Simulation of Clock Synchronization in Multi-Cluster FlexRay Systems

Mohamed Soubhi, Klaus Echtle, University of Duisburg-Essen,
Institute for Computer Science and Business Information Systems,
45141 Essen, Germany, (msoubhi | echtle)@dc.uni-due.de

Abstract

This paper deals with the simulation of Clock Synchronization in Multi-Cluster systems that use FlexRay, a communication protocol designed by the FlexRay industry consortium in order to meet the fault-tolerance and dependability requirements of distributed real-time applications. Using SiMuFlex (**Si**mulation model for **Mu**lti-Cluster **Flex**Ray Systems), we investigate the performance of the clock synchronization algorithm and determine the amount of the system precision which is a measure for the quality of the clock synchronization. The simulation experiments are focused on the evaluation of the convergence and stability of the clock synchronization algorithm when the clock drift rates of several nodes undergo systematic variations. Moreover, we investigate the impact of the blackout and delays on the gateway component which interconnects the system clusters.
Keywords: Simulation, Clock Synchronization, FlexRay, Multi-Cluster, SiMuFlex.

1. Introduction

Distributed fault-tolerant real-time systems are deployed for various safety-critical applications in automotive ("x-by-wire") and aeronautic industry ("fly-by-wire") [1-3] as well as in railways, automation and process control. Time-triggered systems such as FlexRay [4] are becoming the technology of choice due to their deterministic behaviour. They enable predictable transmission of messages and a fault-tolerant global notion of time among all communication units termed nodes. Each node is equipped with a clock oscillator in order to meet its service requirements related to the timeliness. Keeping the local times of the nodes within a system synchronized even in the presence of arbitrary faults is a challenging task due, on the one hand to physical characteristics of the clock oscillators, on the other hand to varying message transmission delays (jitter)[5-6]. Thus, continuous clock synchronization is an indispensable primitive function of time-triggered systems. Hence, most of such systems consist of a single cluster i.e. a set of nodes that share a reliable communication medium and communicate over it in dedicated time intervals. In order to ease the complexity in system design, to reduce the development effort, to improve resource usage and to overwhelm bandwidth limitations, large real-time systems should be built-up into so called multi-cluster systems. The interconnection between these clusters is performed by a particular component termed gateway. Structuring the system in multi-clusters imposes additional efforts regarding inter-cluster communication and clock synchronization. For the FlexRay protocol we proposed in [7] a technique to establish and maintain a system-wide common time base that is directly derived from the clock synchronization algorithm used for single FlexRay clusters. This technique integrates global and local clock synchronization. The local synchronization means that the clock synchronization is performed within a cluster and the global is accomplished by sending/receiving additional time informations to/from other clusters. In this paper we investigate the performance of the clock synchronization in multi-cluster FlexRay based systems by means of simulation using SiMuFlex (**Si**mulation model for **Mu**lti-Cluster **Flex**Ray Systems). SiMuFlex provides simulation of various FlexRay services such as system startup, communication, clock synchronization and protocol error detection. It includes a fault-injection module which allows the simulation of node failures such as blackout (transient failures) as well as crash (permanent failures). Moreover, it permits to set up the transmission delays (from the sender to the receiver) which can be caused by various system components and manipulate them systematically (according to a specific function) or/and stochastically. The fault-injection module enables also to vary the system drift of each node in compliance with a specific function or stochastically. The system drift rate value of a node gives an insight about

whether this node is running fast or slow against the real-time.

2. FlexRay Protocol

FlexRay becomes the de-facto standard for the communication in the automotive domain. It allows the sharing of the bus among time-driven and event-driven messages in order to capture the benefits of both kinds of protocols. The time-triggered communication is carried out in the static segment und is characterised by its deterministic behaviour since the messages (termed frames) are delivered within an upper bound in time (TDMA). The event-driven communication is executed with the best-effort access strategy and is performed in the dynamic segment. Real-time is divided into cycles that consist of the mentioned static segment, dynamic segment, symbol window and a network idle time window (NIT). The static segment consists of a set of slots that could be assigned to nodes exclusively in order to transmit frames in each cycle.

2.1. Local Clock

Each node is equipped with a device for time measurement that contains a counter and a physical oscillator mechanism (e.g. quartz oscillator) that periodically generates an event to increase the counter [2]. The difference between the occurrences of two consecutive events (termed microticks) represents the granularity of the clock. The local clock often exhibits a varying drift rate that is affected by temperature variations, variation in intermediate devices, capacitive coupling, material imperfections, and aging of the crystal. The drift rate of the clock determines the deviation of this clock from reference clock assumed to be in perfect agreement with International Atomic Time TAI. The drift of the local clock comprises a systematic part and a stochastic part that is approximately 100 times smaller than the systematic one [9]. The oscillator manufacturer guarantees that the drift rate of the resonator does not exceed a specified bound (typically in the range of 10^{-2} to 10^{-7} sec/sec depending on the quality of the resonator) which is often specified in the data sheet of the resonator.

2.2. Global Time

In FlexRay based systems, all actions are triggered using a global notion of time (denoted macrotick) which is established among all system nodes. A macrotick consists of an integral number of microticks **(Figure 1)** that may vary from the nominal number as each node should synchronize its clock by adapting the number of microticks per macrotick according to the output of the clock synchronization algorithm. The clock synchronization of the nodes takes place in the NIT time window at the same point in time based on their local view of the global time.

Figure 1: Timing hierarchy in FlexRay

2.3. Clock Synchronization in FlexRay

Clock synchronization aims at bringing all nodes of the underlying system into timely agreement. It strives to maintain the clock skew within the cluster precision which enables all operations to occur in the logically correct order. The cluster precision is the upper limit for the time differences between the fastest and the slowest node within the cluster during the communication. The clock synchronization in time-triggered systems is a cyclic activity that must be performed by each node. The process of synchronization proceeds in three steps:

1. Remote clock reading: Each node derives clock values from a specific set of nodes using synchronization frames which are transmitted in dedicated slots. Each node in FlexRay uses a-priori known action points to transmit frames. The receiving node measures the time difference between the observed and the expected arrival time which is the sender's action point **(Figure 2)**.
The obtained values represent only an estimation of the remote because of jitters and clock drifts [7].

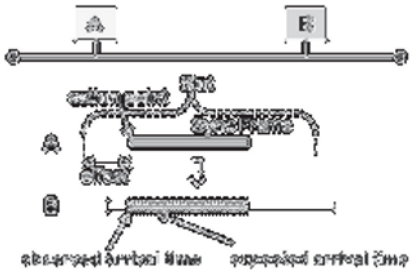

Figure 2 : Deviation measurement

Since the arbitration of the communication in the static segment is executed deterministically, sync frames could be sent only in the static slots. The measured deviation values (expressed in microticks) are stored in an appropriate data structure in order to calculate the adjustment values. Invalid values due to e.g. a reception that is outside the corresponding slots will not be stored.

2. Execute the clock synchronization algorithm: As previously mentioned the clock synchronization process calculates the correction terms relying on the remote

clock values. FlexRay uses a combination of clock state and clock rate correction **(Figure 3)**. The calculation task is executed by means of the FTM algorithm [8]. It proceeds by sorting the stored deviation values and discarding k-highest and k-lowest values where k is a configurable number that depends on the number of the stored values. The algorithm returns than the midpoint of the remaining ranges which serves as the state correction term [4]. This term indicates by how many microticks the node's communication cycle length should be adapted. The algorithm takes into account that faulty clocks may run either too slow or too fast and that correct clocks are in-between.
The rate correction term is calculated by comparing the corresponding measured time differences from two consecutive communication cycles. The FTM is applied to the collected values in order to calculate the clock rate correction term [4]. This term indicates by how many microticks the node's communication cycles should be likewise changed in the next double cycles.

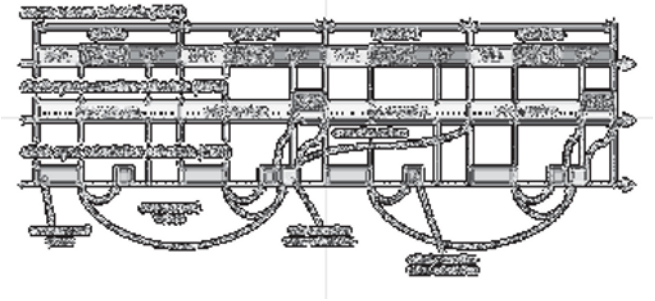

Figure 3: Execution of clock synchronization functions

3. Clock adjustment: The calculated correction value should be applied to the local clock in discrete or in continuous manner [4]. The clock state correction is accomplished during the NIT in each odd cycle discretely. Clock rate correction takes place during the entire communication cycle in a continuous manner.
The local clock of a node i after correction can be defined as follows:

$$Clk_i : R\Gamma \rightarrow C_{lk}\Gamma \ \wedge \ Clk_i(t) = H_i(t) + Adj_i \quad (1)$$

$R\Gamma$ is the set of real-time values and $C_{lk}\Gamma$ the set of the local clock values.
$H_i(t)$ corresponds to the hardware clock of node *i* which is generated by the oscillator at time *t*.
After synchronization, each two correct nodes i and j are clock-synchronized with precision π if the following property holds:

$$\forall t \in R\Gamma : \left| Clk_i(t) - Clk_j(t) \right| \leq \pi \quad (2)$$

3. Multi-Clusters

Multi-cluster structures permit the integration and the reuse of already known and best experienced single clusters. The interconnection between clusters is realized by a device called gateway. Moreover, multi-clusters provide performance benefits in terms of extensibility and Flexibility. We proposed in [7] a technique to establish and maintain a system-wide common time base that is directly derived from the clock synchronization algorithm used for single FlexRay clusters. This technique integrates global and local clock synchronization. The local synchronization means that the clock synchronization is performed within a cluster and the global is accomplished by sending/receiving additional time informations to/from other clusters.
The example shown in **Figure 4** consists of two clusters C_1 (with nodes A, B, C and D) and C_2 (with nodes E, F and G). Each node in the system adopts the same parameter's configuration so that the nominal cycle, slot and even the macrotick durations are identical. Both clusters are connected via the gateway denoted FlexWay. Depending on the schedule FlexWay should forward frames from one cluster towards the other.
These frames are called global frames. FlexWay forwards global frames immediately rather than buffering them. Frames that do not pass through the FlexWay are called local frames. The main benefit of this technique is that local communications can be performed synchronously (high bandwidth utilization) and global communication is dedicated for the cluster synchronization. For this purpose FlexWay provides switching services and has access to the global time.

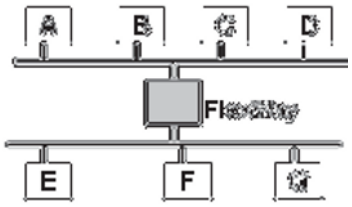

Figure 4: Example of a multi-cluster system.

Figure 5 depicts a simple schedule example for the multi-cluster system presented above (The letters in the boxes show the sender of the respective frame).

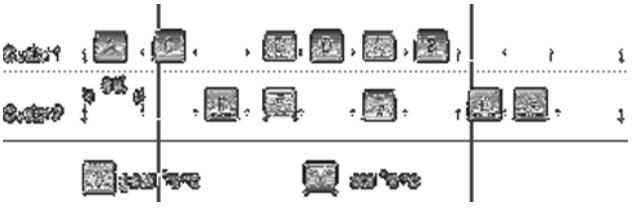

Figure 5: Schedule example

As shown in **Figure 5** node A sends a global frame in slot 1. The frame could be received by all nodes as the FlexWay forwards it towards C_2. Node D and E which share the same slot (slot 4) execute local communication each inside its cluster. In that case the FlexWay blocks the forwarding of both local frames towards the mutual cluster. Hence, this approach uses the FlexRay protocol specification without any change since the nodes handles each received frame as it comes from the same cluster.

3.1 Clock Synchronization

We distinguish between local and global synchronization:

- Local synchronization is to maintain a global time base among all nodes that belong to the same cluster. The maximum deviation between any two correct nodes over the entire operation time is called cluster precision. Local synchronization can be established and maintained if at least two sync local frames are transmitted periodically.

- Global synchronization is to maintain a global time base among all nodes of the systems. These nodes may belong to the same or to several clusters of the system. The maximum deviation between any two correct nodes of the system is called system precision. Global synchronization can be established and maintained if at least three sync frames are transmitted periodically [5]. In order to achieve a better system precision it is necessary to convey all sync frames throughout the FlexWay [7].

4. Simulation of Multi-Cluster FlexRay Systems

Simulation represents a powerful and an adequate way to investigate the functionality and structure of distributed systems. SiMuFlex has been developed for the simulation of FlexRay based systems that show a single bus topology as well as multi-cluster topology. It provides the simulation of FlexRay protocol services such as system startup, communication, clock synchronization and protocol error detection. It incorporates a failure module which is intended to inject nodes with faults. SiMuFlex is implemented in Java and comprises a simulation runtime module**(Figure 6)**, a graphical user interface to control the simulation runs and the appropriate output control **(Figure 7)**. The validation of SiMuFlex has been taken place on the base of reference tests on real systems using a VHDL model for the gateway. The results obtained shown that SiMuFlex follows the behaviour of the reference model.
The main objective of the simulation experiments is to investigate the amount of the system precision which represents the main performance measure for the clock synchronization algorithm. The amount of the system precision reflects the bandwidth loss of this system.

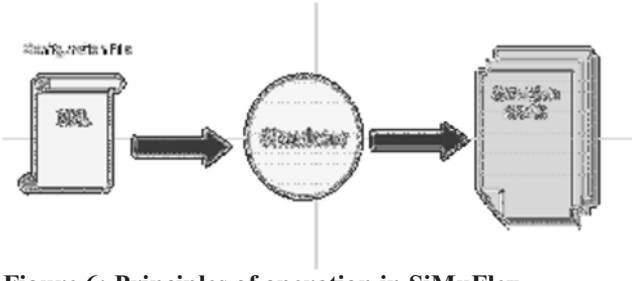

Figure 6: Principles of operation in SiMuFlex

Figure 7: Layout of SiMuFlex

4.1 Simulation Experiments

4.1.1. System model

In the simulation experiments, we use the maximum possible number of nodes (n = 15) in each cluster as specified in [4] (**Figure 8**). Hence, we presume the following configuration values for both clusters:

Macrotick duration	1µs
Microtick duration	0,05 µs
Microtick per macrotick	20
Communication cycle length	5000 MT
NIT	34 MT
Static slot	147 MT
Clock sync nodes (for each cluster)	0, 5, 14
Clock sync slots	40, 60, 70
Nominal frequency F_{nom}	20 MHz
Frequency tolerance $\Delta f(0)$	10 ppm (10^{-5})
Frequency stability $\Delta V_f(T)$	500 ppm (5×10^{-5})
Simulation time T	100 cycles
Drift of node i at time t $\rho_i(t)$	variable

The frequency tolerance $\Delta f(0)$ of a clock oscillator is the initial deviation frequency as compared to the absolute at 25 °C. It indicates the degree of the deviation from the nominal frequency F_{nom} at the simulation start time ($t=0$).
The frequency stability over the simulation time is defined as the frequency deviation compared to the measured frequency at the simulation startup time.
The drift rate ρ_i defines the frequency deviation of node i from the nominal frequency F_{nom} in parts per million (ppm).This means that the microtick durations in i are not strictly the same among the time. The node that runs with the maximum clock drift rate is the fastest node and vice versa. The drift rate of node i at a time t is the sum of the initial drift rate (at the simulation startup time) and the term that account for drift rate changes that must be

smaller than the frequency stability value which is the guaranteed upper limit for clock drift variation of an oscillator. To ensure that the drift rates of all nodes of each cluster are equally spread over the frequency tolerance interval $\Delta f(0)$, we adopt the following equation:

$$\rho_i(0) = \frac{(n-1-2i)}{2\times(n-1)} \times \Delta f(0) \quad (3)$$

This means that the initial clock drift rates are (**Figure 9**):

$$\rho_0 = +5\cdot10^{-6}, \rho_1 = +4,28\cdot10^{-6} ..... \rho_{14} = -5\cdot10^{-6}$$

Notational convention

- $t \in [0, T]$ denotes a point in simulation time.
- t_{cx} denotes the start of communication cycle *x*.
- t_{NIT_x} denotes the start of network idle time of communication cycle *x*.
- The drift rate ρ_i is expressed in second/second.

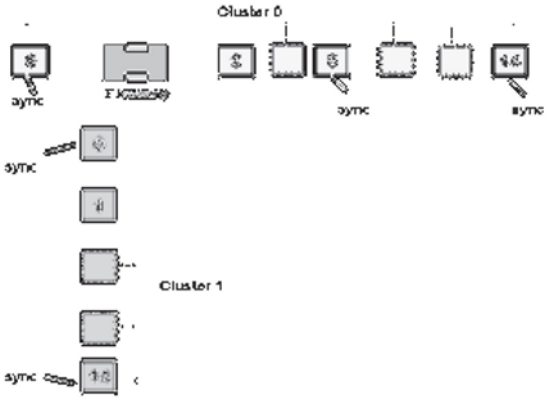

Figure 8: System structure

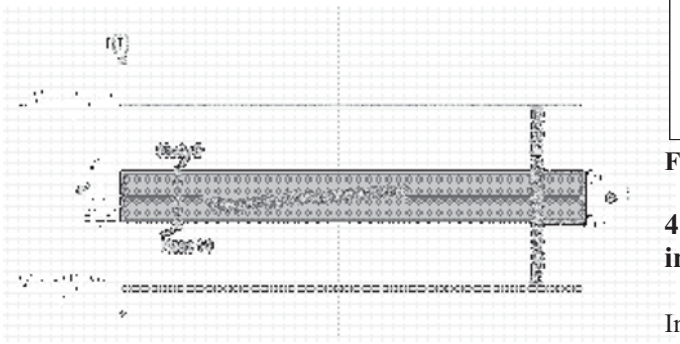

Figure 9: Tolerance intervals of the oscillators

4.1.2. Experiment 1: Invariable clock drift rates

In this experiment, we inquire about the amount the precision in case of invariable clock drift rates. The initial drift rates $\rho_i(0)$ that are given in equation (3) remain constant during the simulation time. **Figure 10**, **Figure 11** and **Figure 12** depict the achieved cluster and system precision. When the simulation starts, all nodes run free until the first clock state correction which is performed at the end of cycle 1. The first clock rate correction terms are applied in cycle 2 und cycle 3. Cluster 0 and cluster 1 achieve a precision of 6 microticks while the system precision attaints 10 microticks due to the delay caused by the gateway switching service. The switching delay varies randomly between 0 and a 0,125 microseconds.

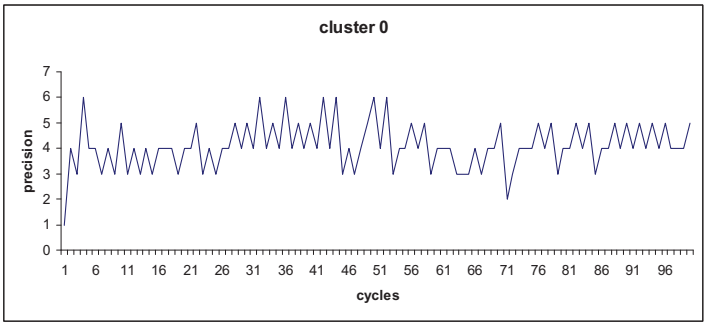

Figure 10: Experiment 1- Cluster 0 precision

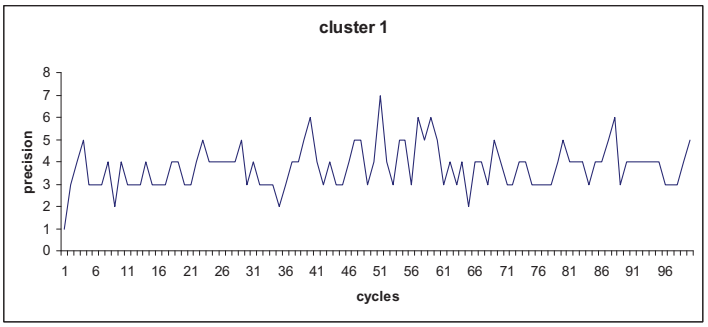

Figure 11: Experiment 1-Cluster 1 precision

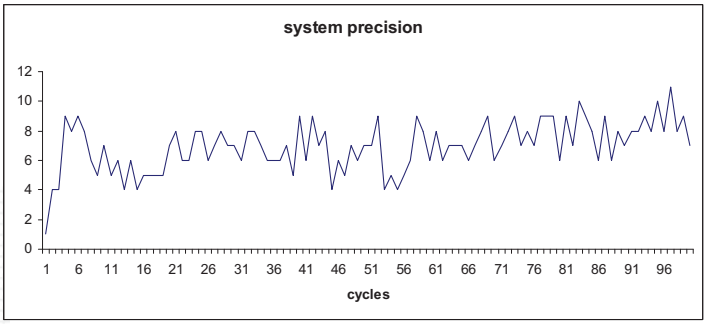

Figure 12: Experiment 1-System precision

4.1.3. Experiment 2: Clock drift rate changes immediately

In this experiment, we aim to scrutinize the convergence of the clock synchronization algorithm and to determine the system precision in case the fastest node in cluster 0 which is node 0 changes its clock drift rate immediately. Precisely, at time t_{c20} node 0 immediately changes its clock drift rate (**Figure 13**) and maintains it till the end of the simulation.

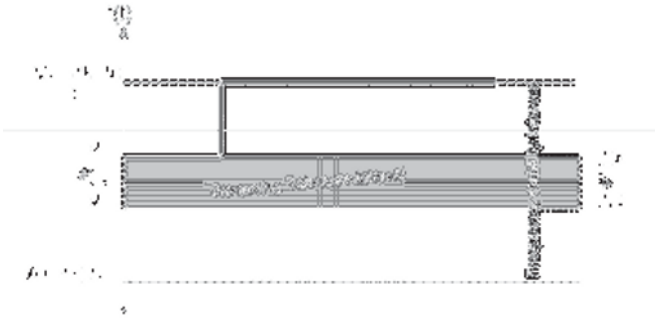

Figure 13: Experiment 2-Clock drift change function

$$\left.\begin{aligned} Vf_0(t) &= 0 \quad t < t_{c_{20}} \\ Vf_0(t) &= 2{,}5\times10^{-4} \quad t \geq t_{c_{20}} \end{aligned}\right\} \qquad (4)$$

Each node *i* in the system begins the simulation with clock drift rate $\rho_i(0)$. At the beginning of cycle 20 the clock drift rates $\rho_i(t)$ have been compensated by the clock rate correction algorithm. The clock state correction has been also accomplished in the NIT of cycle 19 (odd cycle). This means that the deviation of all clocks shows a local minimum. The next clock state correction is performed two cycles later during NIT of cycle 21. The drift rate change at node 0 of cluster 0 is indicated by the rising edge in the precision chart (**Figure 14**). The value of cluster 0 precision exceeds the duration of one macrotick immediately after the clock drift change. The clock state correction at the boundary of t_{c22} decreases the cluster 0 precision to 9 microticks. However, the change of the clock drift rate of node 0 (which sends sync frames) is completely compensated at the boundary of t_{c24}. As the clock drift rate at node 0 remains stable after cycle 20, the clock state correction in the NIT of cycle 23 brings the precision to a minimum. The course of the system precision undergoes a similar behaviour and achieves a value of 46 microticks. The clock synchronization algorithm compensates for the clock deviations during the reaming cycles as the clock drift rates are kept stable.

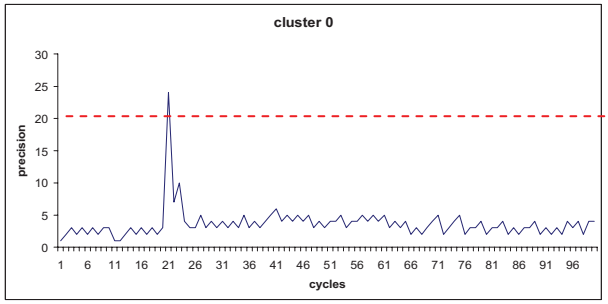

Figure 14: Experiment 2- Cluster 0 precision

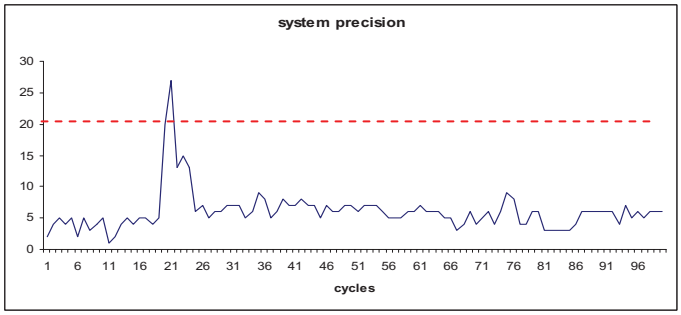

Figure 15: Experiment 2- System precision

4.1.4. Experiment 3: Clock drift rate changes linearly

In this experiment, we investigate the behaviour of the clock synchronization algorithm when the fastest node which is node 0 changes its clock drift systematically and according to a linear function during 10 cycles. As previously stated each node *i* starts the simulation with the clock drift $\rho_i(0)$ in compliance with equation (3). At t_{c20}, node 0 changes its clock drift rate from 5×10^{-6} to $2{,}55\times10^{-4}$ until t_{c30} according to equation (5).

Figure 16 elucidates the results for this experiment. It is obviously that the cluster 0 precision and the system precision do not exceed the duration of one macrotick although the clock drift rate achieve the maximum clock drift value $2{,}55\times10^{-4}$. This is because the clock drift rate at the fastest node changes slowly enough so that the clock synchronization compensate for the clock deviations.

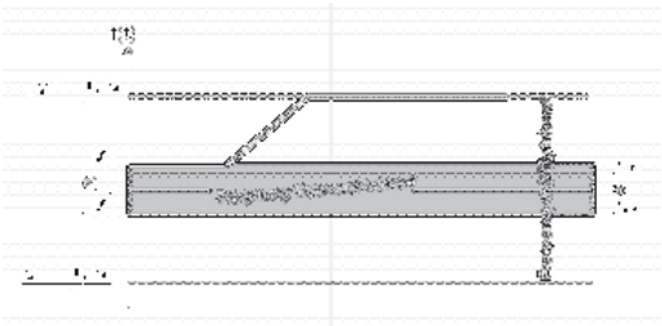

Figure 16: Experiment 3 - Clock drift change function

$$\left.\begin{aligned} Vf_0(t) &= 0 \quad t < t_{c_{20}} \\ Vf_0(t) &= 5\cdot10^{-3}\,\mathrm{t} \quad t_{c_{20}} \leq t < t_{c_{30}} \\ Vf_0(t) &= 2{,}5\times10^{-4} \quad t \geq t_{c_{30}} \end{aligned}\right\} \qquad (5)$$

The peaks pointed in the chart account for the clock state corrections during the clock drift change period. The clock rate correction algorithm ensures that the system precision stays bounded (about 17 microticks). From t_{c30} the clock drift rates of all nodes remain constant. Thus, the clock algorithm reaches a better system precision (about 8 microticks).

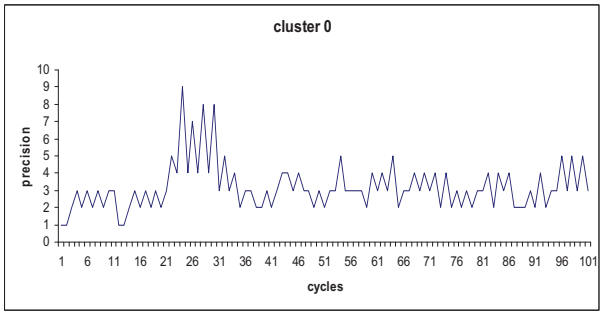

Figure 17: Experiment 3 - Cluster 0 precision

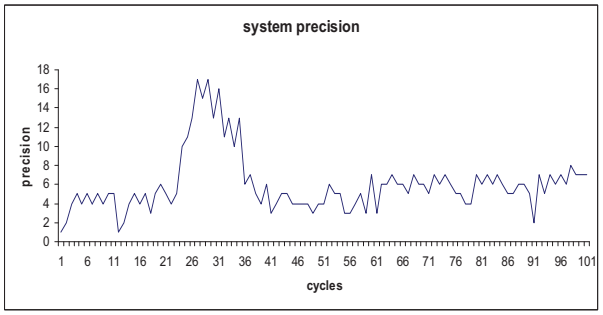

Figure 18: Experiment3 - System precision

4.1.5. Experiment 4:

This experiment is similar to experiment 3 with the difference that the change period of the clock drift rate of node 0 is reduced to 4 cycles instead of 10 cycles (equation (6)).

$$\left.\begin{aligned} Vf_0(t) &= 0 && t < t_{c_{20}} \\ Vf_0(t) &= 2\cdot 10^{-2}\,\mathrm{t} && t_{c_{20}} \le t < t_{c_{25}} \\ Vf_0(t) &= 2{,}5\times 10^{-4} && t \ge t_{c_{25}} \end{aligned}\right\} \quad (6)$$

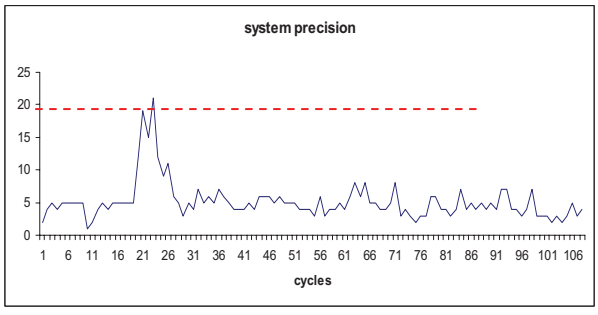

Figure 19: Experiment 4 - System precision

From experiment 3 and experiment 4 we conclude that the longer the period of the linear change of the clock drift rate the better is the compensation speed (convergence) of the clock synchronization algorithm.

4.1.6. Experiment 5:

This experiment, we considers the case when the clock drift rate of node 0 (the fastest node in cluster 0) and node 14 of cluster 1 (slowest node in cluster 1) change their clock drift rates immediately at t_{c20} (equation 4 and 7).

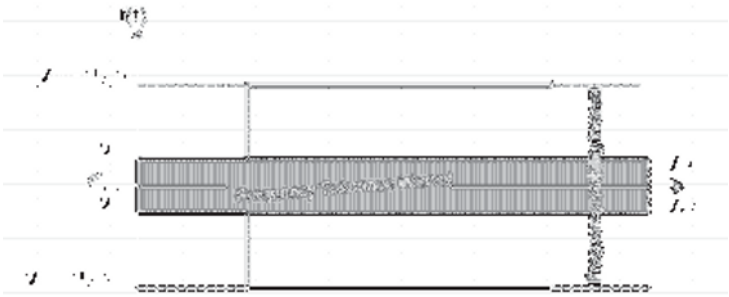

Figure 20: Experiment 5 - Clock drift rate change function

$$\left.\begin{aligned} Vf_{14}(t) &= 0 && t < t_{c_{20}} \\ Vf_{14}(t) &= -2{,}5\times 10^{-4} && t \ge t_{c_{20}} \end{aligned}\right\} \quad (7)$$

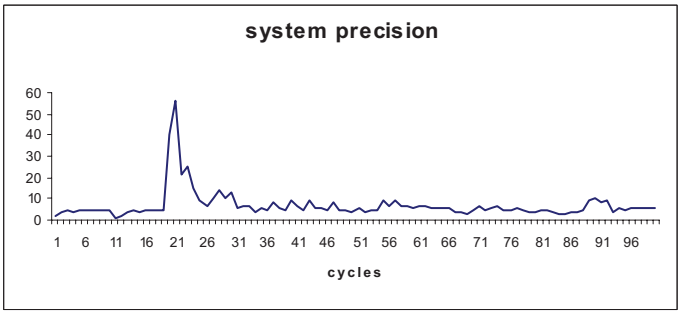

Figure 21: Experiment 5 - System precision

Figure 21 shows that the precision achieved in this experiment is deteriorated (about 56 microticks). This because the time deviation between the fastest node in cluster 0 and the slowest node in cluster drastically grows up during cycle 20 and 21. However, the state as well as the clock rate correction algorithm is able to compensate for this value after the clock drift change period.

4.1.7. Experiment 6:

In this experiment, we ascertain the system precision in case the fastest node in cluster 0 (node 0) and the slowest node in cluster 1 (node 14) change their clock drift rates linearly as outlined in **Figure 22** and equation 5 and 8. **Figure 23** points out the results obtained for this experiment. The system precision achieved is about 32 microticks and represents twice the system precision achieved in experiment 3.
After the period during which the clock drift rates change, the clock synchronization algorithm provides accurate clock readings which improve the system precision over the simulation time.

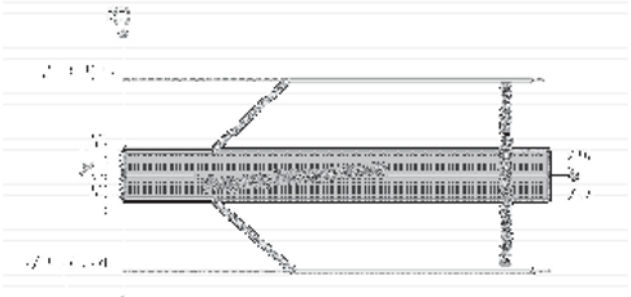

Figure 22: Experiment 6 - Clock drift rate change function

$$\left.\begin{array}{ll} Vf_{14}(t) = 0 & t < t_{c_{20}} \\ Vf_{14}(t) = -5\times10^{-3}\,\text{`t} & t_{c_{20}} \le t < t_{c_{30}} \\ Vf_{14}(t) = -2,5\times10^{-4} & t \ge t_{c_{30}} \end{array}\right\} \quad (8)$$

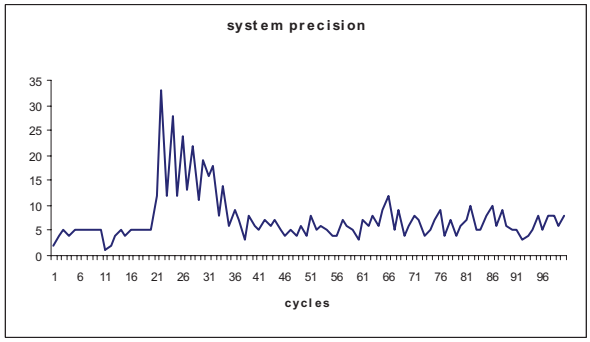

Figure 23: Experiment 6 – System precision

4.1.8. Experiment 7: Oscillating clock drift change at two nodes

In this experiment, we investigate the amount of the system precision in case the fastest node in cluster 0 and the slowest node in cluster 1 change their clock drift rates according to the functions described in equation (9) and equation (10)

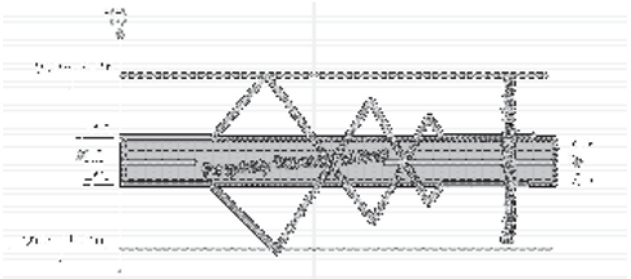

Figure 24: Experiment 7 – Clock drift rate change function

$$\left.\begin{array}{ll} Vf_0(t) = 0 & t < t_{c_{20}} \\ Vf_0(t) = 5\cdot10^{-3}\,\text{t} & t_{c_{20}} \le t < t_{c_{30}} \\ Vf_0(t) = -1,25\times10^{-3}\,\text{t} & t_{c_{30}} \le t < t_{c_{50}} \\ Vf_0(t) = 1,6\times10^{-3}\,\text{t} & t_{c_{50}} \le t < t_{c_{65}} \\ Vf_0(t) = -5\times10^{-5}\,\text{t} & t_{c_{65}} \le t < t_{c_{70}} \\ Vf_0(t) = 5\times10^{-6} & \text{t} \ge t_{c_{70}} \end{array}\right\} \quad (9)$$

$$\left.\begin{array}{ll} Vf_{14}(t) = 0 & t < t_{c_{20}} \\ Vf_{14}(t) = -5\cdot10^{-3}\,\text{t} & t_{c_{20}} \le t < t_{c_{30}} \\ Vf_{14}(t) = 1,25\times10^{-3}\,\text{t} & t_{c_{30}} \le t < t_{c_{50}} \\ Vf_{14}(t) = -1,6\times10^{-3}\,\text{t} & t_{c_{50}} \le t < t_{c_{65}} \\ Vf_0(t) = 5\times10^{-5}\,\text{t} & t_{c_{65}} \le t < t_{c_{70}} \\ Vf_{14}(t) = -5\times10^{-6}\,\text{t} & \text{t} \ge t_{c_{65}} \end{array}\right\} \quad (10)$$

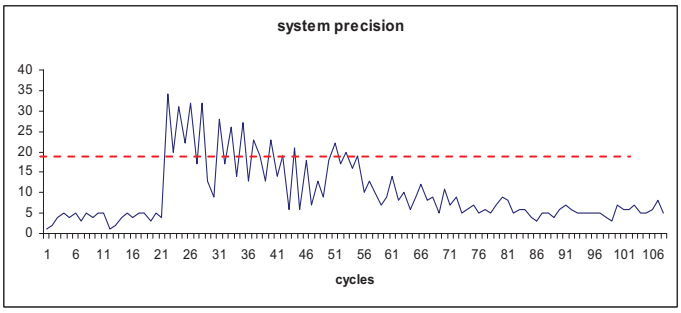

Figure 25: Experiment 7 – System precision

Figure 25 depicts the course of the precision achieved for this experiment. It is obviously that the precision increases drastically. The peaks correspond to the clock state corrections during each change period.
It is obviously, that the clock synchronization algorithm is able to compensate for the clock deviations as the clock drift rate remain constant. The clock rate correction part of the algorithm ensures that the precision does not exceeds 34 microticks and the clock state part.

4.1.9. Experiment 8: Impact of the delay on the gateway

In this experiment, FlexWay systematically delays from t_{c10} the forwarding of sync frames towards the other cluster until cycle 20 and remains constant until the end of the simulation. In the fact FlexWay does not own a communication controller which is able to synchronize its clock to the remaining clocks. FlexWay is connected via a specific interface such as SPI (serial peripheral interface) in order to retrieve timing informations. Precisely, the connected node sends a control signal which induces the blocking or the forwarding of the frames. This command is sent before the start of each slot. Depending on the characteristics of the used interface, the control signal may be delayed for any reason. Thus, it seems to be necessary to investigate the impact of the delay which could be cause by the FlexWay. The additional delay is estimated by up to 2 microticks in each cycle.

Figure 26 shows that the precision increases likewise systematically and that the clock synchronization algorithm is able to compensate for the clock deviations caused by the delays at the FlexWay.

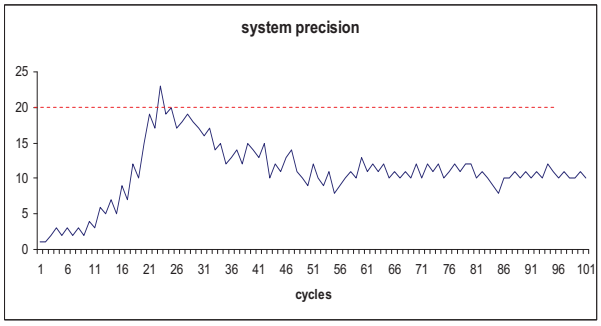

Figure 26: Experiment 8 – System precision

4.1.10. Simulation of the gateway blackout

In the following experiments (9a and 9b), we investigate the convergence behaviour of the clock synchronization algorithm as well as the amount of the precision in case FlexWay suffers a blackout failure. The blackout starts at t_{c10} and persists until t_{c14} for experiment 9a and until t_{c20} for experiment 9b.

Figure 27 and **Figure 28** show the results obtained for each experiment. The course of the system precision in both runs shows that the blackout leads to a considerable increase of the system precision. This because both clusters do not exchange sync frames and thus drift apart from each other. After the blackout period the clock state and clock rate correction algorithm of each node calculate the correction values (most falling edge) apply these values .Although the system precision decreases to a minimum values, the algorithm is de-stabilized for a time interval that depends on the period of the blackout.

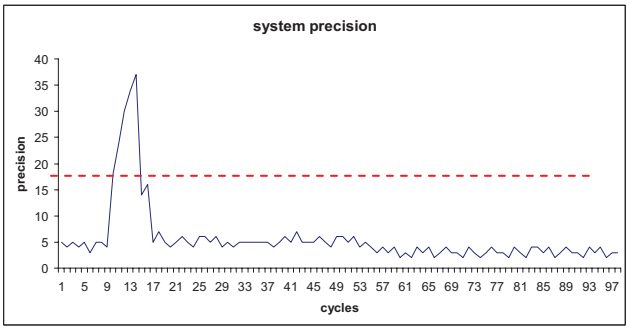

Figure 27: Experiment 9a – System precision

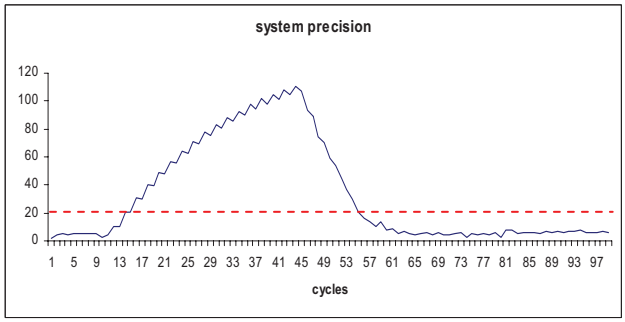

Figure 28: Experiment 9b – System precision

4.1.11. Impact of clock quality

In Experiments 10a and 10b, we deal with the impact of the oscillator quality on the system precision. As stated in previous sections, each oscillator is characterized by the frequency tolerance and frequency stability values which indicate how accurate it operates. In Experiment 10a, we aim to change the frequency stability of the fastest node in cluster 0 and the slowest node in cluster1 from initially 500 ppm to 1500 ppm (the maximum possible value). The new clock drift rates at the start of the simulation are set to $\rho_0(0) = 1,5\times10^{-3}$ and $\rho_{14}(0) = -1,5\times10^{-3}$ which corresponds to the frequency margin of each clock. The clock correction values remains stable during the whole simulation.

Figure 29 shows the results for Experiment 10a. After the start of the simulation the system precision deteriorates and achieves 22 microticks. As the clock drift rates stay stable the clock synchronization algorithm compensates for the deviation of the clocks (startup phase). However, the system precision remains remarkable higher than the achieved value in Experiment 1.

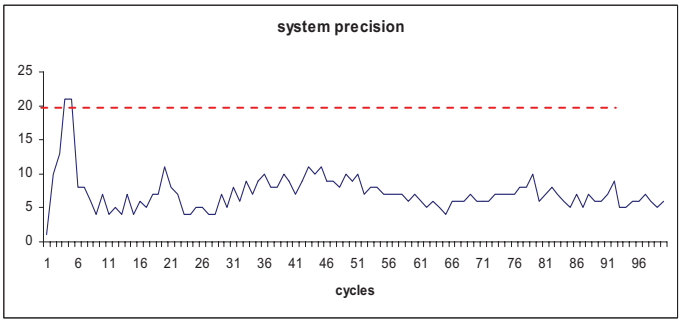

Figure 29: Experiment 10a – System precision

In Experiment 10b, we configure all nodes of both clusters that send sync nodes to run with high quality oscillator (low frequency stability value) and the remaining nodes with low oscillator frequency.

Figure 30 shows the results obtained for this Experiment. It is obviously that the system precision does not exceed 10 microticks.

$$\left.\begin{array}{ll} Vf_i(T) = 200\,ppm & i \text{ sends sync frames} \\ Vf_j(T) = 1,5\times10^{-3} & j \text{ does not send sync frames} \end{array}\right\} \quad (11)$$

From this Experiment and Experiment 10a we conclude that nodes that send sync frames should be equipped with high quality oscillators. For the rest of the node it is not necessary to operate with high quality oscillator in order to save costs.

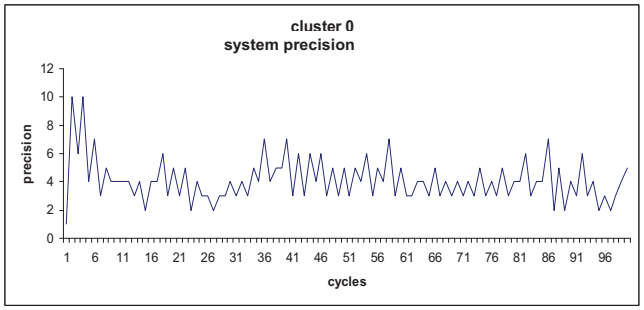

Figure 30: Experiment 10b – System precision

5. Conclusion

This paper deals with the clock synchronization in multi-cluster FlexRay based systems by means of simulation. A set of experiments are performed in order to asses the performance and stability of the clock synchronization in presence of clock drift rate changes of the key nodes (the fastest and the slowest node) in the multi-cluster system. The simulation experiments have shown that:

- The attained system precision is not necessary dependent on the clock drift rates of several nodes if these remain stable. This outcome holds even if the initial clock drift rates of several nodes operate with the maximum possible values (frequency tolerance and stability intervals). The clock synchronization algorithm (clock rate part) is able to damp the cluster drift and consequently to achieve a better system precision.

- In case the clock drift rates of the key nodes (fastest and the slowest nodes) change to the maximum permissible values, the system precision becomes worse in course of the change. The maximum system precision depends on the speed of the change. Immediate change of the clock drift rates leads to high system precision and to de-stabilization of the clock synchronization algorithm. When the clock drift rates change linearly the performance of the clock synchronization depends on the gradient of the change as shown in the experiments. However, the clock rate part of the clock synchronization algorithm compensates for the clock deviation of both clusters and achieves reasonable system precision values as the period of the clock drift variation is finished.

- The delay caused by the gateway component impacts the performance of the clock synchronization and leads to higher system precision which is self-evident.

- The system precision heavily deteriorates in case the gateway component suffers blackout. The effect of the deterioration depends on blackout period.

- It is not necessary to equip nodes that do not transmit sync frames, with high quality oscillator and thus costs could be saved.

6. References

[1] M. Baleani, A. Ferrari, L. Mangeruca, A. Sangio vanni-Vincentelli, Maurizio Peri, and Saverio Pezzini. Fault-tolerant platforms for automotive safety-critical applications. In CASES ’03: Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems, pages 170–177, 2003.

[2] Markus Krug, Hermann Kopetz, Elmar Dilger, Lars-Ake Johansson, U. Panizza, Peter Lidn, G. McCall , P. Mortara, Bernd Mller, Stefan Poledna, Anton Schedl, J. Sderberg, M. Strmberg, and Thomas Thurner. To wards an architecture for safety related fault tolerant systems in vehicles. ESREL ’97, June 1997, Portugal, Jun. 1997.

[3] Philip Koopman, editor. Critical Embedded Automo tive Networks, volume 22–4 of IEEE Micro. IEEE Computer Society, July/August 2002.

[4] FlexRay. FlexRay Communications System Protocol specification Version 2.1.Specification, FlexRay Consortium, 2005.

[5] Leslie Lamport, P. M. Melliar-Smith, “Byzantine Clock Synchronization”, Annual ACM Symposium on Principles of Distributed Computing, 1984

[6] XUE Fangxia , YAN Liaoliao , XIE Hong , YAO Yip ing ,LIU Haiye, “Study on technique of high time co herence used in real time distributed interactive simu lation”, Computer Applications, Vol. 26 No. 4. pp. 989-994, Apr. 2006.

[7] Klaus Echtle, Soubhi Mohamed, “Clock Synchroniza tion Issues in Multi-Cluster Time-Triggered Networks”, MMB & DFT 2010, 15-th international GI/ITG conference on “Measurement, Modelling and Evaluation of computing systems” and “Dependability and Fault Tolerance”, March 15-17 2010, Essen

[8] 12. J. Lundelius and N. Lynch. A new Fault-Tolerant Algorithm for Clock synchronization. In Proceedings of the 3th annual ACM symposium on Principles of Distributed Computing, 1984.

[9] W.Schabl. Der Einfluss zufälliger und systematischer Fehler auf die Uhrensynchronization in verteilten Echtzeitsystemen. PhD Thesis, Technische Universi tät Wien. Institut für Technische Informatik, Austria, 1988.