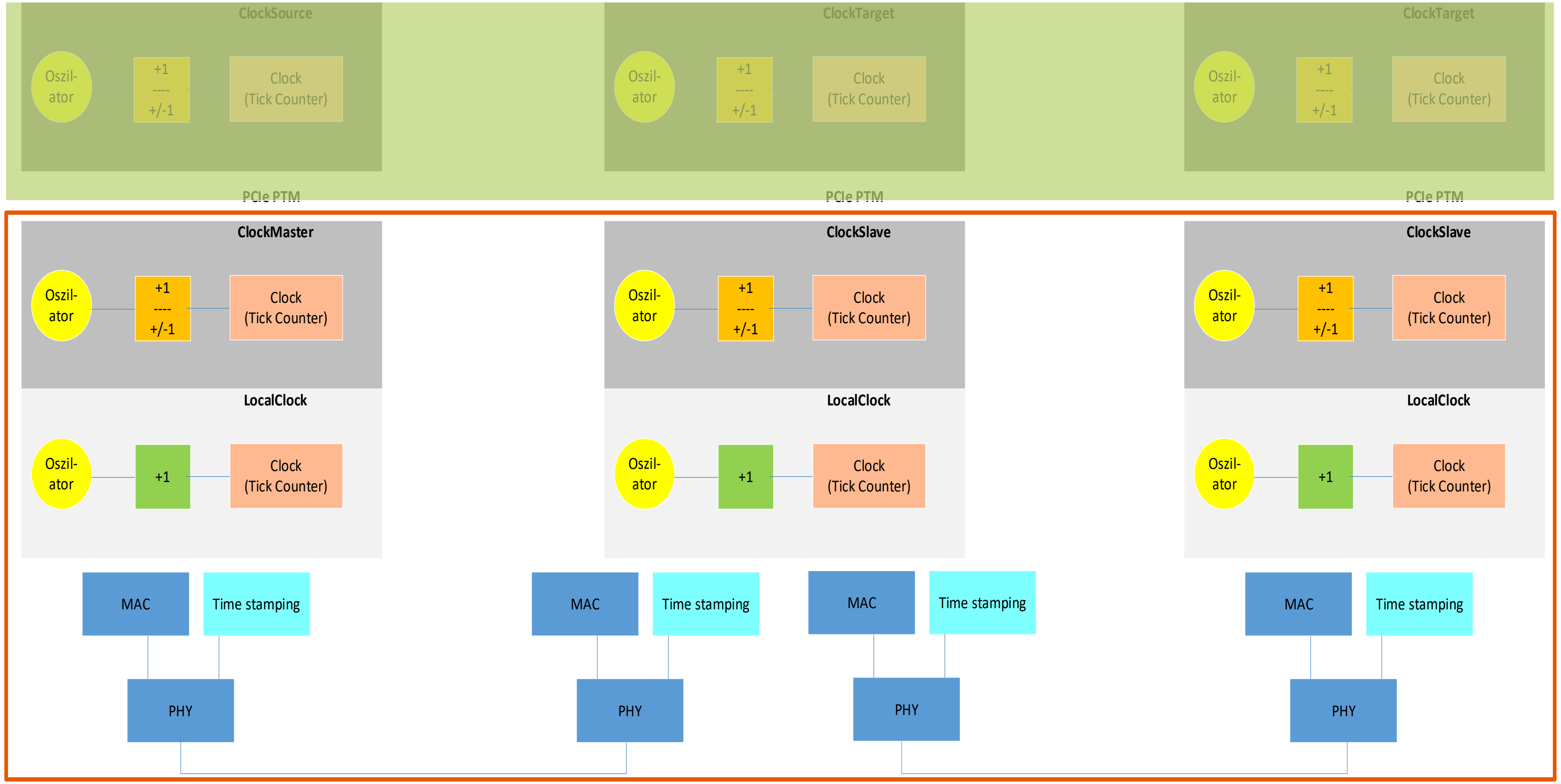
ClockSlave PI Controller: Definition and Implementation

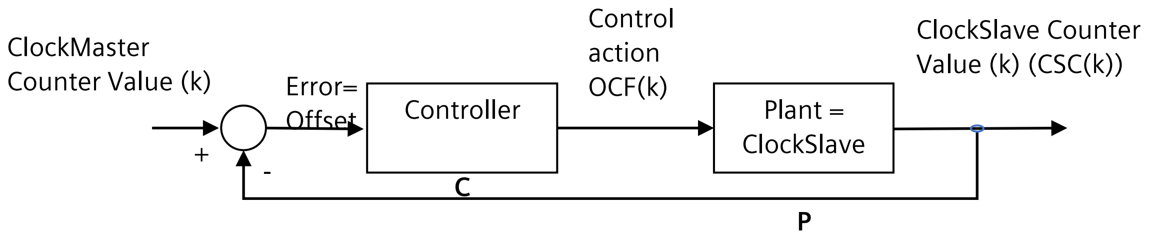
Dragan Obradovic, Siemens AG

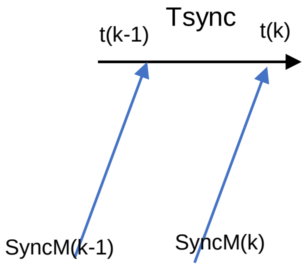
ClockSlave


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

ClockSlave Control Loop

- Control loop necessary to enable „**tracking**“ of the ClockMaster time 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


ClockMaster Counter Value (k)
+
-
Error= Offset
Controller
C
Control action OCF(k)
Plant = ClockSlave
P
ClockSlave Counter Value (k) (CSC(k))



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

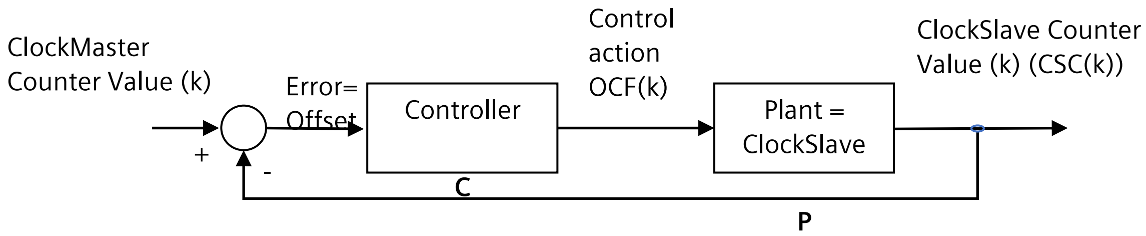
$$OCF(k) = OCF(k-1) + K_p \cdot (Offset(k) - Offset(k-1)) + K_I \cdot Offset(k-1) \cdot Tsync$$

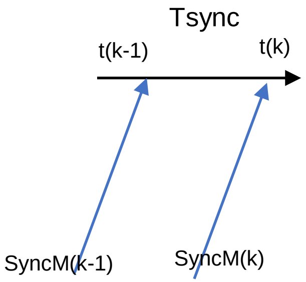
$$CSC(z) = \frac{f \cdot Tsync}{z-1} \cdot OCF(z)$$

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

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

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


ClockMaster Counter Value (k)
Error= Offset
+
-
Controller
C
Control action OCF(k)
Plant = ClockSlave
P
ClockSlave Counter Value (k) (CSC(k))



$$K_p = \frac{XX}{fnom \cdot 1sec}; \; K_I = \frac{YY}{fnom \cdot 1sec^2}$$

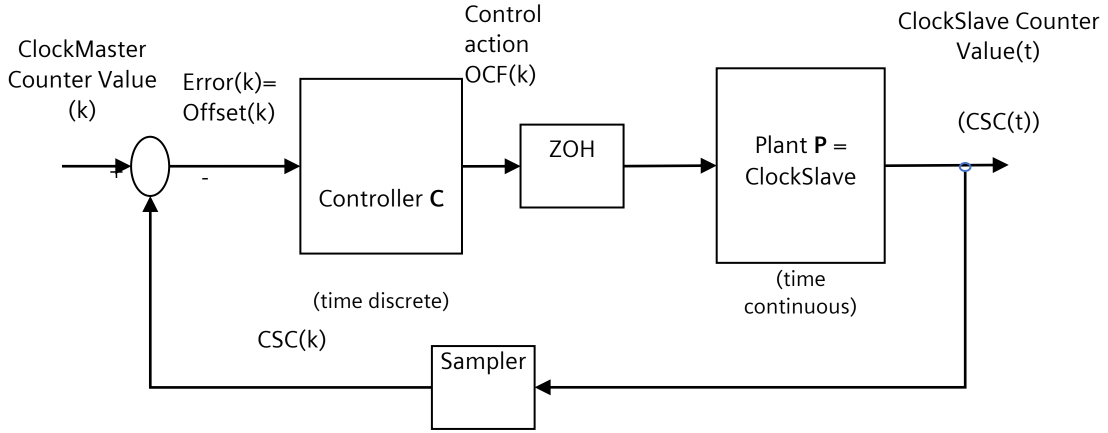
Where XX and YY are the parameters of the time continuous PI controller which was discretized

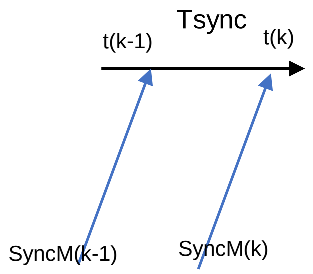
ClockSlave Control Loop: Time Discrete Representation

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

- But ClockSlave provides time continuously

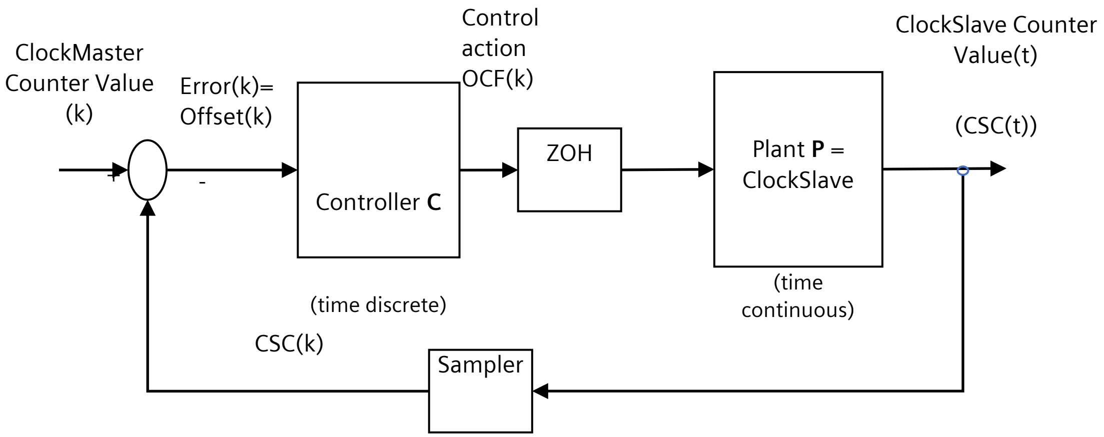
- How is this achieved: by ZOH function, i.e. by making the time discrete output of the PI controller constant within the sampling interval

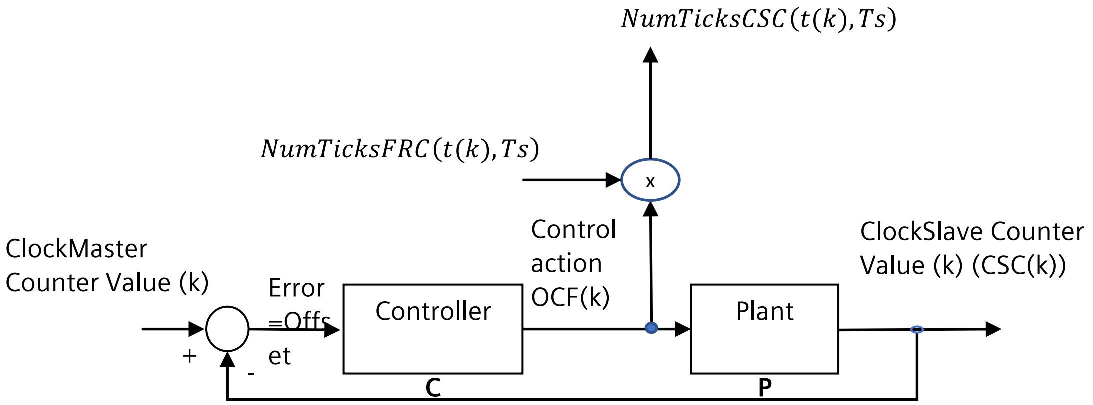




ClockSlave Control Loop: Mixed Time Discrete & Time Continuous Representation

- Alternative Representation





Limitation of the ClockSlave Control Loop

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

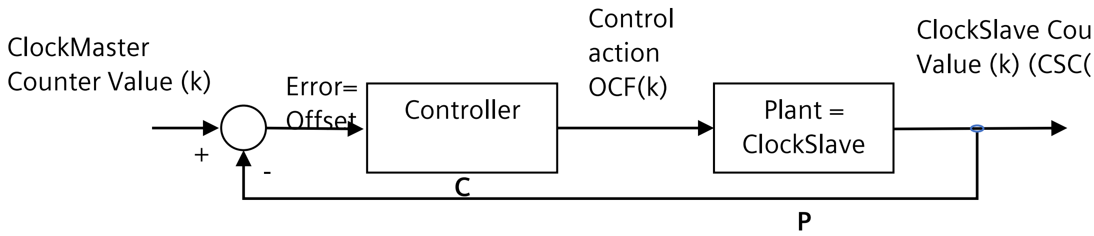
→ Have Tsync small

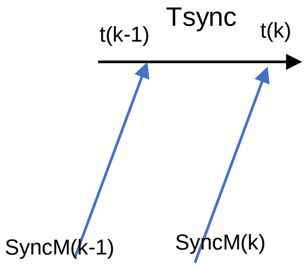
- **Noise**: the MasterTime is noisy (we have to keep the signal2noise ratio large enough

→ reduce stamping errors

- Delay: MasterTime information is delayed by aggregated ResidenceTimes and pDelays as well as the Tsync

→ Short ResidenceTimes


ClockMaster Counter Value (k)
Error= Offset
Controller
Control action OCF(k)
Plant = ClockSlave
ClockSlave Counter Value (k) (CSC(k))
+
-
C
P



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