Interrupt based Switching for Multi-Cluster FlexRay Systems

Mohamed Soubhi
Dependability of Computing Systems
ICB, University Duisburg Essen
Essen, Germany
msoubhi@dc.uni-due.de
Dependability of Computing Systems
ICB, University Duisburg Essen
Essen, Germany
echtle@dc.uni-due.de

Abstract
This paper deals with a new straightforward approach to significantly increase the bandwidth usage in FlexRay networks. We propose a new hardware component, termed FlexWay, that facilitates partitioning the network into multiple clusters. This component provides a switching service that enables the execution of synchronous communications inside each cluster. Further, it is in compliance with the FlexRay protocol specification v2.1 as no change at the protocol layer is required. We place a great deal of emphasis on minimizing the inter-frame gap regarding the new topology and the proposed switching mechanism which is based on interrupts. Furthermore, we describe how FlexWay improves the fault-tolerance of the whole system by limiting the impact of babbling idiot and short circuit failures. Various experiments, carried out for an example system, show that all nodes keep their local clocks synchronized even in presence of a single failure and that the proposed approach perceptibly increases both the bandwidth and the safety.

1 Introduction
Accruing complexity in distributed real-time systems and the steadily increasing multitude of sensors, actuators and electronic control units, place higher demands on high-speed communication protocols. Moreover, systems that are targeted to operate in safety critical environments have to rely on protocols that provide a high degree of dependability and fault-tolerance. Because of their deterministic behavior and their integrated fault-tolerance measures, time-driven communication protocols are becoming the technology of choice for the design of safety-critical distributed systems. One of the most popular representatives of time-driven protocols is FlexRay[1]. It has proven its aptitude to deal with real time requirements of advanced automotive applications (x-by-wire) and to provide features that best suit to designer needs in terms of speed and compulsory safety. Providing requirements related to message delivery timeliness even in presence of arbitrary faults is not self-evident due, on the one hand to the physical characteristics of the node’s clocks, on the other hand to varying message transmission delays (latency and jitter) [2-3] which mainly depend on the used topology. Although FlexRay systems are able to afford up to 10 Mbps gross data rate per channel, only a fraction of the bandwidth can be effectively used due to overhead. In order to overwhelm bandwidth limitations, to enhance flexibility and to ease the integration of future functionalities, we propose a new device, termed FlexWay, that permits to organize a large system in Multi-Clusters, to interconnect them and to accomplish switching service without frame buffering. The bandwidth increase is achieved by allocating numerous switching slots in which communications are simultaneously executed within each cluster. The switching, that is the forwarding/blocking of messages according to a specific schedule, necessitates temporal strictness. Basically, the timely switching demands the implementation of the complex message header decoding and the commensurate synchronization paradigm. The proposed switching concept, however, overcomes this problem by using a control signal provided by one or more nodes via a point-to-point connection. This control signal notifies to FlexWay whether it has to forward or to block the next frame/s. The sending out of the control signal by the node which remote controls FlexWay is done within the associated interrupt service routine of the enabled timer interrupt. As a node may
act as the slowest respectively the fastest node in the network, the inter-frame gap should be configured to be large enough. Configuring large gaps, however, signifies a bandwidth loss. Thus, a particular trade-off must be scoped to the right inter-message gap that takes the interrupt latency, the propagation delay of the point-to-point connection, the propagation delay of FlexRay messages and the clock variation of each node into consideration. Furthermore, we describe how FlexRay increases the safety by limiting the impact of the babbling idiots to the cluster to which the faulty node or channel belongs.

This paper is organized as follows. Section 2 gives a brief overview of FlexRay and its key features such as clock synchronization. Section 3 states the potential utilities of Multi-clusters in FlexRay networks by means of an example. In section 4, the proposed switching concept is thoroughly described. Section 5 deals with the fault hypothesis with respect to the new architecture. Section 6 is concerned with the timing analysis of the FlexRay network as well as of the switching method. Section 7 outlines the conducted experiments and the obtained results. This paper ends with section 8 which provides a conclusion.

2 The Time-Triggered FlexRay Protocol

FlexRay is a state-of-the-art communication protocol for next-generation vehicles that provides flexibility and determinism. Flexibility is achieved by combining a scalable static and dynamic message transmission capturing, thus, the merits of synchronous and asynchronous protocols. The static part of a communication cycle is based on TDMA (Time Division Multiple Access) which is suitable for safety critical applications while the dynamic part relies on FTDMA (Flexible Time Multiple Division Access). A cycle contains also the idle phase NIT during which clock synchronization is executed. The static part, called static segment, is subdivided into slots that exhibit the same magnitude. The allocation of these slots for transmission and reception is accomplished offline for each node. FlexRay supports the concept of scalable fault-tolerance so that the system under design may be configured with a replication channel and local or central bus guardian [6]. Moreover, the used fault-tolerant clock synchronization algorithm [7] considers the transient/permanent as well as symmetric/asymmetric faults. The clock synchronization in FlexRay is a cyclic activity that aims at bringing all connected nodes into timely agreement so that they communicate in correct order. The key issue of any clock synchronization is to provide a smaller precision value.

2.1 Time Representation in FlexRay

The time domain inside FlexRay nodes is structured in four timing levels as depicted in Figure 1: The lower level consists of microticks which corresponds to the ticks of the node’s clock oscillator and represents the infinitesimal time granularity in FlexRay. The second level is formed by macroticks that comprise a defined integral number of microticks. Each action performed by the communication controller of a node is triggered based on its local view of the global time (macrotick counter). The next level is the slot level and the last one is represented by the cycle which corresponds to the common round in a time-triggered protocol.

Figure 1: Timing levels in FlexRay

2.2 Clock Synchronization

Each node in FlexRay maintains a clock oscillator that periodically generates microticks. The occurrence time of two consecutive microticks may vary due to the drift rate that represents the deviation of this clock from a reference clock assumed to be in perfect agreement with the International Atomic Time TAI. It comprises a systematic part and a stochastic part that is approximately 100 times smaller than the systematic one [9]. The drift rate is often affected by temperature variations, variation in intermediate devices, capacitive coupling, material imperfections, and aging of the crystal. Each node has a macrotick counter that keeps the current macrotick and a microtick counter for each macrotick. This latter may
be subject to modification in order to adjust the local view of the global time. Before adjusting this time, each node proceeds as follows:

- **Measurement of the clock deviations:** the transmission of a frame in FlexRay is performed at the beginning of an a priori defined offset time since the slot start (termed action point offset) as depicted in Figure 2. Each node derives clock deviation values based on special messages, termed sync frames, by calculating the time difference between the observed and the expected arrival time in microticks granularity. The expected arrival time at the receiver corresponds to the action point. The obtained values are stored in a convenient data structure known as deviation table.

![Figure 2: Measurement of the clock deviation between node A and B.](image)

- **Execute the clock synchronization algorithm:** the calculation of the correction term for the offset (phase difference) and rate (frequency difference) is performed using the FTM algorithm[7] in each odd cycle. It first sorts the stored values in ascending order and discards the k smallest and the k largest values. Then, it builds the average on the resulting range bounds. This average indicates by how many microticks the network idle time NIT should be shortened or lengthened in order to be in agreement with the rest of the network nodes. For the calculation of the rate correction term, the algorithm uses the time deviation of the same sync frames in two consecutive cycles.

- **Application of the clock correction terms:** FlexRay uses a combination of discrete clock adjustment (offset correction) and continuous clock adjustment (rate correction). The calculated offset correction value is applied in the same odd cycle while rate correction is spread over the consecutive two cycles (odd and even).

3 System Model

3.1 Single Cluster

FlexRay supports bus, star as well as hybrid topologies. A hybrid topology incorporates a bus and up to two active stars [1]. In order to elucidate the bandwidth limits of standard topologies, we introduce the example shown in Figure 3. It displays a single cluster that is built as bus connecting six nodes. We assume that the communication cycle has a length of 5 ms since many process control feedback loops in the automotive field are carried out within this time period. Further, we assume that the nodes communicate with the maximum payload[1] according to the schedule depicted in Table 1:

![Figure 3: FlexRay Bus topology example.](image)

<table>
<thead>
<tr>
<th>slot</th>
<th>sender</th>
<th>receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>A</td>
<td>all</td>
</tr>
<tr>
<td>3</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>4</td>
<td>C</td>
<td>all</td>
</tr>
<tr>
<td>5</td>
<td>B</td>
<td>A, E, F</td>
</tr>
<tr>
<td>6</td>
<td>C</td>
<td>A, B, D</td>
</tr>
<tr>
<td>7</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>8</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>9</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>10</td>
<td>D</td>
<td>all</td>
</tr>
<tr>
<td>11</td>
<td>C</td>
<td>D</td>
</tr>
<tr>
<td>12</td>
<td>D</td>
<td>C</td>
</tr>
<tr>
<td>13</td>
<td>E</td>
<td>C, D, F</td>
</tr>
<tr>
<td>14</td>
<td>F</td>
<td>E</td>
</tr>
<tr>
<td>15</td>
<td>E</td>
<td>F</td>
</tr>
</tbody>
</table>

Table 1: Communication schedule example

As stated in table 1 node A is assigned three slots. The first frames is sent in slot 1 and should be consumed by node B. The second frame is a

---

1The obtained values represent only an estimation of the remote clocks because of jitter and clock drifts [8].

2127 two-byte-words is the maximum payload in FlexRay
broadcast frame (all) and may be configured as sync frame. The last frame is transmitted in slot 7 and should be consumed once again by node B (self-explanatory for the remaining nodes). The resulting bus utilization is shown in Figure 4. This latter may show also the maximum number of frames that can be effectively exchanged during the cycle. The achieved effective bandwidth for this example is about 5 Mbps which is half the gross data rate. Generally, the number of slots and thus of frames decreases pursuant to the increase of the used maximum payload. Figure 5 outlines the relation between the number of slots and the associated effective bandwidth according to the used maximum payload for this example. The course denoted by min respectively max corresponds to a configuration with the minimum respectively maximum parameter values of FlexRay[1] as given in Table 2. moreover, the bandwidth courses do not take into account that neither a dynamic segment nor the symbol window are used and that some nodes will not necessarily allocate the full payload and thus not best exploit the frame length what additionally affects the net bandwidth.

<table>
<thead>
<tr>
<th>TSS [bits]</th>
<th>Action point offset [µs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>6</td>
</tr>
<tr>
<td>typical</td>
<td>7</td>
</tr>
<tr>
<td>max</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>63</td>
</tr>
</tbody>
</table>

Table 2: Parameter values

### 3.2 Multi-Clusters

The system presented in previous section is know structured in three clusters that are connected via FlexWay device as depicted in Figure 6. Each cluster contains two nodes that share one or two half-duplex channels.

In order to perform the communication described in Table 1, FlexWay blocks (switch off) and forwards (switch on) frames according to Figure 7. For instance, frame A2, D2 and C1 are broadcast frames and should be, thus, conveyed towards all other clusters that differ from the source. Frame B2, C2 and E1 should be forwarded to only one cluster (multicast) while the remaining frames are inhibited to leave their source clusters. We distinguish between local communication, that is when frames are exchanged within the same cluster, and between global communication (broadcast

---

3This result is obtained for a cycle configuration that consists of only a static segment and 100 MT NIT [1]. The static segment comprises of the longest possible slots (395 MT) for 10 Mbps data rate.
and multicast). In order to achieve a greater bandwidth exploitation, local frames that belong to different clusters share the same slots (i.e., A1, D1, E2). Further, slots which are dedicated to multicast communication can be also shared (i.e., D3 and B2). Since local communications are carried out simultaneously, only nine instead of fifteen slots are needed. This corresponds to a performance increase of 40% in terms of bandwidth. In other words, sechs slots are saved and slot 4 could be still used for local communication within cluster 2 or cluster 3 or for an inter-cluster communication between them. In many other examples, up to 70% slots can be economized depending on the desired schedule. Another benefit when using this kind of topology is that FlexWay can be deployed to implement bus guardian functionalities including the reshaping of singles[6] and to interface other communication systems such as CAN and LIN. Moreover, it permits also the integration and the re-use of already known and best experienced single clusters providing more flexibility and extensibility.

4 Switching Service

As explained in the introduced example, FlexWay should provide a switching mechanism to enable global and local communication. In contrast to many switches of various communication protocols, FlexWay can not execute frame ID based switching due to the timeliness requirements and to the constraints of the FlexRay physical layer[10]. More precisely, the switching decision based on frame ID requires the decoding of the significant part of the frame header including the ID and the CRC [1]. The decoding process should also take the transmission start sequence time that precedes any frame into consideration. In other words, FlexWay will delay a frame by at least 4000 ns (bit time is 100 ns for 10 Mbps data rate) which is not in compliance with the FlexRay specification and yields a considerable bandwidth loss. For this purpose, FlexWay should make the decision before the occurrence of the action point offset which implies that it has to have access to the global time base of the FlexRay network. The access to the global time base can be accomplished via the following methods:

**Method 1:** FlexWay performs clock synchronization to get into timely agreement with the FlexRay nodes. This requires the implementation of the full clock synchronization process as specified in [1]. In other words, FlexWay should generate microticks, macroticks, measure the time deviations, store them and perform the FTM algorithm. Further, it has to implement a control unit that enables the switching from global to local communication and vice-versa. It has to provide enough memory to store the schedule and possibly to build-up the full FlexRay IP-core. The switching should be done at the start of the slot in which the communication should alternate from global to local or vice-versa.

**Method 2:** FlexWay relies on timing informations of a FlexRay node. As each node executes the clock synchronization paradigm and has access to its local macrotick counter, it is sufficient to connect FlexWay to a specific node via an extra connection and to align the switching task to the FlexRay time base of this node. However, the time needed to provide these informations (slot and macrotick counter values) may be considerably high so that the macrotick value will not be valid when it is received. This is because the FlexRay time progresses independently. On account of this, FlexWay should add

---

4 The allowed maximum propagation delay for active stars is 250 ns.
5 The frame ID of a FlexRay frame appears 16 bits and the CRC 34 bits after the begin of its header.
a fix offset to the received macrotick value which corresponds to the transmission and the decoding delay of the timing signal. Due to jitter, the added offset causes a high imprecision that should be taken into account.

**Methode 3**: This method is similar to Method 2 with the difference, that no timing informations are needed. At the start of the slots in which a switching from local to global communication or vice-versa should be performed, the host of the node which is connected to FlexWay sends them an eight bit control signal that issues the switch on respectively the switch off command when 1 respectively 0 is received. In order to handle the problem of noise and glitch, the controlling node sends 0xFF instead of 1 in order to apply the majority voting decision. This technique requires neither clock synchronization of FlexWay nor the retention of the schedule.

In this paper, two related approaches of controlled switching according to method 3 are addressed. These are explained on the introduced example.

**Approach 1**: As pictured in Figure 8, node B is connected to FlexWay via the serial peripheral interface (SPI). At the start of each cycle, the FlexRay communication controller of node B generates an interrupt request to the host. The host executes the interrupt service routine ISR1 and start an absolute timer that elapses when the last macrotick of slot 1 occurs. The communication controller generates then an interrupt request due to the timeout condition. The host executes the interrupt service routine ISR2, restarts the timer that will expire at the end of slot 2 and transmits the control signal via the SPI-connection with 0xFF as content. Upon the receipt of this signal, FlexWay switches on enabling the forwarding of the global frame A2 that can be sensed at the earliest at the action point offset of node A (sender). When the second ISR2 is executed by the host of B, a control signal with the content 0x00 is transmitted and the control register of the used timer is assigned a new value that corresponds to the end of slot 4. For the remaining of the slots, node B drives FlexWay in the same manner (see Figure 10).

**Approach 2**: This approach differs from approach 1 slightly. At the start of each slot, node B transmits a 16 bit control signal that contains its slot counter value (corresponds to the 11 bit long frame ID) and a parity bit that covers this value. The remaining MSB are not considered. When this signal is received, FlexWay increments its internal slot counter and checks whether the counter value matches the received value. Otherwise, the internal counter adopts the new value and increments a specific error counter indicating that node B has behaved faulty. This kind of faults can be avoided if the mentioned signal is replicated by another node via an additional SPI connection or by the use of a second FlexWay device that should be connected to the second FlexRay channel and controlled by another node as displayed in Figure 9. FlexWay checks then whether or not the switching should be performed. This is done by reading the corresponding bit of a register which is mapped to the received slot value. If the corresponding bit is set and the previous slot was dedicated to local communication, it switches on. The same holds in the case the mapped bit is not set.

Through this approach, the forwarding of frames sent by faulty nodes during a global communication phase may be inhibited. This is because FlexWay is aware from which port (each cluster is mapped to a port) a global frame is expected, if any is scheduled for the slot on question. For instance, node A becomes faulty and tries to send a frame in slot 8. FlexWay, however, expects a frame from port 2 as D is scheduled to transmit D2. Even if the frame of the faulty node is first detected or is sensed at the same time as D2, it will be ignored and an appropriate error counter will be incremented for diagnosis purposes. The inhibition holds also in the case when FlexWay receives a frame in a specific slot x during global communication phase whilst no transmission is scheduled in x. In other words, the propagation of babbling idiot failures and short circuits could be impeded as long as, however, the faulty node that wants to grant bus access belongs to a cluster from which no frame is expected. This safety characteristic is similar to the specified central bus

---

6FlexWay must count at least five bits with 1 value to switch on and at least five 0 bits to switch off. 

7Node B has suffered a fail omission failure or it sends a signal with wrong content (fail commission). Both failure types belong to the category of byzantine failures.
guardian[6]. Otherwise, nodes of the system rely on the fault protection mechanisms of FlexRay such as CRC to detect collisions and to invalidate the received faulty frames.

![Figure 8: Connecting FlexWay to node B via SPI.](image)

5 Fault hypothesis

The multi-cluster system has to provide a sufficient degree of fault-tolerance to guaranty its safety-critical function under single-failure hypothesis. FlexWay and each node build one fault-containment region (FCR) that may suffer an arbitrary failure at any time and for an arbitrary time interval. Furthermore, each channel may delay a frame or exhibit noises that affect frames. Nevertheless, this paper is focused on the failures that may knock out the timing requirements of the FlexRay system such as Slightly-Off-Specification, that is when solely a subset of the nodes receive a frame yielding an asymmetric behavior. We consider the following fault assumptions:

- Sync node\(^8\) may behave fail silent (known as outgoing link error), delay a sync frame or transmit outside its slot [5].
- Receiving node may suffer incoming link error during the reception of a sync frame (i.e. the bus driver suffer a transient failure).
- FlexWay may behave faulty and thus delay sync frames. It may have a long power on setup phase and may suffer a blackout or crash failure. Further, it may receive the control signal too late.
- Noises may occur on a channel and thus may invalidate a frame.

![Figure 9: Redundant multi-cluster FlexRay system.](image)

5.1 Fault-Tolerance Capabilities and Safety Measures

One of the key aspects of the FlexRay protocols is its capability to handle many single-failure modes. It provides a 11 bit CRC for a relevant part of the header and 24 bit CRC for the entire frame with a hamming distance of 6 if the frame length is less than 248 bytes and 4 if it is between 249 and 254. Further, it can make use of the replication channels to overcome channel failures. The protocol operation control moves to passive mode or halt mode if error conditions are detected [1]. The fault-tolerance algorithm FTM tolerates transient/permanent and symmetric/asymmetric fault class. Bus designer must reckon with longer power-on setup times, forwarding delays, blackout and the crash of FlexWay. In order to mask these failures, a redundant FlexWay component should be deployed as depicted in Figure 9. This architecture will not cause modification neither at the protocol nor at the application layer.

![Figure 10: Interrupt based switching example.](image)
6 Timing Analysis

This section gives a basic analysis of the clock synchronization paradigm in FlexRay regarding the multi-clusters architecture. We evaluate whether the inter frame gap is sufficient to perform the switching according to the described method 3. As stated in section 2, a sender of a frame starts the transmission at the action point offset from its point of view of global time. This offset represents a safety margin that takes the time deviation between any two correct nodes on the network into consideration. Basically, bus designer have to configure the action point offset with a large value in order to avoid slot boundary violations and thus to heighten the robustness. A high value, however, will heavily decrease the bandwidth. For this reason, a trade off that assures a sufficient safety degree and a high bandwidth usage must be made. The lower bound for the action point offset is given as follows[1]:

\[ a = \left\lfloor \frac{2 \times \Pi + \delta_{\text{min}}^N + 2 \times I_{\text{error}}}{T_{\text{MT}} \times (1 - c_{\text{max}})} \right\rfloor \] (1)

Herein \( \Pi \) stands for the assumed precision expressed in \( \mu s \), \( \delta_{\text{min}}^N \) is the minimum propagation delay, \( I_{\text{error}} \) is the maximum initialization error that an integrating node may show during the startup phase, \( T_{\text{MT}} \) represents the duration of a macrotick and \( c_{\text{max}} \) accounts for the maximum clock deviation of the used clock oscillators expressed in ppm (part per million)\(^9\). The lower bound of the initialization error can be given as follows[1]:

\[ I_{\text{error}}^{\text{min}} = 2 \times T_{\text{tick}} \times (1 + c_{\text{max}}) + \delta_{\text{max}}^N \] (2)

and the upper value is:

\[ I_{\text{error}}^{\text{max}} = \Pi \] (3)

The parameter \( T_{\text{tick}} \) in equation (2) denotes the maximum\(^10\) nominal duration of a microtick over all nodes.

Equation (1) states that the action point offset depends on the assumed worst case system precision, the propagation delay and the maximum clock tolerance over all nodes \( c_{\text{max}} \). The employment of high quality oscillators which have a small clock tolerance value will decrease the action point offset value as long as the network propagation and the precision do not exceed specific bounds. However, high quality oscillators are very expensive. Therefore, it is necessary to match the delays and derives the lower bound of the system precision based on the maximum clock tolerance value specified for FlexRay\(^11\).

6.1 Network delay

In order to calculate the delays and the worst case precision, the following assumption are made:

- **Bounded transmission delays**: the transmission delay of a frame and a control signal does not exceed a specific delay.
- **Bounded drift rate**: the absolute value of the drift rate \( \rho_i \) of node i clock is bounded by \( c_{\text{max}} \).
- **Minimum number of sync nodes**: at least \( 3F + 1 \) sync nodes are required to mask up to \( F \) arbitrary failures [7].

Moreover, we introduce the following variable:

- **Transmission delay** caused by the bus driver of the transmitter: \( \tau \)
- **Bus delay**: \( \beta \)
- **Receiving delay** caused by the bus driver of the receiver: \( \sigma \)
- **Activity detection delay** to detect the beginning of an incoming frame plus the forwarding delay: \( \varphi \)

In what follows we define for each variable \( x_{\text{min}} \) the minimum value and \( x_{\text{max}} \) the maximum value. The propagation delay for a local frame in

---

\(^9\)Each clock is characterized by the frequency stability and the frequency tolerance.

\(^10\)Clocks of the nodes may configure different values for the microtick duration. The nominal duration of a macrotick, however, is the same for all clocks.

\(^11\)The maximum permissible clock tolerance value specified for FlexRay is 1500 ppm[1]
cluster \( n \) is:

\[
\delta_{N_n}^{\min} = \tau_{\min} + \beta_{n}^{\min} + \sigma_{\min}^{\min} \tag{4}
\]

\[
\delta_{N_n}^{\max} = \tau_{\max} + \beta_{n}^{\max} + \sigma_{\max}^{\max} \tag{5}
\]

The propagation delay of a global frame is:

\[
\delta_{N}^{\min} = 2 \times \tau_{\min} + \beta_{1}^{\min} + \beta_{2}^{\min} + \varphi_{\min} + 2 \times \sigma_{\min}^{\min} \tag{6}
\]

\[
\delta_{N}^{\max} = 2 \times \tau_{\max} + \beta_{1}^{\max} + \beta_{2}^{\max} + \varphi_{\max} + 2 \times \sigma_{\max}^{\max} \tag{7}
\]

### 6.2 Worst Case Network Precision

The worst case precision is the maximum clock deviation between any two correct nodes during the entire communication. It represents the performance measure of the clock synchronization. The FTM algorithm\[7\] ensures convergence even in the presence of up to \( k \) byzantine-faulty nodes. The amount of \( k \) depends on the number of sync nodes. The worst case precision is given as follows[1]:

\[
\Pi = (34 + 20 \times \lambda) \times T_{\text{tick}} + 2 \times \delta_{\text{max}}^{\text{max}} \tag{8}
\]

Substituting \( I_{\text{error}} \) by its maximum value, we obtain:

\[
a_{\text{max}} = \left[ 8 \times \delta_{N}^{\max} + \delta_{N}^{\min} + 4 \times (34 + 20 \times \lambda_{\text{pec}}) \times T_{\text{tick}} \right] \frac{T_{\text{MT}}}{1 - \epsilon_{\text{max}}} \tag{9}
\]

### 6.3 Best-Case Network Precision

The best case precision is the maximum time deviation between any two correct nodes within the network when byzantine errors are not taken into consideration. This means that all nodes rely on the same sync frames to calculate the clock deviations. The best case precision is given as follows[1]:

\[
\Pi_{\text{best}} = (12 + 6 \times \lambda) \times T_{\text{tick}} + \delta_{N}^{\max} - \delta_{N}^{\min} \tag{10}
\]

### 6.4 Calculation of the Control Signal Delay

In order to meet the timing requirements of the system, determining the worst case control signal delay is a necessary step. The transmission of this signal takes place within the interrupt service routine requested by the cycle start and by the timer interrupts. There are three possible delay sources:

- **Interrupt delay:** the communication controller of the controlling node generates an interrupt request to the host each time the timeout of the used timer is expired. The host should interrupt the execution of the running program and service the request according to a specific priority scheme. The time spent to execute the first line of the service routine, known as interrupt latency, is the time required to finish the current instruction\[12\] to save the cpu registers on the stack, to push the program counter on the stack, to fetch the interrupt vector and to modify the program counter. The time spent since the start of slot or cycle by the communication controller to generate the appropriate interrupt should also be taken into account. The accumulated time is denoted : \( t_{\text{int}} \). For the used \( \mu \text{-controller} \) (see section 7.2 up to 21 bus clock cycles are needed to service an interrupt request. The maximum value of the bus clock frequency is 24 MHz. That means that the amount of \( t_{\text{int}} \) is at least \( 24 \times 21 \) plus the time needed by the communication controller to generate the interrupt request.

- **SPI transmission:** the host must transmit the control signal with a switching command (approach 1) or with the current slot as content (approach 2). The host should first get access to the slot counter register which is hold in the controller memory. It has then to shift the bits with the selected SPI baud rate. The time spent to transmit the control signal via SPI is denoted: \( t_{\text{spi}} \).

- **Processing time within FlexWay:** upon the receipt of the control signal, the control logic of FlexWay stores the content of the received signal and prepares the switching. The time spent to perform these steps is denoted: \( t_{\text{sw}} \)

The control signal delay can be given as:

\[
t_{\text{c}} = t_{\text{int}} + t_{\text{spi}} + t_{\text{sw}} \tag{11}
\]

### 6.5 On the proper Value of the Action Point Offset

As depicted in Figure 12, bus designer must reckon with the worst case scenario, that is when

\[\text{The slowest instruction of the used \( \mu \text{-controller} \) (host) should be considered.}\]
the controlling node is the slowest node of the system. Any value of the action point offset must contemplate the potential delay of the control signal as well as the worst case system precision to ensure a reliable switching. According to the result obtained in section 6.1, 6.2 and 6.4, there are two possible cases:

**case 1:** the sum of the action point offset value and the minimum network propagation delay is greater than the amount of the worst case precision plus the maximum control signal delay.

\[ \Pi + t_c = \Pi + t_{int} + t_{spi} + t_{sw} < a + \delta_N^{min} \]  

**case 2:** the amount which consists of the action point offset and the minimum network propagation delay does not covers the maximum control signal delay plus the worst case precision. In that case, the action point offset has to adopt the amount of the precision plus the transmission signal delay. Regarding this new amount, the bandwidth diminishes in each cycle by \( a^* \):

\[ a^* = (t_c + \Pi - a) \times n \]  

Thereby, \( n \) stands for the number of slots used in the static segment and \( a \) is calculated using equation (9). The general formula for the action point offset can be given as follows:

\[ a = \max \left( \Pi + t_c, \left[ \frac{2 \times \Pi + \delta_N^{min} + 2 \times I_{error}}{T_MT \times (1 - e^{max})} \right] \right) \]

7 Implementation and Experimental Results

7.1 Implementing FlexWay

In order to experimentally evaluate the proposed switching concept, a VHDL (Hardware Description Language) model at the register transfer level was deployed (Figure 22). The programmed code was executed on a DE2 FPGA board[12] and represents the described FlexWay component. This latter is granted access to the FlexRay bus using a plug-in circuit that maps two FlexRay transceivers to its general purpose I/O pins. Each transceiver is connected to a cluster. The FPGA receives a frame from cluster 0 when the associated signal of port BD0_RXEN goes low. If the forwarding is enabled for this slot, it activates BD1_TXD for the retransmission and maps the received frame data from BD0_RXD to BD1_TXD. The same holds for the other direction. Figure 13 depicts the logical block diagram of the realized FlexWay and the associated ports. With respect to Figure 8, the FPGA board establishes a SPI connection to one node[13].

Upon the receipt of the control signal, the following process updates the value of the register \( reg_{val} \). This latter holds the switching command for approach 1 and the slot counter value for approach 2:

\[ \text{write : process (reset, w_req) is begin} \]
\[ \text{if reset = '1' then} \]
\[ \text{reg_val <= (others => '0');} \]
\[ \text{elsif w_req = '1' then} \]
\[ \text{reg_val(31 downto 0) <= w_data(31 downto 0);} \]
\[ \text{end if;} \]
\[ \text{end process;} \]

The switching inside the board is realized as a Moore state machine with two states. The following listing summarizes the relevant part of the VHDL code:

\[ \text{ctrl_fsm : process (reset, clk) is begin} \]
\[ \text{if reg_val(0) = '1' then} \]
\[ \text{--switch on} \]
\[ \text{when IDLE =>} \]
\[ \text{--idle state} \]
\[ \text{if BD0_RXEN = '0' then} \]
\[ \text{state := FORWARD;} \]
\[ \text{direction := BUS0_to_BUS1;} \]
\[ \text{counter := COUNTER_INIT;} \]
\[ \text{elsif BD1_RXEN = '0' then} \]
\[ \text{state := FORWARD;} \]
\[ \text{direction := BUS1_to_BUS0;} \]
\[ \text{counter := COUNTER_INIT;} \]
\[ \text{end if;} \]
\[ \text{when FORWARD =>} \]
\[ \text{--forwarding state} \]
\[ \text{if direction = BUS0_to_BUS1 then} \]

Figure 13: Relevant part of FlexWay block diagram.
forward_bus0_to_bus1 <= '1';
elseif direction = BUS1_to_BUS0 then
  forward_bus1_to_bus0 <= '1';
end if;

if counter > 0 then
  counter := counter - 1;
-- frame forwarding
else
  state := IDLE;
  -- transmission
end if;
-- is finished

........
end process;

forward : process(clk,reset) is
if reset = '1' then
  //...
elsif rising_edge(clk) then
  -- forwarding from BUS0 to BUS1
  if (forward_bus0_to_bus1 = '1') then
    BD0_TXD <= '1';
    BD0_TXEN <= '1';
    BD1_TXD <= BD0_RXD;
    BD1_TXEN <= BD0_RXEN;
  -- forwarding BUS1 to BUS0
  elsif (forward_bus1_to_bus0 = '1') then
    BD0_TXD <= BD1_RXD;
    BD0_TXEN <= BD1_RXEN;
    BD1_TXD <= '1';
    BD1_TXEN <= '1';
  end if;
end if;
end process;

7.2 Experimental Results

7.2.1 Experiments for approach 1

The implemented version of approach 1 was tested in a real system comprising two clusters as shown in Figure 14. Each contains two FlexRay nodes that own a HCS12 μC as host and MFR4300 as FlexRay communication controller [13]. The MFR4300 is configured to generate cycle start and an absolute timer interrupts. Both of them belong to the category of masquable interrupts. The MFR4300 controller has the highest priority comparing with the existing peripheral devices of the nodes. In addition, we implemented the two-level priority strategy to ensure the interruption of any other interrupt service routine that is executed when MFR4300 interrupts occur. This is done by clearing the I-bit in the first line of each ISR that services any masquable interrupt request initiated by a source other than MFR4300. In order to conduct the appropriate experiments, we calculate first the amount of the action point offset using the values summarized in Table 3: The worst case precision (equation (8)) and the action point offset (equation (9)) adopt, thus, the following values:

\[ \Pi_{\text{best}} = 687 \text{ ns} \]
\[ \Pi = 2402 \text{ ns} \]
\[ a = 10 \text{ MT} \]

Table 3: Values of the configuration parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>value</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta_0 )</td>
<td>2.5 ns</td>
<td>5 ns/m</td>
</tr>
<tr>
<td>( \tau_{\min} )</td>
<td>30 ns</td>
<td>measured</td>
</tr>
<tr>
<td>( \tau_{\max} )</td>
<td>50 ns</td>
<td>data sheet [14]</td>
</tr>
<tr>
<td>( \sigma_{\min} )</td>
<td>28 ns</td>
<td>measured</td>
</tr>
<tr>
<td>( \sigma_{\max} )</td>
<td>30 ns</td>
<td>data sheet [14]</td>
</tr>
<tr>
<td>( \phi_{\min} )</td>
<td>68 ns</td>
<td>measured</td>
</tr>
<tr>
<td>( \phi_{\max} )</td>
<td>71 ns</td>
<td>measured</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>2 \mu T</td>
<td>configured</td>
</tr>
<tr>
<td>( T_{\text{tick}} )</td>
<td>25 ns</td>
<td>configured</td>
</tr>
<tr>
<td>( T_{\text{MT}} )</td>
<td>1 \mu s</td>
<td>configured</td>
</tr>
<tr>
<td>( c_{\max} )</td>
<td>500 ppm \mu s</td>
<td>configured</td>
</tr>
</tbody>
</table>

Figure 14: Base topology for the experiments.

The interrupt delay \( t_{\text{int}} \) as well as the transmission time of the control signal via the SPI interface \( t_{\text{spi}} \) sent by the controlling node can be measured using its local view of the global FlexRay time. The global time, the current macrotick in the cycle, is maintained in the control register MCTCTR. Because the ISR and the related transmission of this signal are executed by the host, read access of MCTCTR causes some delays that affect their exact measurement. Further, the value of the register is expressed in macrotick granularity that corresponds to 1 \( \mu \)s. For this reason, the values obtained for \( t_{\text{int}} \) and \( t_{\text{spi}} \) using the following code represent an overestimation of the real values.

```c
// interrupt FR_ISR(void)
{
  if(pMFR4300->sInterrupt.PIFR0 & 0x01) // cycle start ISR
    time1 = pMFR4300->sProtocol_Status.MTCTR;
    transmit control signal
    TransmitSpiData(SPI0, 0x00);
    time2 = pMFR4300->sProtocol_Status.MTCTR;
    printf("start=%d---end=%d\n",time1,time2);
    pMFR4300->sTimer.TI1MTOR = 0x230; // slot 15
    // clear interrupt flag
    reg = pMFR4300->sInterrupt.PIFR0;
    pMFR4300->sInterrupt.PIFR0 = reg;
  }
  // timer is elapsed
  if(pMFR4300->sInterrupt.PIFR0 & 0x02)
    if(pMFR4300->sProtocol_Status.SLTCTAR == 15)
      // next timeout at slot 35
      pMFR4300->sTimer.TI1MTOR = 0x550;
      TransmitSpiData(SPI0, 0xFF); // switch on
    else if(pMFR4300->sProtocol_Status.SLTCTAR == 35)
      //...
```
pMFR4300->sTimer.TICCR |=0x4;
TransmitSpiData(SPI0, 0x0); // stop timer
if(pMFR4300->sProtocol_Status.MTCTR > 1600) {
    // stop timer
    pMFR4300->sTimer.TICCR |=0x4;
    TransmitSpiData(SPI0, 0x0);
    } else if(pMFR4300->sProtocol_Status.SLTCTAR == 36) {
    TransmitSpiData(SPI0, 0xFF);
    // switch on
}

The bus clock frequency used by the host is 24 MHz. This frequency allows the generation of 12 MHz baud rate for the SPI interface.

\[
t_{int} \approx 2 \, MT = 2 \mu s
\]
\[
t_{spi} \approx 4 \, MT = 4 \mu s
\]

The maximum value of \( t_{sw} \) has been determined using the timing analysis tool[12].

\[
t_{sw} = 8\, ns
\]

With respect to inequation 12 the parameter \( a \) is configured to be 10 MT. For all experiments the cycle is configured to be 2 ms with 40 slots. Each slot has a duration of 40 MT. Moreover, each experiment is repeated 10 times in order to achieve a high degree of confidence for the gathered results.

**Experiment 1: Switching Service Test**

Several experiments has been conducted in order to test the implemented switching service. Figure 15 shows the output of node 0 (left) and node 3 (right) that belong to the two clusters\(^\text{13}\). Thereby, slot 1 until slot 14 as well as slot 35 are allocated for local communication while global communication is executed in the remaining slots. It is obviously, that node 0 and node 3 receive different frames in slot 35 while the same frame is received in slot 15. The table below outlines the schedule used in this experiment:

<table>
<thead>
<tr>
<th>slot</th>
<th>sender</th>
<th>receiver</th>
<th>frame type</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>local cluster 0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>local cluster 1</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>all</td>
<td>global</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
<td>all</td>
<td>global (sync)</td>
</tr>
<tr>
<td>25</td>
<td>3</td>
<td>all</td>
<td>global (sync)</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>all</td>
<td>global (sync)</td>
</tr>
<tr>
<td>35</td>
<td>1</td>
<td>0</td>
<td>local cluster 0</td>
</tr>
<tr>
<td>35</td>
<td>2</td>
<td>3</td>
<td>local cluster 1</td>
</tr>
<tr>
<td>36</td>
<td>1</td>
<td>all</td>
<td>global (sync)</td>
</tr>
</tbody>
</table>

Table 4: Base schedule for all experiments

is equivalent to 150 ns. In many other runs, its value fluctuates between 5 and 6 \( \mu T \).

\[\text{Figure 15: Switching test.}\]

**Experiment 2: Fault free**

The objective of this experiment is to compare the precision achieved in the previous experiment with the precision obtained when only nodes of cluster 0 (master) transmit sync frames. Thereby, the schedule shown in Table 4 is modified so that slot 20 and 25 remain unallocated. Figure 17 outlines the course of the network precision. The maximum value obtained in several runs oscillates between 10 and 11 \( \mu T \). This result is, however, not surprising since the clock synchronization algorithm achieves a better performance when all nodes exchange their local views of global time.

**Experiment 3: Master slave synchronization**

The objective of this experiment is to compare the precision achieved in the previous experiment with the precision obtained when only nodes of cluster 0 (master) transmit sync frames. Thereby, the schedule shown in Table 4 is modified so that slot 20 and 25 remain unallocated. Figure 17 outlines the course of the network precision. The maximum value obtained in several runs oscillates between 10 and 11 \( \mu T \). This result is, however, not surprising since the clock synchronization algorithm achieves a better performance when all nodes exchange their local views of global time.

**Experiment 4: Immediate Rate and Offset Change**

In this experiment the hosts of cluster 0 nodes add to the calculated rate and offset correction values 7 \( \mu T \) (external rate and offset correction [1]) while the hosts of cluster 1 nodes subtract 7

\[\text{Figure 17: Network precision.}\]
µT from the calculated rate and offset values. The immediate change of the offset and rate is done in the network idle time of cycle 39. It simulates the occurrence of transient errors which cause both clusters to drift apart from each other. This experiment is carried out using the same schedule shown in Table 4. The effect of the injected values is shown in Figure 18. The maximum value of the system precision deteriorates during cycle 40 and 41 until the correction of the offset is done. The influence on the rate, however, linger during cycle 42 and 43. The maximum value observed over 10 runs is 15 µT which corresponds to 375 ns. This value is still lower than the best case precision.

Experiment 5: Recurring Rate and Offset Change
Analogously to previous experiment, nodes of cluster 0 apply an external offset and rate correction with positive values (+7µT) while nodes of cluster 1 add negative values (-7µT). However, the application of these external values is done in each odd cycle from cycle 39 till cycle 100. Figure 19 shows the course of the system precision. The achieved system precision is perspicuously increased when compared with the results of experiment 4.

Experiment 6: Crash of one node
In this experiment, we investigate the amount of the system precision when one node suffers crash error. For this purpose, the host of node 0 sends a freeze command to the communication controller at the boundary of the network idle time of cycle 39. On the start of cycle 41 node 0 host sends a run command in order to perform a re-integration attempt into the ongoing communication. This re-integration persists for five cycles as depicted in Figure 20. It is also shown that the amount of the precision does not deteriorate despite the unavailability of node 0.
7.2.2 Experiments for approach 2

The implemented version of approach 2 was subject to several simulation experiments with fault-injection. For example, a test bench is generated so that BD1_RXEN signal goes low in slot 9,10,11 and 20 requesting the forwarding of the simulated frames (arbitrary values of the BD1_RX). According to the configured schedule, FlexWay should switch off upon the receipt of the control signal in slot 20 for the first time in the cycle. Thus, the time duration from slot 1 until 20 corresponds to a global communication phase. In this period, only the frame with ID 10 should be forwarded. In other words, frames with ID 9 and 11 are sent by faulty nodes as no transmission was scheduled in these slots. The results pictured in Figure 21 shows that FlexWay stays in the idle state in slot 9 and 11 inhibiting the forwarding of faulty frames[5]. This holds, however, as long as the faulty nodes belong to a cluster from which no frame is expected in the global communication slots.

8 Conclusion

We presented an approach to structure FlexRay systems in multi-clusters. We showed that the proposed FlexWay device provides a set of benefits in terms of bandwidth and safety. We analyzed the timing requirements and verified them using several experiments that are based on two approaches. These experiments showed that all nodes of the clusters remain synchronized even in the occurrence of a single failure. In other words, the amount of the system precision does not exceed the calculated worst case value. The proposed switching concept is submitted to detailed formal verification using SAL model checker. The results will be published as soon as possible.

References


[3] XUE Fangxia , YAN Liaoliao , XIE Hong , YAO Yiping ,LIU Haiye, Study on technique of high time coherence used in real time distributed interactive simulation. Computer Ap-


