FAN	Project: FlexRay Analysis	FlexRay Consortium Uni Duisburg-Essen
University of Duisburg-Essen, ICB / Informatik, 45117 Essen, Germany Sabine Böhm, Prof. Dr. Klaus Echtle, Jens Lisner		
A4c Sep. 14, 2002	Fault-tolerance of the FlexRay Clock Synchronization (Extended) Workpackage A4	

Fault-tolerance of the FlexRay Clock Synchronization (Extended)

Klaus Echtle, Jens Lisner

Abstract

This paper presents a study of the fault-tolerant distributed clock synchronisation used in the FlexRay distributed network protocol, which is currently under development. This algorithm is meant to keep the clocks synchronized with low error margins. Extra messages are not used. Instead the frames sent in a TDMA time pattern are observed. The algorithm provides both rate and offset correction. The study focuses on the offset correction part of the algorithm. The rate correction has been investigated in [Müller2001]. It is shown that a faulty node cannot shift the start of a cycle beyond known limits both in the byzantine and non-byzantine case. Sufficent convergence is obtained provided that some conditions are satisfied. Algorithmic issues of the used variant of the FTA decision function are dealt with in [Study A4-2].

Contents

1 Introduction

FlexRay is a network protocol for use in automotive environments. It features real time, as well as fault tolerance capabilities. FlexRay works in comunication cycles, which are in general devided into a static and a dynamic part. The static part works in a TDMA fashion which means that time slots are formed to manage the bus arbitration. This introduces the need of a known time base for all participating nodes. In FlexRay a distributed time base is used. Every node uses its own local hardware oscillater which has to synchronize to other clocks in the network in regular intervals, to prevent e.g. collisions on the bus. FlexRay contains a distributed clock synchronization algorithm consisting of two parts: a rate and an offset correction. This paper investigates the FlexRay method for offset correction. It is based on the work by the FlexRay Group, in particular [PWD 0.6.0], [Müller2001] and [ReqSpec2].
Another concept is discussed in [Kopetz]. Concepts in comparision to FlexRay are discussed in [Rushby2001] and [Kopetz2001].

2 System Model

FlexRay is a network consisting of a set $\mathcal{N}$ of nodes n_i, with $i \in \{1, \ldots, |\mathcal{N}|\}$. The physical layer network consists of two broadcast channels A, B. All nodes are connected to either one or both channels. This is represented by a function $\mathbf{CH} : \mathcal{N} \longmapsto \{A, B, AB\}$. In FlexRay time measurement is done by measuring the arrival times of sync frames. Sync frames can only be sent once per cycle by sync masters $\mathcal{S} \subseteq \mathcal{N}$. The "sync slaves" [1] are passive participiants to the FlexRay clock synchronization algorithm. While sync slaves may be connected to only one channel, sync masters are always connected to both channels, this means that $\forall n_i \in \mathcal{S} : \mathbf{CH}(n_i) = AB$. Nodes connected to both channels use the minimum of the values measured via channels A and B. Implications of this principle are dealt with in section 4.6.
All nodes in a FlexRay network use a hardware oscillator with a fixed frequency as clock. As basis for all clocks, t denotes the real time, which is used throughout this paper as "perfect" reference clock, and as a comparable time base for the local clock time in different controllers. Real time may be measured e.g. in units of nanoseconds or bit time. The notion of a local time is built on top of microticks. Microticks have a nominal length in real time. The real length of a microtick varies through inaccurencies of the local hardware oscillators. In FlexRay the nominal time in units of bit time may be different for different clocks. This is necessary for balancing the timing of oscillators of known different qualities.
The time in FlexRay is counted in notions of macroticks and cycles. A macrotick is defined as a nominal number of microticks. In the same manner cycles are defined through macroticks. The clock synchronization is responsible for two different conditions to hold: First, all cycles in all clocks should have approximately the same duration in real time. The rate correction is responsible for this. Second, the difference in real time between the points in time when a particular clock time is reached by different clocks should be as small as possible which is the job of the offset corrections. To achive this the clock synchronization has to work in

[1] This is not official FlexRay terminology.

notions of microticks. Synchronization through offset correction is done each double cycle. The correction value is derived from the measurement of the sync masters clocks. These measurements are done differently for rate and offset correction. While rate correction observes both cycles and uses the deviation between the measurements in both cycles, offset correction measures only once the time distances between remote clocks and its own clock. This measurement is performed once in the second cycle of a double cycle. Let z be the length of one cycle in microticks. This value may vary each double cycle since rate correction adapts the clock speed by stretching or shortening the cycle through changing the number of microticks. Since this investigation focuses on the invariant that has to hold for offset correction through a double cycle, this fact is not relevant. The effects of rate correction can be taken into account by a lower inaccuracy of a local clock.
The synchronization in a local node is done after $2z$ microticks. In general this is not done at the same points in real time.
In this investigation a worst case scenario is assumed with two clocks s and f with the maximum real time distance at every clock time, with s the slowest and f the fastest clock. The purpose of the offset correction is to keep the real time distance of all clocks in given bounds after correction every second cycle. Therefore an invariant condition must be fulfilled.

3 The Algorithm

In this section the offset correction part of the FlexRay clock synchronization is described. The algorithm is split up into three phases. First the measurement is done every second cycle. The measured values are the input to the calculation phase, which gives a correction value which is applied in the network silence phase at the end of the double cycle. All calculations in the nodes are done in local clock time. The basics of the clock synchronization in general are discussed in [Lynch88], [Lynch96], [Tel00], [SchWeiss] and [Lönn99]. A comparision between several algorithms is found in [AncPua98].
[PfSchwHe99] and [PalGra92] shows example verification of clock synchronization in time-triggered context.

3.1 Measurement

All nodes use their clocks to calculate the time when the frames from all other nodes are expected. Typically the frame is not received at that time. The difference of the actual arrival time and the expected time is taken as clock difference between sender and receiver. This is done only for sync frames in the second cycle of the double cycle. The sync frame experiences a delay on the physical layer network. This causes a measurement error. To keep this as low as possible, the delay is compensated by a value which is derived from the minimum known delay a frame may exhibit between two nodes. [2] These values are highly dependant on the network topology. Important factors are for example wire length, number of star couplers to pass and internal transceiver delays.
The resulting value is the measurement value $M_{i,j}$ by which node $n_i \in \mathcal{N}$ is measuring a sync node $n_j \in \mathcal{S}$.

[2] At the time of writing this was not finally decided. Another possibility is to use the half of the expected maximum delay.

Number of values	k-value
$1 \ldots 2$	0
$3 \ldots 7$	1
≥ 8	2

Table 1: Adaptive k-value selection

$M_{i,i}$ is the "difference" to the own local clock which is always zero: $M_{i,i} = 0$.

3.2 Calculation

After the static part every (fault free) node has a vector of measurement values. This is used to calculate a correction value. For this purpose the FTA (Fault Tolerant Midpoint Algorithm) as described in [Lynch88] is used.
This method is defined for $|\mathcal{S}| \geq 3F + 1$ clocks with F the maximum number of tolerated faults. After sorting the vector of measurement values the F smallest and F greatest values are deleted. The maximum and minimum of the values left are taken, and the arithmetic average of these values, called mid-value FTA_i, is taken as correction value for node n_i.
For FlexRay this algorithm was slightly modified to adapt the number of values to be deleted (the k-value) to the number of measurement values in the vector. Table 1 shows how the k-value is selected. While it is possible to use these values in the case of benign faults, it is important to point out that k is not always equal to F. With three values it is not possible to tolerate arbitrary faults. On the other hand with more than six values it is in general possible to tolerate two faults. By using $k = 1$ in the case of seven values, as chosen in FlexRay, only one fault can be tolerated, although the generally proven formula $|\mathcal{S}| \geq 3F + 1$ allows for double fault tolerance in this case ($|\mathcal{S}| = 7$, $F = 2$). For further discussions on this topic see [Study A4-2].

3.3 Correction

The offset correction is done at the end of the double cycle during the network silence phase. The strategy is to shorten or extend the length of the macroticks by a correction value expressed in a number of microticks. The modification of the length of one macrotick is limited. The microticks have to be distributed over the first few macroticks in the network silence phase. Extending macroticks means to halt the clock for a while, while shorten microticks has the effect of speeding it up for a short span of time.

4 Proof of Convergence

The goal of the FlexRay offset correction algorithm is to synchronize all clocks every two cycles. To proof the correctness (which means sufficient convergence) of the offset correction algorithm a worst case scenario

is considered. The initial conditions are presented first, and then the steps measurement and correction are investigated. The goal is an invariant condition that has to be fulfilled to prevent the clocks from divergence. Sync slaves are not considered in this scenario, because they are no active participants and do not influence other clocks in correction value calculation. The properties of the offset correction in conjunction with sync slaves is discussed in section 4.5. The consequences of the presence of two channels are discussed in section 4.6.
All symbols used in this section are summerized in appendix D.

4.1 Preconditions

$C_i(t)$ represents the local clock time of clock i at real time t, and $T_i(c)$ the inverse function. The real time distance $\Delta_{i,j}(c) = |T_i(c) - T_j(c)|$ between two clocks i and j at a particular clock time c is defined as the distance of the points in real time when the local clocks are reaching clock time value c. In this scenario the earliest clock f which starts the new cycle is considered to be the fastest, the last clock s is considered to be the slowest. It is pointet out, that s and f exhibit the most extremal behaviours which still satisfy the specification for faultless clocks. The real time when the clocks f and s start the cycle are given by $T_f(0)$ and $T_s(0)$, respectively. The initial maximum distance of all clocks is defined as $\Delta_0 = \Delta_{s,f}(0)$.
Let r_i be the inaccuracy of a clock i. The inaccuracy (typically expressed in ppm: parts per million) is defined as the relative deviation of clock speed to the progress of real time. For a perfect clock $r_i = 0$ holds. An inaccuracy $r_i = -0.0001$ characterizes a clock which is 100 ppm too slow. The maximum inaccuracy r is defined as $r = \max_{i \in \mathcal{F}}\{r_i\}$, where $\mathcal{F} \subseteq \mathcal{N}$ is the set of faultfree nodes.
To convert a portion of real time to a local clocks' value, the factor $\gamma_i = 1 + r_i$ is defined for clock i. To do the conversion from a local clock time to real time the factor $\tau_i = \frac{1}{1+r_i}$ is introduced. With these definitions the local clock time for a clock i is

$$C_i(t) \quad = \quad \gamma_i t - \gamma_i T_i(0) \tag{1}$$

with the inverse function

$$T_i(c) \quad = \quad \tau_i(c - C_i(0)) \tag{2}$$

where

$$C_i(0) \quad = \quad -\gamma_i T_i(0) \tag{3}$$

To achive the worst case scenario $r_s = -r_f = -r$ is assumed. This makes $\gamma_s = 1 - r$ and $\tau_s = \frac{1}{1-r}$ as well as $\gamma_f = 1 + r$ and $\tau_f = \frac{1}{1+r}$. The behaviour of clocks s and f is illustrated by a diagram in appendix B.

4.2 Measurement

In the worst case scenario the measurement takes place right at the beginning of the cycle so that the measured values are most likely to get inaccurate. While measuring the distance between the expected and the actual

arrival time the signal experiences a real time delay $d_{i,j}$ on the physical layer network, when the signal is sent from node n_j to receiver node n_i. Generally this value varies from transmission to transmission. Let $d = \max_{i,j}\{d_{i,j}\}$. This is compensated by a value o, defined in the system configuration (typically expressed by a number of microticks). The local expected signal arrival time in a clock i is z, the duration of the first cycle within the double cycle. Assuming the measurement right at the beginning of the second cycle is most pessimistic, because this may lead to the maximum inaccuracy at the end of the second cycle. Transformed to real time, the sender n_j sends its signal at time $T_j(z)$. With delay and compensation the measurement value $M_{i,j}$ of a clock i measuring a clock j is

$$M_{i,j} \quad = \quad C_i(T_j(z) + d_{i,j}) - z - o \tag{4}$$

In appendix B an example is given: the measurement $M_{f,s}$, where the fastest clock f measures the slowest clock s. The measurement error ϵ_i of measuring node n_i is the uncompensated portion of the delay in real time. Note that ϵ_i can be both positive or negative, depending whether the delay is partially or overly compensated. This depends on the compensation method. According to (5) the extremal values of ϵ_i, denoted by ϵ_{min} and ϵ_{max}, respectively, depend on the variation of the delay $d_{i,j}$ and the value of the compensation o. To be independent from the compensation method, let $\epsilon_{min} = \min\{0, \epsilon_1, \ldots, \epsilon_{|\mathcal{N}|}\}$ and similar $\epsilon_{max} = \max\{0, \epsilon_1, \ldots, \epsilon_{|\mathcal{N}|}\}$ be the set of all measrement errors.
A special case exists where sync master n_i measures itself. In this case the result of the measurement is set to $M_{i,i} = 0$.

$$\epsilon_i \quad = \quad d_{i,j} - \tau_i o \tag{5}$$

This can be used to get to an alternative formula for $M_{i,j}$.

$$\begin{aligned} M_{i,j} &= C_i(T_j(z) + d_{i,j}) - z - o \\ &= \gamma_i T_j(z) + \gamma_i d_{i,j} - \gamma_i T_i(0) - C_i(T_i(z)) - \gamma_i \tau_i o \\ &= \gamma_i(T_j(z) - T_i(0) + \epsilon_i) - \gamma_i(T_i(z) - T_i(0)) \\ M_{i,j} &= \gamma_i(T_j(z) - T_i(z) + \epsilon_i) \end{aligned} \tag{6}$$

If the measuring clock i is faster than the measured clock j, then it reaches z earlier than clock j. This means $T_i(z) < T_j(z)$ and thus $T_j(z) - T_i(z) > 0$. From this inequality we can conclude the tendency: A clock i which has observed mainly positive measurements $M_{i,j}$ should be slowed down. A sufficient number of negative measurements will cause a speed up of clock i instead. See section 4.3 for details on correction.
Since ϵ_i may vary it can become ϵ_{min} or ϵ_{max}. By $M_{i,j,min}$ we denote the measurement value $M_{i,j}$ with a measurement error of ϵ_{min}. $M_{i,j,max}$ is defined accordingly.

4.3 Correction

The maximum distance of clocks i and j after two cycles is defined as $\Delta_1 = \Delta_{i,j}(2z)$. After applying the correction value FTA_i in clock time the clock i starts its next cycle at real time $U_i = T_i(2z + FTA_i)$. After correction the distance of clocks i and j is changed to $\Delta_2 = |U_s - U_f|$ at most.
In FlexRay the calculated number of microticks is distributed during the network idle time. While the clocks are still inaccurate in this process, the real time $\tau_i FTA_i$ passess. This effect is also taken into account by the definition of U_i.

4.4 Convergence

To show convergence $3F + 1$ clocks are needed, where F is the maximum number of faults to be tolerated. To obtain the worst case, i.e. the maximum difference between U_f and U_s, the following scenario is assumed: The number of sync masters is exactly $3F+1$. A group of $F+1$ faultfree clocks are all slowest and completely identical to each other. Furthermore, a group of $F - 1$ faulty clocks are slowest as well. Hence, at least one (faulty or faultless) clock q besides clock s is slowest. q and s are two different elements in the group of slowest clocks. A further group of F faultfree clocks are all fastest and completely identical to each other. Clock f belongs to this group. A single faulty clock p is fastest and sends its signal such that it is accepted by clock f, but ignored by clock s. In all, these are $3F + 1$ clocks. After clock s has removed F smallest and F greatest measurements only slowest clocks are left. FTA_s decides to follow the slowest clocks by appliying a very small local offset correction $FTA_s \approx 0$. After clock f has removed F smallest and F greatest measurements a group of slowest and one fastest clocks are left. Consequently, FTA_f decides to follow the average of the two by applying a local offset correction of $FTA_f \approx \frac{M_{f,s}}{2}$.
Now the distance between s and f can be greater than just FTA_f, because s and f "might have seen the world from different viewpoints". This means: The measurement errors for s and f can be maximally different such that the difference between s and f is enlarged. This value is ϵ_f as shown in (5), because the compensation in realtime is lower since clock f is faster.
Based on this worst case scenario we calculate U_f and U_s.

$$\begin{aligned} U_f &= T_f(2z + FTA_f) \\ &= T_f(2z) + \tau_f FTA_f \end{aligned}$$

It is assumed that the fastest clock p causes the worst case measurement error ϵ_{min} when measured by the fastest clock f, and that the slowest clock q causes the worst case measurement error ϵ_{max} when measured by the slowest clock s. To achive maximum distance after correction the minimum measurement error is chosen for FTA_f which will lead to the maximum Δ_2. With the definition of τ_i, γ_i and equations (6) the real time of the beginning of the new double cycle of f is:

$$U_f = T_f(2z) + \tau_f \frac{M_{f,p,min} + M_{f,s,min}}{2}$$

$$\begin{aligned} &= T_f(2z) + \tau_f \frac{\gamma_f \epsilon_{min} + \gamma_f (T_s(z) - T_f(z) + \epsilon_{min})}{2} \\ &= T_f(2z) + \frac{T_s(z) - T_f(z)}{2} + \epsilon_{min} \\ &= T_f(2z) + \frac{\frac{z}{1-r} - \frac{z}{1+r} + T_s(0) - T_f(0)}{2} + \epsilon_{min} \\ &= T_f(2z) + \frac{\tau_s z - \tau_f z + \Delta_0}{2} + \epsilon_{min} \end{aligned} \quad (7)$$

To achieve maximum Δ_2 clock q is chosen as input to the correction value calculation, with $M_{s,q,max} = \gamma_s \epsilon_{max}$. A further slowest clock with the same behaviour as q is named v.

$$\begin{aligned} U_s &= T_s(2z) + \tau_s FTA_s \\ &= T_s(2z) + \tau_s \frac{M_{s,v,max} + M_{s,q,max}}{2} \\ &= T_s(2z) + \tau_s \frac{\gamma_s 2\epsilon_{max}}{2} \\ &= T_s(2z) + \epsilon_{max} \end{aligned} \quad (8)$$

To achieve convergence the following condition $\Delta_2 = \alpha \Delta_0$ with $\alpha < 1$ has to be fullfilled. For this worst case scenario (8) and (7) are put together by the definition of Δ_2.

$$\begin{aligned} \Delta_2 &= U_s - U_f \\ &= T_s(2z) + \epsilon_{max} - T_f(2z) - \frac{\Delta_0 + \tau_s z - \tau_f z}{2} - \epsilon_{min} \\ &= T_s(2z) - T_f(2z) - \frac{(\tau_s - \tau_f)z}{2} - \frac{\Delta_0}{2} + \epsilon_{max} - \epsilon_{min} \\ &= (\tau_s - \tau_f)2z + \Delta_0 - \frac{(\tau_s - \tau_f)z}{2} - \frac{\Delta_0}{2} + \epsilon_{max} - \epsilon_{min} \\ &= (\tau_s - \tau_f)\frac{3z}{2} + \frac{\Delta_0}{2} + \epsilon_{max} - \epsilon_{min} \end{aligned} \quad (9)$$

Convergence can be achieved if $\Delta_2 < \Delta_0$.

$$\begin{aligned} (\tau_s - \tau_f)\frac{3z}{2} + \frac{\Delta_0}{2} + \epsilon_{max} - \epsilon_{min} &< \Delta_0 \\ (\tau_s - \tau_f)\frac{3z}{2} + \epsilon_{max} - \epsilon_{min} &< \frac{\Delta_0}{2} \\ (\tau_s - \tau_f)3z + 2(\epsilon_{max} - \epsilon_{min}) &< \Delta_0 \end{aligned} \quad (10)$$

Inequation (10) shows the quality of the correction depending on the span $\epsilon_{max} - \epsilon_{min}$ of the measurement error, the cycle length z and the drift. The expression $\tau_s - \tau_f$ is a function of the maximum drift r:

$$\begin{aligned} \tau_s - \tau_f &= \frac{1}{1-r} - \frac{1}{1+r} \\ &= \frac{2r}{1-r^2} \end{aligned}$$

Using this equation we can rewrite (10) as follows:

$$\Delta_0 \quad > \quad \frac{6rz}{1-r^2} + 2(\epsilon_{max} - \epsilon_{min}) \tag{11}$$

By Δ_1 the upper bound for the distance of the clocks is given.

$$\begin{aligned} \Delta_1 &= T_s(2z) - T_f(2z) \\ &= (\tau_s - \tau_f)2z + \Delta_0 \\ &> (\tau_s - \tau_f)5z + 2(\epsilon_{max} - \epsilon_{min}) \\ &= \frac{10rz}{1-r^2} + 2(\epsilon_{max} - \epsilon_{min}) \end{aligned} \tag{12}$$

4.5 Offset correction with sync slaves

By definition a fault free sync slave $a \notin \mathcal{S}$ operates with an inaccuracy r_a with $r_s \leq r_a \leq r_f$. In this case convergence can be achieved, if the condition (10) holds. But consider a clock s' which is the slowest sync master with respect to the actual situation. In contrast to s, clock s' needs not be maximally slowest with respect to the requirements to faultless nodes. Clocks f' and f have to be distinguished in an analoguous way. A problem may arise if sync slave a is itself slower than s' or faster than f'.
Now it is shown which conditions have to be fulfilled to synchronize a clock a slower than s'. The conditions for the opposite case ("a is faster than f") are similar.
A worst case scenario is built similar to the scenario in section 4.4. Clock a measures F values higher than $M_{a,s'}$. Since s is the slowest fault free sync master, there are no more than F measured values lower than $M_{a,s'}$. Consequently the highest and the lowest value in this scenario is $M_{a,s',max}$ and $M_{a,q',max} = M_{a,s',max}$, $q' \neq s'$, if another value is available. With choosing the slowest sync master for correction value calculation the absolute minimal possible correction is made for a clock slower than s'.
The clock a ends its clock time cycle after correction at real time U_a.

$$\begin{aligned} U_a &= T_a(2z) + \tau_a \frac{M_{a,s',max} + M_{a,q',max}}{2} \\ &= T_a(2z) + \tau_a M_{a,s',max} \\ &= \tau_a 2z + T_a(0) - \tau_a z - T_a(0) + T_{s'}(z) + \epsilon_{max} \\ &= \tau_a z + T_{s'}(z) + \epsilon_{max} \end{aligned}$$

Let $\Phi_0 = T_a(0) - T_f(0)$ be the initial distance between clocks a and f. To achive convergence $\Phi_2 = U_a - U_f < \Phi_0$ has to be satisfied.

$$U_a - U_f \quad =$$

$$\begin{aligned}
\tau_a z + T_{s'}(z) + \epsilon_{max} - T_f(2z) - \frac{\tau_s z - \tau_f z + \Delta_0}{2} - \epsilon_{min} &= \\
\tau_a z - \frac{2\tau_f 2z}{2} + T_{s'}(z) - T_f(0) - \frac{\tau_s z - \tau_f z + \Delta_0}{2} + \epsilon_{max} - \epsilon_{min} &= \\
\tau_a z - \frac{3}{2}\tau_f z - \frac{1}{2}\tau_s z - \frac{\Delta_0}{2} + T_{s'}(z) - T_f(0) + \epsilon_{max} - \epsilon_{min} &= \\
\tau_a z - \frac{3}{2}\tau_f z - \frac{1}{2}\tau_s z - \frac{T_s(0) + T_f(0)}{2} + T_{s'}(z) + \epsilon_{max} - \epsilon_{min} &< \Phi_0 \qquad (13)
\end{aligned}$$

Note that the most pessimistic situation can be achieved if a is the slowest faultfree clock ($r_a = -r$) and the lowest possible correction value applies. This situation occurs if $n_{s'} = n_s$. This makes $\tau_a = \tau_s = \tau_{s'}$ and $T_{s'} = T_s$. This simplifies the above equation:

$$\begin{aligned}
& \tau_a z - \frac{3}{2}\tau_f z - \frac{1}{2}\tau_s z - \frac{T_s(0) + T_f(0)}{2} + T_{s'}(z) + \epsilon_{max} - \epsilon_{min} \\
= \quad & \frac{1}{2}\tau_s z - \frac{3}{2}\tau_f z + \tau_s z + T_s(0) - \frac{T_s(0) + T_f(0)}{2} + \epsilon_{max} - \epsilon_{min} \\
= \quad & \frac{3z}{2}(\tau_s - \tau_f) + \frac{\Delta_0}{2} + \epsilon_{max} - \epsilon_{min}
\end{aligned}$$

As expected, this is exactly the same as (9). Consequently the same conclusions can be drawn as in (10) and (12).

4.6 Clock synchronization in a double channel system

In the investigation above it was not taken into account that there are two channels in the physical layer network. This comes into play by the fact that sync masters can completely ignore one channel, since always the minimum measurement value of the two channels is taken.
To show the impact on clock synchronization different cases are considered:

1. If both channels operate at nearly the same speed, no problem arises.

2. If the speed on one channels differs by a significant difference σ the clock synchronization runs on only one channel. In this case, sync slaves connected to the slower channel exclusively, expierence a difference of σ in the measurement errors. It is clear that σ has to be small enough to keep these sync slaves synchronized.

Although clock synchronization works perfectly, the problem in case 2 may arise. Note that a faulty slower channel can increase σ. A faulty faster channel can increase the delay up to σ which affects also the sync masters for the following reason: Some messages may be received from both channels. Then the replica from the faster channel is measured. Some other messages may be partially corrupted on the faster channel, such that some receivers measure the replica from the faster, others from the slower channel. In any case, σ directly influences the measurement error and thus must be kept sufficiently small - even in the presence of faults !

	r	Δ_0	Δ_1
1	50 ppm	60	84
2	100 ppm	96	144
3	250 ppm	204	324
4	500 ppm	384	624
5	1500 ppm	1104	1824

Table 2: Influence of the clock inaccuracy with fixed cycle length $z = 120000$ and $\epsilon_{min} = 0$, $\epsilon_{max} = 12$

	z	Δ_0	Δ_1
1	60000	60	84
2	120000	96	144
3	240000	168	264
4	480000	312	504

Table 3: Influence of the cycle length with fixed inaccuracy $r = 100$ppm and $\epsilon_{min} = 0$, $\epsilon_{max} = 12$

5 Offset Correction: Examples

The following tables show some example configurations, and the corresponding limits Δ_0 and Δ_1. All values are nominal microticks, which have a fixed length in nanoseconds, except r which is given in ppm. r represents the clock inaccuracy and z the duration of a cycle. Remember, that Δ_1 shows the state after a double cycle. The ϵ parameters represent properties of the current topology.

Table 2 shows the influence of the clock inaccuracy r on the guaranteed precision of the offset correction. The value for ϵ_{max} may represent a topology with three cascaded star couplers.

In table 3 different cycle lengths are assumed. These essentially consist of a static part with a fixed number of slots with the same length, a fixed sized dynamic part if present and a network idle time.

In table 4 a different topology is considered. For simplification it is assumed that a star coupler hop takes approximately four nominal microticks. Other sources of delays have to be compensated likewise. Case 2 shows the influence of overcompensation. In case 4 the compensation value was chosen too low.

	Number of star couplers	ϵ_{min}	ϵ_{max}	Δ_0	Δ_1
1	1	0	0	72	120
2	2	-2	6	88	136
3	3	0	8	88	136
4	3	2	12	92	140

Table 4: Influence of the topology with fixed inaccuracy $r = 100$ppm and cycle length $z = 120000$

6 Investigation of rate correction effects

6.1 Introducing rate correction into the existing model

This section discusses offset correction under the assumption of a limited gradient of the clock speed.
The FlexRay rate correction (see [Müller2001]) is an additional clock correction algorithm, using the same measurement, selection and correction calculation algorithm each cycle. In contrast to the offset correction, the measurement is done twice in a double cycle. The rate is calculated by taking the difference of both calculated values. To apply the rate correction, the following double cycle is enlarged or shortened by the result of the rate correction algorithm, i.e. it takes more or less microticks to finish a double cycle.
The rate correction part of the clock synchronization causes the deviations of clock speeds of all clocks to be bounded by a small constant at the beginning of a double cycle. In contrast to previous sections, the speed of the extremal clocks s and f is considered to be time dependant. The absolute maximum and minimum of all clocks are still given by γ_f and γ_s. If $\gamma_i(t)$ denotes the current speed of clock i at real time t, the following condition holds for every non-faulty clock:

$$1 - r = \gamma_s \leq \gamma_i(t) \leq \gamma_f = 1 + r \tag{14}$$

The worst case in this scenario is given by two clocks f and s with the maximum real time distance at the beginning of the current cycle and the maximum and minimum initial speed $\gamma_f(0)$ and $\gamma_s(0)$. The initial speed of both clocks is not necessarly γ_f or γ_s since they represent the result of the latest rate correction.
If $\gamma_\epsilon \ll 2r$ is the maximum difference in speed of all clocks after rate correction, the effect of the rate correction at the beginning of the double cycle at real time 0 can be expressed as follows.

$$\left|\gamma_s^0 - \gamma_f^0\right| \quad \leq \quad \gamma_\epsilon \tag{15}$$

With this definition γ_ϵ expresses the quality of the rate correction.
If the clocks could accelerate and slow down without limit the clock speed of the slowest and fastest clocks could gain their maximum/minimum values — in the worst case, immediately after cycle start. In this scenario all results of further scenarios apply without changes. In a realistic scenario the acceleration $\dot{\gamma}_i(t)$ of the clocks is bounded by a constant value ξ. In this case the following condition applies.

$$-\xi \quad \leq \quad \dot{\gamma}_i(t) \quad \leq \quad \xi \tag{16}$$

ξ is a property of faultfree oscillators rather than of the rate correction algorithm. The worst case scenario is now splitted into two phases. In the first phase the clocks needs some time to gain the overall maximum or minimum speed. In the second phase the clocks keep their current speed, because they cannot drift further by definition of a faultfree clock. This is illustrated in figure 1.
It is also possible that the double cycle ends before the second phase begins. Since the worst case scenario assumes maximum clock differences, the acceleration of the fastest clock is assumed to be maximum, while

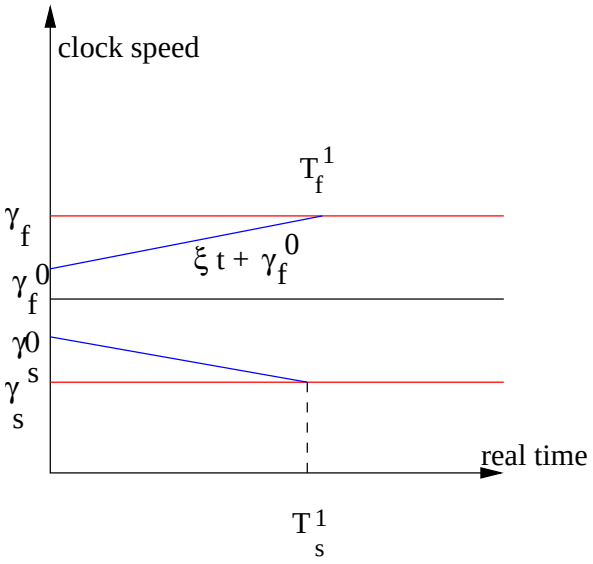


Figure 1: Minimum and maximum speed limits in a model with acceleration.

the acceleration of the slowest clock is assumed to be minimum.
This means that the fastest clock speeds up by ξt, while the slowest clock slows down by $-\xi t$ in the first phase. The current speed of the fastest and slowest clocks in the first phase can be expressed as follows.

$$\aleph_f(t) = \gamma_f^0 + \xi(t - T_f(0)) \quad (17)$$
$$\aleph_s(t) = \gamma_s^0 - \xi(t - T_s(0)) \quad (18)$$

At real time T_f^1 and T_s^1 the fastest and slowest clock are ending their first phase. This is the time when $\aleph_f(t)$ becomes γ_f and $\aleph_s(t)$ becomes γ_s.

$$\begin{aligned} \gamma_f &= \gamma_f^0 + \xi(T_f^1 - T_f(0)) \\ T_f^1 &= \frac{\gamma_f - \gamma_f^0}{\xi} + T_f(0) \quad (19) \\ \gamma_s &= \gamma_s^0 - \xi(T_s^1 - T_s(0)) \\ T_s^1 &= \frac{\gamma_s^0 - \gamma_s}{\xi} + T_s(0) \quad (20) \end{aligned}$$

Depending on acceleration the calculation of the local clock time changes. Equations (21) and (22) are showing the clock time of clocks f and s in phase one.

$$C_f^1(t) = \int_{T_f(0)}^{t} \aleph_f(t')\ dt'$$

$$= \gamma_f^0 \left(t - T_f(0)\right) + \xi \left(\frac{1}{2}\left(t - T_f(0)\right)^2 - T_f(0)\left(t - T_f(0)\right)\right) \quad (21)$$

$$\begin{aligned} C_s^1(t) &= \int_{T_s(0)}^{t} \aleph_s\left(t'\right) \; dt' \\ &= \gamma_s^0 \left(t - T_s(0)\right) - \xi \left(\frac{1}{2}\left(t - T_s(0)\right)^2 - T_s(0)\left(t - T_s(0)\right)\right) \end{aligned} \quad (22)$$

If $t > T_f^1$ the linear expression $\gamma_f \left(t - T_f^1\right)$ has to be added. Putting this together the local clock value for clock f at real time t is given by the following expression.

$$C_f(t) = \begin{cases} \gamma_f^0(t - T_f(0)) + \xi(\frac{1}{2}(t - T_f(0))^2 - T_f(0)(t - T_f(0))), & \text{if } t \le T_f^1; \\ \gamma_f^0(T_f^1 - T_f(0)) + \xi(\frac{1}{2}(T_f^1 - T_f(0))^2 - T_f(0)(T_f^1 - T_f(0))) + \gamma_f(t - T_f^1), & \text{if } t > T_f^1. \end{cases} \quad (23)$$

The same applies for clock s.

$$C_s(t) = \begin{cases} \gamma_s^0(t - T_s(0)) - \xi(\frac{1}{2}(t - T_s(0))^2 - T_s(0)(t - T_s(0)), & \text{if } t \le T_s^1; \\ \gamma_s^0(T_s^1 - T_s(0)) - \xi(\frac{1}{2}(T_s^1 - T_s(0))^2 - T_s(0)(T_s^1 - T_s(0))) + \gamma_s(t - T_s^1), & \text{if } t > T_s^1. \end{cases} \quad (24)$$

For further calculations we need the inverse function $T_i(c) = C_i^{-1}(c)$ which maps clock time c to real time t. For the first phase the equations 23 and 24 have to be resolved.
Two solutions exist per equation, but only one fulfill the requirement $T_i(c = 0) = T_i(0)$:

$$T_f^{1st}(c) = -\frac{1}{\xi}\left(\gamma_f^0 - 2T_f(0)\xi - \sqrt{(\gamma_f^0 - \xi T_f(0))^2 + 2\xi c}\right) \quad (25)$$

Now it is possible to express the real time at a particular clock time. The second phase starts at $C_f^1 = C_f(T_f^1)$.

$$T_i(c) = \begin{cases} T_i^{1st}(c), & \text{if } c \le C_i^1; \\ T_i^1 + \tau_i(c - C_i^1), & \text{if } c > C_i^1. \end{cases} \quad (26)$$

$$\begin{aligned} C_f(T_f^1) &= \gamma_f^0 \left(T_f^1 - T_f(0)\right) + \xi \left(\tfrac{1}{2}\left(T_f^1 - T_f(0)\right)^2 - T_f(0)\left(T_f^1 - T_f(0)\right)\right) \\ &= \gamma_f^0 \left(\tfrac{\gamma_f - \gamma_f^0}{\xi}\right)^2 + \xi \left(\left(\tfrac{\gamma_f - \gamma_f^0}{2\xi}\right)^2 - T_f(0)\left(\tfrac{\gamma_f - \gamma_f^0}{\xi}\right)\right) \\ &= \gamma_f^0 \left(\tfrac{\gamma_f - \gamma_f^0}{\xi}\right)^2 + \tfrac{(\gamma_f - \gamma_f^0)^2}{4\xi} - T_f(0)\left(\gamma_f - \gamma_f^0\right) \end{aligned} \quad (27)$$

The same applies for clock s.

6.2 Examples

The following scenarios should show, how much limited acceleration and rate correction influence the quality of clock synchronisation. The maximum distance Δ_1 was roughly estimated and bound by $T_s(2z) - T_f(2z)$.

The measurement errors ϵ_{max} and ϵ_{min} deviate slightly, by using $M_{i,j} = C_i(T_j(z) - \epsilon) - z$ for calculating measurement errors.
Table 5 in appendix C shows the parameters used in the example calculations.
Table 6 shows the results of the calculation of the staurization points T_f^1 and T_s^1.
The measurement scenario used in table 7 follows the calculation in section 4.2. Clock f measures a fastest and a slowest clock both with minimum error. Clock s measures two slowest clocks with maximum errors.
Table 7 shows the Δ_1 and Δ_2 values which are of final interest. The last column shows the Δ_1 with unlimited acceleration. The effect of limited gradient of clock speed can be seen by comparing Δ_1 with limited and unlimited acceleration.

7 Conclusion

The above investigation shows that the FlexRay clock synchronization method is able to provide sufficent convergence with up to F faulty clocks, provided conditions (10) and (13) are satisfied and σ is sufficiently small. With increasing convergence one of the conditions may be violated sooner or later. In case of violating condition (10) by letting Δ_0 getting too low, Δ_2 will be the upper bound for the distance of slowest and fastest sync masters in the next cycle (by definition). This process will proceed until condition (10) complies again.
All in all, it is expected that the distance will jump back and forth within these bounds. In any case, and at any time, the clock difference is bounded by Δ_1 as shown in (12). When limited gradients of clock speed are assumed the clocks are closer to each other (Δ_1 is smaller) as shown in table 7 (two rightmost columns).
Remark: Rate correction tends to benefit more from limited acceleration, because the drift of a clock during a cycle is limited. This keeps the overall distance between two clocks in certain boundaries.
Together with [Study A4-2] it can be concluded that FlexRay is able to tolerate benign byzantine faults within the known bounds, while the tolerance of malicious byzantine faults is limited. FlexRay provides sufficent convergence of faultfree sync masters and slaves even in the presence of up to F faulty nodes. However, the time difference σ between the two channels must be very small in any case. The question whether the current hardware solution guarentees a sufficently small σ is beyond the scope of this paper. The worst case σ in the presence of faults should be investigated carefully. Sync slaves connected to only one channel may lose synchronization if this very channel is faulty, of course.
With respect to the algorithmic issues of the FTA variant used here, the reader is referred to [Study A4-2].

References

[AncPua98] Anceaume, E., Puaut, I.: Performance Evaluation of Clock Synchronization Algorithms. Rapport de recherche No. 3526, INRIA, 1998

[Fuhr02] Fuhrmann, P., FlexRay Clock Tolerance Issues. Slides distributed Aug 5, 2002, as `FlexRay-ClockTolerancesIssue-(Philips)-05-08-2002.pdf`

[Kopetz] Kopetz, H., Hexel, R., Krüger, A., Millinger, D., Schedl, A.: A Synchronization Strategy for a TTP/C Controller, Technical University of Vienna

[Kopetz2001] Kopetz, H.: A Comparision of TTP/C and FlexRay. Research Report 10/2001, Institut für Technische Informatik, Technische Universität Wien, May 2001

[Lönn99] Lönn, H.: Synchronization and Communication Results in Safety-Critical Real-Time Systems. Department of Computer Engeneering, Chalmers University of Technology, 1999

[Lynch88] Welch, J. L., Lynch, N.A.: A New Fault-Tolerant Algorithm for Clock Synchronization. Information and Computation, Vol. 77, No. 1, April 1988. pages 1–36

[Lynch96] Lynch, N.A.: Distributed Algorithms, 1996

[Müller2001] Müller, B.: On FlexRay Clock Synchronisation, 2001

[PalGra92] Palumbo, D.L., Lynn Graham, R.: Experimental Validation of Clock Synchronization Algorithms, NASA Langley Research Center, June 1992

[PfSchwHe99] Pfeifer, H., Schwier, D., von Henke, F.W.: Formal Verifikation for Time-Triggered Clock Synchronization. Dependable Computing and Fault-Tolerant Systems, Vol. 12, pp. 207–226, IEEE Computer Society, January 1999

[PWD 0.6.0] Protocol Working Document Version 0.6.0, FlexRay Group, 2002

[ReqSpec2] FlexRay Requirements Specification Version 2.0.0, FlexRay Group, February 2002

[Rushby2001] Rushby, J.: Bus Architectures for Safety-Critical Embedded Systems, October 2001

[SchWeiss] Schossmaier, K., Weiss, B.: An Algorithm for Fault-Tolerant Clock State & Rate Synchronization, Technische Universität Wien, Department of Automation

[Study A4-2] Study A4-2: Clock Synchronization Decision Function Resilient to Benign Byzantine Faults; K. Echtle, FAN Project Group at University of Essen, 2002.

[Tel00] Tel, G.: Introduction to Distributed Algorithms, 2000.

A Startup

Startup issues are beyond the scope of this study. Nevertheless this Appendix contains some remarks concerning startup and the transition from startup to normal clock sync operation.

A.1 Reference to General Definitions

Non-faulty clocks are never faster or slower than the ideal clock $\pm r$. If this is the case, and the requirement of $|\mathcal{N}| \geq 3F + 1$ clocks with F faulty clocks is satisfied, clock correction is proofed to work correctly in a running network.
The startup procedure is responsible to adapt the rate of the clocks to meet these requirements.
A fault free clock can accelerate up to the maximum or minimum speed as shown in section 6. If the clocks' rate is already corrected to operate near the limit $1 \pm r$, the clock can accelerate, violating the requirements. In this model it was assumed that a clock can change its speed by $\pm \xi t$. A clock i may drift by $\pm\ \xi T_i(2z) \frac{\mu T}{\mu T_{nom}}$, during s double cycle. The absolute value of these clock speed drifts is bounded by $2r$, when a clocks drifts from slowest to fastest, or from fastest to slowest.[3] This is also the maximum distance between the speed γ_i^0 at the beginning of the double cycle and the maximum or minimum speed of a fault free clock. This requires that the corrected rate γ_i^{corr} is never greater than $\gamma_f - \xi T_i(2z)$ during runtime. In normal operation this is ensured by rate correction.

A.2 Basics of Clock Synchronization Startup

The FlexRay startup procedure introduces a clock in two different ways. The coldstart path is selected, if no other node is detected to be active. In this case no clock synchronization is needed. If there are active nodes in the network, the starting node selects the integration path. To synchronize the clock to the active nodes clocks the following steps are taken:[4]

1. The node listens, until a valid sync frame of an even cycle (the first cycle in a double cycle) is received.

2. The node waits until the next sync frame with the same id arrives. If this is not the case within a certain receive window, the node selects another sync frame and retries. The selection of this parameter is dealt with in [Fuhr02].

3. The node adopts the reference node's rate.

4. The rest of the cycle is skipped.

5. The node continues to listen on the bus without sending anything. Moreover normal clock synchronization (rate and offset correction) takes place. Normal operation starts if a plausibility check is passed.

[3] Such a behaviour for fault-free clocks would be hazardous for rate correction.
[4] Only the relevant details for this scenario are present. See [PWD 0.6.0] Chapter 8 for details.

Before a node can perform regular clock synchronization the second sync frame (with the same ID as the first) is expected within a receive window $z - W_{early} \leq C_r^a \leq z + W_{late}$, where C_r^a is the arrival time of the second sync frame (see step 2 above). For a fault free node r applies $C_r^a = \gamma_i^{hw}(T_r(z) + \delta_{r,i})$, where $\delta_{r,i}$ is the signal delay between the two nodes, and γ_i^{hw} the uncorrected native clock speed.[5] $W = \max\{W_{early}, W_{late}\}$ will be used as shorthand from now.
This means that the reference clock is only accepted, if

$$\left|\gamma_i^{hw} T_r(z) + \delta_{r,i} - \gamma_i^{hw} T_i^{hw}(z)\right| \leq W \tag{28}$$

Provided the arrival of the second sync frame falls into the receive window, the maximum applied rate change is W/γ_i^{hw}. If $\delta_{r,i} > 0$, the corrected rate of clock i is now slightly slower than clock r, so $\gamma_i^c = \gamma_r + \gamma_i^{hw}\delta_{r,i} = \gamma_i^{hw} + W/\gamma_i^{hw}$.
Then, the integrating node proceeds with steps 4 and 5 (see above).

A.3 Transition from Startup to Normal Operation

Many questions concerning startup of the clock synchronization algorithm in principle address the fact that startup is non-fault-tolerant while normal operation preserves fault tolerance over a long period. Where does initial fault tolerance come from?
Due to its non-redundant nature the FlexRay startup procedure can be fault-detecting at best. When the clock of the second integrating node follows the clock of the coldstarter node, any of the two may be faulty. Consequently, all three cases can happen:

- Both nodes agree on a common time which is acceptable for all faultfree nodes. In this case subsequent clock synchronization and fault-tolerant normal operation will work correctly.
- Both nodes do not agree on an acceptable time (either by disagreement or by agreeing on the wrong time). In a recent study, [Fuhr02] has defined a condition which avoids this case if the secondly integrating node is faultfree.
- Both nodes behave in a way that some faultfree nodes accept it as a common time base while other faultless nodes don't. Hence, only a subset of the nodes will complete startup successfully. This scenario is not avoided by the suggestion in [Fuhr02].

In the latter two cases a faulty node may mislead a faultless one. Further countermeasures have to be taken to prevent adverse effects of misled nodes.
Dynamic behaviour could add further problems. When nodes are in the process of integrating one after the other, then sudden faults could cause undesired transitions between the three cases. It seems that at least some of such transitions can be avoided by the condition defined in [Fuhr02]. An investigation in this direction can be helpful.

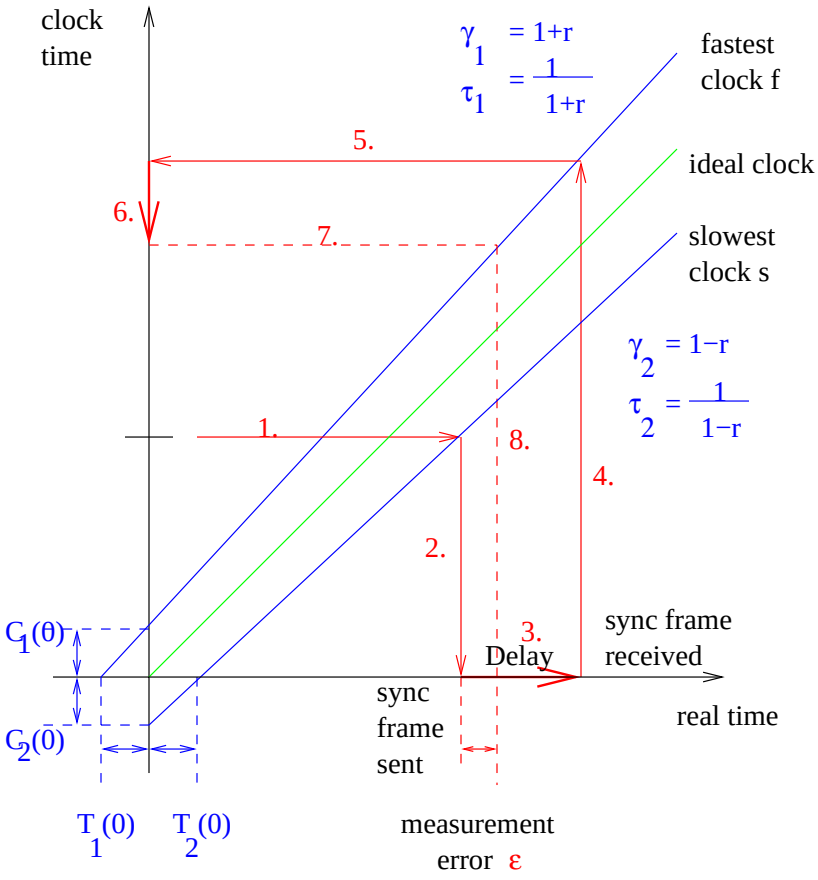
[5] FlexRay introduces a constant rate modification by default. The symbol γ_i^{hw} stands for the modified "hardware" speed, in [PWD 0.6.0] referred as $pMicroTickOverheadNom$.

One can also think of conditions other than [Fuhr02], which do not require that high oscillator quality. If one decides for an alternate approach, a deep investigation must follow as well.

The transition from startup (however it is performed) to normal operation of clock synchronization needs special care. The fault tolerance during normal operation requires redundancy (which is not present during startup). Consequently, the transition implies a membership problem, which can be solved by an agreement protocol. If normal operation requires, say n nodes plus s spare nodes, then all faultfree nodes must agree on the presence of all n + s nodes and their correct time behaviour. In this sense a membership-agreement protocol can provide the initial fault tolerance.

The three cases above become less critical if the protocol for the agreement on membership has the following property: If the protocol cannot be executed correctly due to wrong timing of a node, then the failure will be made obvious to all participating nodes. Even in this early phase protection of the nodes by their guardians is important. Otherwise a faulty node may exhibit arbitrary deviations in its behaviour at any time.

B Illustration



Fastest clock f measures s

1. Sync frame sent at clock time z.
2. Sync frame sent at real time T(z)
3. Delay $d_{1,2}$ on the channels.
4. Sync frame received at real time $T(z)+d_{1,2}$
5. Sync frame received at clock time $C(T(z)+d_{1,2})$
6. Delay compensated by o.
7. Frame assumed at clock time $C(T(z)+d_{1,2}) - o$
8. Frame assumed at real time $T(C(T(z)+d_{1,2}) - o)$

Figure 2: Clock s measures clock f.

C Results of Section 6.2

	Parameters										
	Physical		Network			Cycle Start		current Cycle			
	Inaccuracy	Acceleration	Cycle Length	Topology				initial Speed			
Symbol	r	ξ	z	ϵ_{min}	ϵ_{max}	$T_f(0)$	$T_s(0)$	γ_f^0	γ_s^0		
Unit	ppm	$\frac{\mu T}{\mu T_{nom}^2}$	μT	μT_{nom}	μT_{nom}	μT_{nom}	μT_{nom}	$\frac{\mu T}{\mu T_{nom}}$	$\frac{\mu T}{\mu T_{nom}}$		
1	1500	$1e-12$	120000	0	0	0	0	1.0000000	1.00000		
2	1500	$1e-12$	120000	0	12	−500	500	1.00000	1.00000		
3	1000	$1e-11$	120000	0	8	−500	500	1.00000	1.00000		
4	1000	$1e-11$	120000	0	8	−500	500	1.00001	0.99999		
5	1500	$1e-12$	120000	1	7	0	1100	1.00003	1.0000001		
6	500	$1e-12$	120000	1	7	0	1100	1.00003	1.0000001		
7	500	$1e-10$	120000	1	7	0	1100	1.00003	1.0000001		
8	1500	$1e-10$	120000	1	7	0	1100	1.00003	1.0000001		
9	500	$1e-5$	120000	0	8	0	1100	1.00000	1.00000		
10	500	$1e-7$	120000	0	8	0	1100	1.00005	0.99995		
11	500	$1e-10$	120000	0	8	0	1100	1.00025	0.99975		
12	1500	$1e-5$	120000	0	8	0	1100	1.00075	0.99925		
13	1500	$1e-6$	120000	0	8	0	1100	1.00075	0.99925		
14	1500	$1e-7$	120000	0	8	0	1100	1.00075	0.99925		
15	1500	$1e-8$	120000	0	8	0	1100	1.00075	0.99925		
16	1500	$1e-9$	120000	0	8	0	1100	1.00075	0.99925		
17	1500	$1e-10$	120000	0	8	0	1100	1.00075	0.99925		
18	1500	$1e-12$	120000	0	8	0	1100	1.00075	0.99925		
19	1500	$1e-8$	120000	0	8	0	1100	1.00000	1.00000		
20	1500	$1e-8$	120000	0	8	0	1100	1.00015	0.99985		
21	1500	$1e-8$	120000	0	8	0	1100	1.00075	0.99925		
22	1500	$1e-8$	120000	0	8	0	1100	1.0012	0.9988		
23	1500	$1e-5$	120000	0	8	0	1100	1.0012	0.9988		
24	1500	$1e-12$	120000	0	8	0	1100	1.0012	0.9988		
25	1500	$1e-12$	120000	0	8	0	1100	1.00015	0.99985		

Table 5: Parameters of example calculations. These scenarios include clock acceleration. At the time of this study realistic values for ξ have not been available. For this reason arbitrary values have been taken.

	Results I						
	Quality		Saturation		Saturation (Clock Time)		
	Rate Correction	Offset Correction	fastest Clock	slowest Clock	fastest Clock	slowest Clock
Symbol	γ_ϵ	Δ_0	T_f^1	T_s^1	C_f^1	C_s^1
Unit	$\frac{\mu T}{\mu T_{nom}}$	μT_{nom}	μT_{nom}	μT_{nom}	μT	μT
1	0	0	1500000000	1500000000	1501125000	1498875000
2	0	1000	1499999500	1500000500	1501125000	1498875000
3	0	1000	99999500	100000500	100050000	99950000
4	$2e-05$	1000	98999500	99000500	99049995	98950005
5	$2.99e-05$	1100	1470000000	1500101100	1471124550	1498975001
6	$2.99e-05$	1100	470000000	500101100	470124550	499975000
7	$2.99e-05$	1100	4700000	5002100	4701245	4999750
8	$2.99e-05$	1100	14700000	15002100	14711245	14989751
9	0	1100	50	1150	50	50
10	0.0001	1100	4500	5600	4501	4499
11	0.0005	1100	2500000	2501100	2500937	2499062
12	0.0015	1100	75	1175	75	75
13	0.0015	1100	750	1850	750	749
14	0.0015	1100	7500	8600	7508	7492
15	0.0015	1100	75000	76100	75084	74916
16	0.0015	1100	750000	751100	750843	749157
17	0.0015	1100	7500000	7501100	7508437	7491563
18	0.0015	1100	75000000	75000100	75084375	74915625
19	0	1100	150000	151100	150112	149889
20	0.0003	1100	135000	136100	135111	134890
21	0.0015	1100	75000	76100	75084	74916
22	0.0024	1100	30000	31100	30040	29959
23	0.0024	1100	30	1130	30	30
24	0.0024	1100	300000000	300001100	300405000	299595000
25	0.0003	1100	1350000000	1350001100	1351113750	1348886251

Table 6: Results of example calculations. This table shows the saturation points T_f^1 and T_s^1 of the clocks, where the clocks speed hits maximum. The line numbers refer to table 5.

	Results II									
	Measurement		Offset Correction			Distance	Limits			
	fastest Clock	slowest Clock	fastest Clock	slowest Clock			max. Distance		max. Distance with unlimited acceleration	
Symbol	$M_{f,p,min}$	$M_{s,q,max}$	U_f	U_s		Δ_2		Δ_1		Δ_1
Unit	μT	μT	μT_{nom}	μT_{nom}		μT_{nom}		μT_{nom}		μT_{nom}
1	0	0	239999	240000		0		0		720
2	1000	11	239999	240512		512		1000		1720
3	1000	7	239999	240508		508		1000		1480
4	1002	7	239998	240510		512		1005		1480
5	1104	6	240545	241107		561		1107		1820
6	1104	6	240545	241107		561		1107		1340
7	1106	6	240543	241109		566		1112		1340
8	1106	6	240543	241109		566		1112		1820
9	1220	7	240489	241227		737		1339		1340
10	1218	7	240489	241226		736		1337		1340
11	1161	7	240517	241170		653		1225		1340
12	1461	7	240370	241467		1097		1819		1820
13	1460	7	240370	241467		1097		1818		1820
14	1455	7	240370	241464		1094		1813		1820
15	1405	7	240370	241439		1069		1762		1820
16	1295	7	240438	241316		878		1517		1820
17	1282	7	240457	241290		833		1465		1820
18	1280	7	240460	241288		828		1460		1820
19	1244	7	240374	241354		980		1593		1820
20	1280	7	240370	241375		1005		1636		1820
21	1405	7	240370	241439		1069		1762		1820
22	1452	7	240370	241463		1093		1810		1820
23	1461	7	240370	241468		1097		1819		1820
24	1389	7	240406	241396		990		1676		1820
25	1136	7	240531	241144		612		1172		1820

Table 7: Results of example calculations. The final deviations between the clock can be seen by the values of Δ_1 (before offset correction) and Δ_2 (after offset correction). All deviations are much smaller than those in table 2 (where the gradients of clock speed are not limited). The line numbers refer to the line numbers in table 5.

D Summary of symbols

$\mathcal{N}$ Set of Nodes

n_i Node i; $n_i \in \mathcal{N}$

CH Function $\mathbf{CH} : \mathcal{N} \longmapsto \{A, B, AB\}$ from set of nodes to set of channel configurations. In configuration A or B the node is connected to only one channel, AB is a double channel configuration.

$\mathcal{S}$ Set of sync masters. $\mathcal{S} \subseteq \mathcal{N}$.

z Cycle length

s,q,v Worst case scenario slowest clocks. While s measures q, v it expierences different measurement errors.

f,p Worst case scenario fastest clocks.

$M_{i,j}$ Value clock i gets by measuring clock j, implicit through observating the frame arrival time on the bus. This value includes the error through various delays, and the error compensation. The value is interpreted as a local clocks' time difference.

F Maximum number of tolerated faults.

k Adapted parameter for FTA. Refer to table 1.

FTA_i The mid-value of the minimum and maximum measurement values, after deleting k minimum and maximum values in clock i.

t A clock value of a reference clock ("real time") with no inaccuracy.

c A clock time value of a real clock.

$C_i(t)$ Local clock time value of clock i at real time t.

$T_i(c)$ Real time at local clock time of clock i: $T_i(C_i(t)) = t$.

$\Delta_{i,j}(c)$ Real time distance between two clocks at local time c: $\Delta_{i,j}(c) = |T_i(c) - T_j(c)|$.

Δ_0 Initial real time distance at local clock time $c = 0$ in clocks s and f: $\Delta_0 = \Delta_{s,f}(0)$.

r_i Inaccuracy of clock i. For clock s the inaccuracy was chosen to be $r_s = -r$.

r Maximum inaccuracy of all nodes in the network.

$\mathcal{F}$ Set of faultfree nodes. $\mathcal{F} \subseteq \mathcal{N}$

γ_i Scaling factor for conversion of a real time distance to a local clock distance $\gamma_i = 1 + r_i$.

τ_i Scaling factor for conversion of a local clock time distance to a real time distance $\tau_i = \frac{1}{1+r_i}$.

$d_{i,j}$ Signal transfer delay between clocks i and j.

d Maximum signal transfer delay in the whole network.

o Measurement error compensation value

ϵ_i "Real" measurement error: The portion of the delay that was not compensated in node i.

$\epsilon_{min}, \epsilon_{max}$ Minimum and maximum measurement error.

Δ_1 Real time difference of clocks s, f after one double cycle: $\Delta_1 = \Delta_{s,f}(2z)$.

U_i Real time value of the new cycle start after correction.

Δ_2 Real time distance of clocks s, f after correction: $\Delta_2 = |U_s - U_f|$.

a The slowest sync slaves' clock.

s' The slowest sync master in the worst case scenario considering a sync slave to be the slowest clock. Need not to be maximally slowest clock.

f' The fastest sync master in the worst case scenario considering a sync slave to be the slowest clock. Need not to be maximally fastest clock.

Φ_0 Real time distance of a and f similar to Δ_0.

Φ_2 Real time distance of a and f similar to Δ_2.

σ Signal transmission speed difference between channel A and B.

$\gamma_i(t)$ Speed of clock i at real time t.

γ_i^0 Initial speed of clock i at real time $T_i(0)$.

γ_ϵ Difference between initial speed of clocks s and f.

$\dot{\gamma}_i(t)$ Acceleration of clock i at real time t.

ξ Upper bound of acceleration of all clocks.

T_i^1 Real time when the speed of clock i reaches maximum or minimum value (end of phase one).

$\aleph_i(t)$ Current speed of clock i in the first phase at real time t.

C_i^1 Clock time at end of phase one.

$C_i(t)$ Clock time at real time t. This function includes phases one and two.

$T_i^{1st}(c)$ Real time at clock time c in the first phase.

$T_i(c)$ Real time at clock time c. This function includes phases one and two.

γ_i^{corr} Speed of clock i immediatly after rate correction.

γ_i^{hw} Speed of clock i without any correction applied.

C_r^a Arrival time of the second sync frame clock r receives after entering the integration path during startup.

W_{early} Start of receive window relative to the cycle length.

W_{late} End of receive window relative to the cycle length.

W Maximum number of absolute values of W_{early} and W_{late}.