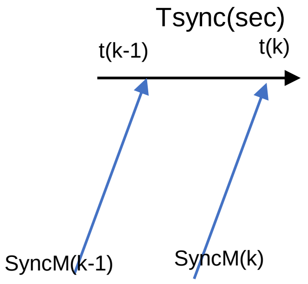
$$OCF(z) = \frac{K_p \cdot (z-1) + K_I \cdot Tsync}{z-1} \cdot err(z)$$

- **Plant** (ClockSlave): an integrator of the free running clock frequency (fnom: nominal and ft: true frequency)

$$CSC(k) = CSC(k-1) + OCF(k-1) \cdot f_t \cdot Tsync$$

$$CSC(z) = \frac{f_t \cdot Tsync}{z-1} \cdot OCF(z)$$



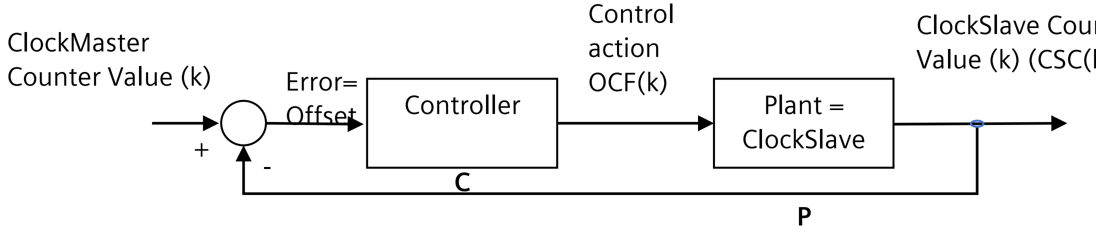


$$K_p = \frac{K_{p1}}{fnom \cdot 1sec}; \; K_I = \frac{K_{I1}}{fnom \cdot 1sec^2}$$

Where K_{p1} and K_{I1} are the parameters of the time continuous PI controller) optimized for a pure integrator) as the plant which was discretized

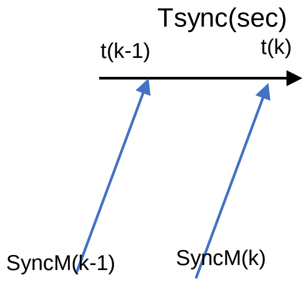
ClockSlave Control Loop: Time Discrete Representation

- **OpenLoop**: → P*C



$$CSC(z) = \frac{f_t \cdot Tsync}{z-1} \cdot \frac{K_p \cdot (z-1) + K_I \cdot Tsync}{z-1} Offset(z)$$

$$= \frac{Tsync \cdot f_t}{f_{nom}} \cdot \frac{K_{p1} \cdot (z-1) + K_{I1} \cdot Tsync}{(z-1)^2} \cdot Offset(z)$$



→ The plant, and consequently the whole control loop, is linear time-invariant only if f_t is constant

→ If f_t is constant but different from the nominal frequency, the ratio $\frac{f_t}{f_{nom}}$ changes the gain of the OL; the controller has to be able to deal with this uncertain gain (its gain margin has to be appropriately large)

$$K_p = \frac{K_{p1}}{fnom \cdot 1sec}; \; K_I = \frac{K_{I1}}{fnom \cdot 1sec^2}$$

Where K_{p1} and K_{I1} are the parameters of the time continuous PI controller) optimized for a pure integrator) as the plant which was discretized

ClockSlave Control Loop: Time Discrete Representation and Implementation

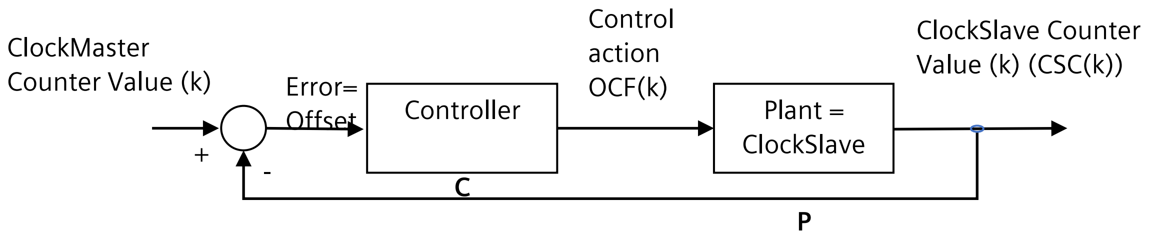
- **Controller**: → time discrete PI controller

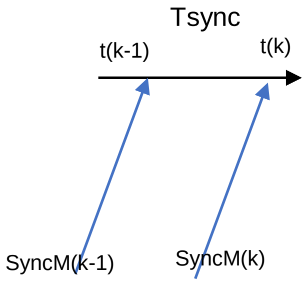
OCF provides information about how many ticks have to be added/subtracted to the number of ticks of the free running clock (e.g. Local Clock) within Tsync

→ We can calculate the time interval in the free running clock (its number of ticks) when the change of one tick takes place

$$OCF_{interval}(k) = round\left(\frac{sign(OCF(k) - 1)}{OCF(k) - 1}\right)$$

The change in ticks is given by: $sign(OCF(k) - 1)$, hence it can be zero or ± 1



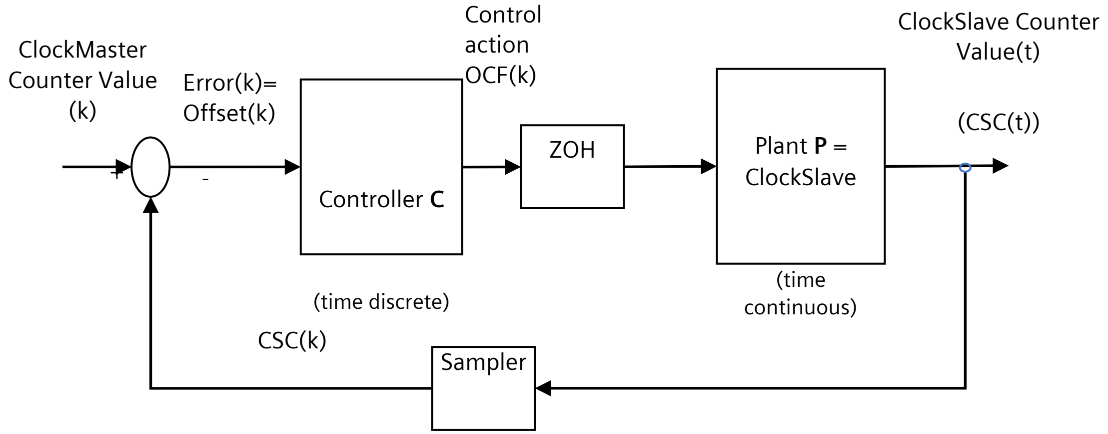


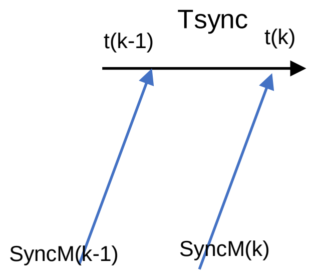
ClockSlave Control Loop: Mixed Time Discrete & Time Continuous Representation

- Time discrete representation of the closed loop: signals are calculated and transmitted only at the discretization events (arrival of Sync Messages)

- <u>But ClockSlave provides time continuously!</u>

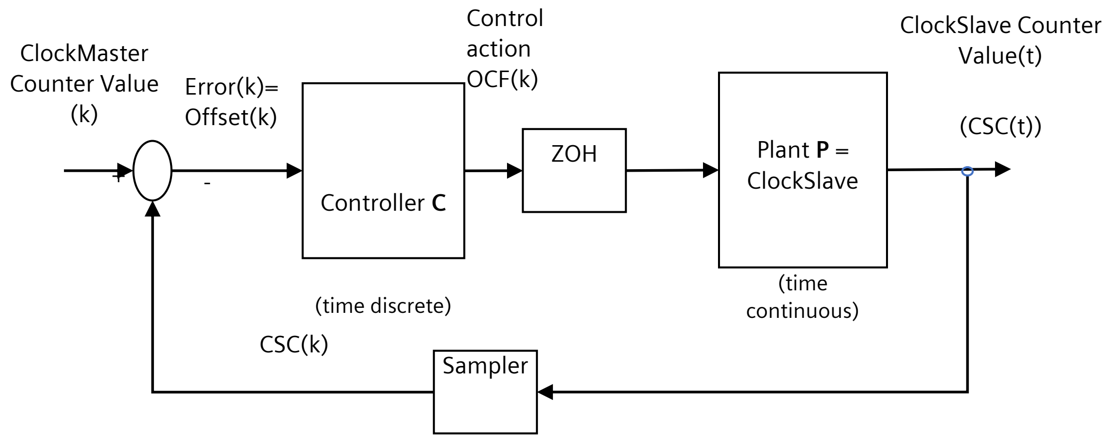
- How is this achieved?: → by ZOH function, i.e. by making the time discrete output of the PI controller constant within the sampling interval



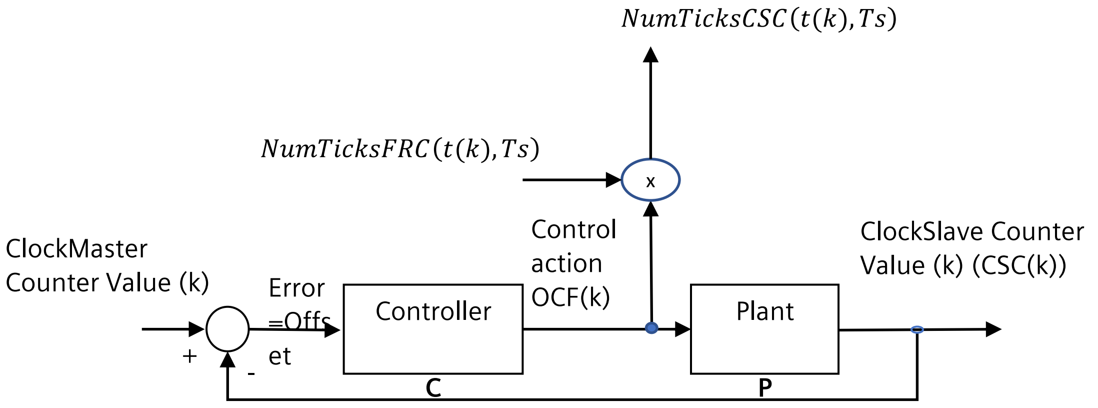


ClockSlave Control Loop: Mixed Time Discrete & Time Continuous Representation

- Alternative Representations



- The ClockSlave time is available continuously!



ClockSlave Control Loop Characteristics

- **Bandwidth**: determined by Tsync (we cannot track frequencies of the MasterTime changes with a period < Tsync)

→ have Tsync and pDelay request intervals small

- **Noise**: the MasterTime is noisy (we have to keep the signal2noise ratio large enough)

→ reduce stamping errors, have Tsync and the pDelay intervals large (contradicting the bandwidth recommendation)

- **Tracking Delay**: MasterTime information is delayed by aggregated ResidenceTimes and pDelays; Other delays also play a role

→ Have short ResidenceTimes