Time Constrained FlexRay Static Segment Scheduling

Zdeněk Hanzálek, David Beneš
Department of Control Engineering
FEE, Czech Technical University in Prague
Prague, Czech Republic
Email: {hanzalek,benesda1}@fel.cvut.cz

Denis Waraus
Department of Measurements
FEE, Czech Technical University in Prague
Prague, Czech Republic
Email: warauden@fel.cvut.cz

***Abstract*—The objective of the paper is the formulation and solution of the frame packing and scheduling problem for the static segment with respect to AUTOSAR FlexRay COM stack. There are several other works on the static segment scheduling. The problem becomes more constrained since the offsets and deadlines of the signals are taken into account. While assuming offsets and deadlines to be multiples of the cycle length, a solution of the scheduling problem is based on decomposition to separate nodes. The article shows simple and efficient scheduling algorithm, its evaluation on benchmarks with up to 3000 signals per node and simple experiment on real FlexRay HW.**

***Keywords*-scheduling; time-triggered communication protocols; FlexRay;**

I. Introduction

The FlexRay [4] protocol seems to establish a new standard in automotive and together with recent automotive protocols such a CAN, MOST and LIN will form a robust platform for modern cars with x-by-wire systems. BMW, the pioneer in innovative technology, uses the new protocol in their latest model for transmission of large quantities of data within the active chassis system.

Communication protocol can offer different number of the static slots based on its configuration. Nowadays, luxury cars contains up to 70 electronic control units (ECU), which require transmission up to 2500 different signals [1]. Hence, it is essential to develop a powerful algorithm for a frame packaging and scheduling with user defined constrains (signal period, offset, and deadline). In this paper, time window is we considered for each signal given by offset and deadline related to the start of the hyperperiod.

II. FlexRay overview

The FlexRay is a time-triggered communication protocol where the data transmission is based on recurring communication cycle which is composed of static segment, dynamic segment, symbol window and network idle time. The example of communication cycle is depicted in Figure 1.

The static segment comprises a fixed number of equally sized time division multiple access (TDMA) static slots. The segment is designed for frames where predictable frame delay is defined. In each static slot a frame (with a unique identifier frameID) can be transmitted by a node. Once a static slot with a unique identifier is allocated to a specific node, only this node is allowed to transmit during this slot and so protocol ensures collision avoidance. In order to ensure proper slot timing, each single node provides synchronization by synchronization frames - frame where synchronization bit is set. Synchronization is embedded in the standard and is based on differences between real and predicted frame receptions.

The dynamic segment employs the flexible TDMA approach and is intended for the non-critical and sporadic messages with varying length. The symbol window is designed for transmission of protocol symbols used for network wake-up, start-up and testing. The part of time-base correction is performed during the network idle time. The dynamic segment and symbol window are optional part of the communication cycle. The frame scheduling on the dynamic segment (see [10], [9]) is out of the scope of this study.

The frame includes previously mentioned frameID which matches to slot identifier in the static segment and the cycle number identification. There are 64 different cycles and up to 2047 static slots. The FlexRay network (cluster) can be arranged in different physical topologies such as a star, a bus, a point-to-point or a hybrid topology. Two independent physical channels should be preferably used for high bitrate or enhanced safety. Every single channel can form a different physical topology. In this paper, it is assumed, that the same data are transmitted on both channels.

III. Related work

The goal of the paper is the formulation and solution of the frame packing and scheduling problem for the static segment with respect to AUTOSAR FlexRay COM stack. There are several other works on the static segment scheduling. Techniques for determining the timing properties of messages transmitted in both the static and the dynamic segments of a FlexRay communication cycle are proposed in [9]. The analysis techniques for messages are integrated in the context of a holistic schedulability analysis that computes the worst-case response times of all the tasks and messages in the system. Authors present and evaluate three optimization algorithms that can be used to improve the schedulability of a system that uses FlexRay. Unfortunately, authors assume knowledge of all signals in system for bandwidth optimization by modification of protocol parameters such as number of static slots, static and dynamic segment duration. The assumption is quite limiting for automotive industry applications where an incremental design is commonly used.

A mathematical model determining the optimal length of static messages that can achieve more efficient use of a FlexRay is presented in [12]. However, knowledge of all signals in a system is once again required. Moreover, the longest high priority frames are reallocated to the dynamic segment and thus system have to be redesigned with each extension; otherwise it cannot be guaranteed that frames in dynamic segment meet deadlines.

The message frame packing that maximizes network efficiency by reducing unused slots is investigated in [11]. Signals are packed into the same size messages to obey the restrictions of the FlexRay protocol, while narrow bandwidth is used. Then authors formulate a nonlinear integer programming problem to compute an optimal message set from given set of signals. They also proposed a scheduling method with low jitter and minimum number of static slots to improve the network efficiency. In contrast to [11], the offsets and deadlines of the signals are taken into consideration in this paper, so the problem becomes more constrained.

The authors of [5] assume usage of AUTOSAR COM stack and use heuristic analysis to schedule relatively large message set. The authors assume that frame packing is done by the application level.

An optimization framework that includes not only signal to frame packing, but also frame to slot assignment, task schedule and the synchronization of signal and task scheduling considering end-to-end delays and the precedence constraints induced by information passing between tasks and signals is presented in [13]. Presented mixed-integer linear programming formulation includes the system-level schedule optimization with the definition of an optimal relative phase in the activation of tasks and signals which accounts for deadlines and precedence constraints. Moreover, usage of AUTOSAR standard is assumed, but framework is evaluated only on typical X-by-wire application.

IV. Problem Statement

Two sets of tasks executed in ECUs are to be considered: the first set contains tasks that are running without synchronization with communication cycle. Signals (ie. communication of one state variable from one ECU) generated by these tasks are called *asynchronous signals*. Signals provided by the second set of task which are synchronized with communication cycle are called *synchronous signals*. The objective is to find packing of signals into frames and allocation of frames to specific slot and cycle.

FlexRay network configuration consists of large parameter set including a cycle length, number of static slots in the

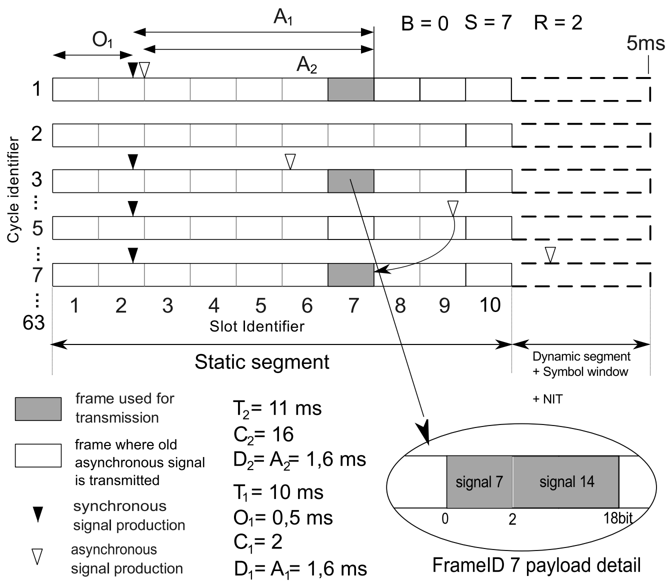

Figure 1. Illustration of generation of synchronous and asynchronous signals

static segment, duration of static slot, duration of NIT segment etc. These parameters are usually predefined by system designer and further followed by ECU suppliers. The following setting based on the BMW network design [2] is used in this paper. The communication cycle duration is equal to F = 5 ms where static segment takes 3 ms and the rest of the communication cycle is filled by the dynamic segment, symbol window and network idle time. There are M = 75 static slots with static slot duration of L = 0,04 ms. The frame payload W = 128 bits.

For each single signal an user can define the following parameters (we use same denotation with [5]):

N_i	unique identifier of ECU, which transmits the signal *i*,
T_i	signal period *i*, the signal is assumed to be transmitted only once in Flexray cycle,
O_i	is the signal offset, it is the latest time after which the first occurence of the signal is produced in relation to the start of period,
C_i	signal *i* size in bits,
A_i	the signal age is the duration between the signal production at the sender side and the end of the first frame carrying the signal on receiver side.
D_i	is the deadline associated to the signal *i*, it represents the maximum age acceptable at the consumer end.

Next assumption is usage of AUTOSAR communication stack[8] which is de facto standard for automotive applications and it is considered also for other industry applications such as aviation. In this context, the AUTOSAR frame is defined by

Q_j	unique ECU identifier, which transmits the frame *j*,
S_j	static slot identifier used for frame *j*,
R_j	the AUTOSAR frame repetition, which is the transmission period of frame expressed as a multiple of the communication cycle with the constrains that the repetition takes its value in $\{2^n \mid n \in \mathcal{N}, n \leq 6\}$,
B_j	the base cycle identifier - the first occurence of the frame *j*. $B_j \leq R_j$.

Each single frame *j* is thus characterized by a tuplet $\{Q_j, S_j, B_j, R_j\}$. The frame contains processing data units (PDU) where each consists of one or more signals. An *update bit* can be defined for each PDU. It informs recipients whether at least one signal in the PDU was updated or not. The signal can be contained in more than one PDU. In AUTOSAR context is defined that scheduling is static and cannot be changed during runtime.

It is further assumed, that maximal signal size cannot exceed data payload of the frame. It means that signal cannot be split into two or more frames. Message fragmentation is ensured by AUTOSAR FlexRay transport protocol layer [7].

By oversampling, the asynchronous signal is transformed to synchronous one with the following parameters:

$$\begin{aligned} D_i = T_i &= 2^n \cdot F + P + U \\ O_i &= 0 \end{aligned} \qquad (1)$$

where P is packing time - time from task request on signal *i* transmission to the start of frame transmission and U is unpacking time.

Synchronous signals can keep deadline shorter than a period. In this case deadline is limited by the following values

$$D_i^S \geq \begin{cases} P + L + U \text{ for } & 0 \leq O_i - n \cdot F \leq P + M \cdot L + U \\ P + L + G + U \text{ otherwise} \end{cases} \qquad (2)$$

where $n \in 1, 2, ..64$, M is number of static slots, L is duration of the static slot, and G is duration of dynamic segment, network idle time and symbolic window.

The Figure 1 depicts a situation where two signals are generated – synchronous one and asynchronous one. The period of synchronous is set to 10 ms and 11 ms for asynchronous. Let us assume that, both signals have the deadline equal to the period. In order to fulfill the deadline constraint, it is necessary to transmit both signals every second cycle, it means $R_j = 2^1$. In the fourth cycle ECU transmits the same data twice due to oversampling of the asynchronous signal. However, this behavior is covered by previously mentioned *update bit*.

A. Message set

The scheduling approaches are applied to the modified Society of Automotive Engineers (SAE) benchmark signal set.

The SAE report describes a set of signals sent between seven different subsystems in an electric car prototype. The Basic SAE signal set defines 53 sporadic and periodic signals. A periodic message has a fixed period 5, 10, 100, 1000 ms, and implicitly requires the latency to be less than or equal to this period. Sporadic messages have latency requirements imposed by the application: for example, all messages sent as a result of a driver action have a latency requirement of 20 ms. The reader is referred to the work of Kopetz [6] for more detailed benchmark description. Due to the high FlexRay bandwitch (10Mb/s) and electronic equipment's requirements in today's cars we extended the SEA benchmark using NETCARBENCH[3] to:

- increase the number of signals while using the same probability distribution function of parameters - up to 3000 signals per node are considered in this article,
- add parameters (such as offset).

B. Time constraints

In the first step, it is needed to find each signal pair $\{E_i, H_i\}$ where E is earliest and H latest slot which can be used for transmission of the first occurrence of the signal *i*. The pair is given by

$$\begin{aligned} E_i &= floor(\frac{P + O_i}{F}) \cdot M + \\ &+ min \quad (ceil(\frac{(P + O_i - floor(\frac{P+O_i}{F}) \cdot F}{L}), M) \end{aligned} \qquad (3)$$

$$\begin{aligned} H_i &= floor(\frac{P + O_i}{F}) \cdot M + \\ &+ min \quad (floor(\frac{P + O_i + D_i - floor(\frac{P+O_i}{F}) \cdot F}{L}), M) \end{aligned} \qquad (4)$$

In order to simplify the scheduling problem, each node is scheduled separately, following decomposition in [11]. For this reason, we express $\{\tilde{O}_i, \tilde{D}_i\}$, earliest and latest cycle to transmit signal i, as follows:

$$\tilde{O}_i = ceil(\frac{E_i}{M}) \qquad (5)$$

$$\tilde{D}_i = floor(\frac{H_i}{M}) \qquad (6)$$

This simplifies our scenario, since offsets and deadlines could be only multiples of the cycle length, because position of signal within the cycle is insignificant with respect to position of signal within hyperperiod. This effectively means, that we don't care about assignment of particular ID to the node, so it is not important, that the node has the first ID, or the last one.

V. Algorithm

The Time Constrained FlexRay Static Segment Scheduling (TCFSSS) algorithm could be shortly described in pseudo code of Algorithm 1 where $schedule$ is the actual ID, cycle and offset in the frame, when signal will be first send over the network.

Input: C, T, $\tilde{O}$, $\tilde{D}$
Output: $schedule$

```
$[signals_{node_1},\, ...,\, signals_{node_{number\, of\, nodes}}] =$
        $= divide\, signals\, to\, nodes(C,\, T,\, \tilde{O},\, \tilde{D})$
$for\ (k = 1 \,...\, number\, of\, nodes)\ do:$
    $messages_{node_k} = frame\, packing(signals_{node_k})$
    $opt\, messages_{node_k} = optimization(messages_{node_k})$
    $schedule_{node_k} = create\, schedule(opt\, messages_{node_k})$
$schedule = [schedule_{node_1},\, ...,\, schedule_{node_{number\, of\, nodes}}]$
$if\ (schedule.slots > M)$
    $error('Could\, not\, find\, correct\, schedule')$
```

Algorithm 1: Pseudo code of the TCFSSS algorithm

A. Divide signals to nodes

Due to the fact that offsets and deadlines are multiples of cycle length, it is not needed to decide which node should have lover IDs than the others and this leads us to the possibility to compute schedules separately for each node. Therefore signal are simply divided into the sets according to the node that send the signal. This is performed in asymptotic time complexity $O(S)$ where S is the number of the signals on the input.

Above means, that our algorithm is able to take advantage of parallel processing on multiple processors. After dividing signals, algorithm could be split into independent parts for each node. When branches successfully finishes their computation, final result is obtained just by putting individual results consequently one after another.

B. Frame packing

The idea of frame packing is described in [11]. Frame packing is done separately for each period of signals. Frame packing groups signals into messages that are not larger than the slot and have all properties as a signal, it has the same period T_i as signals packed into this message, size C_i as a sum of the sizes of individual signals in the message and also computed offset $\tilde{O}_i$ and deadline $\tilde{D}_i$ from those signals.

	C_i	T_i	$\tilde{O}_i$	$\tilde{D}_i$
s_1	26	2	0	2
s_2	2	1	0	1
s_3	2	4	0	11
s_4	6	4	0	13
s_5	6	8	1	8
s_6	8	1	0	1
s_7	2	2	0	2
s_8	4	1	0	1
s_9	32	8	5	8
s_{10}	16	2	1	2
s_{11}	4	4	0	9
s_{12}	14	1	0	1
s_{13}	4	1	0	1
s_{14}	16	2	0	1
s_{15}	10	16	1	6
s_{16}	8	2	0	2
s_{17}	4	2	0	2
s_{18}	2	8	3	7
s_{19}	14	16	1	16
s_{20}	20	1	0	1

Table I
Signals on the frame packing input for one node ($signals_{node_7}$)

The message simply groups the signals to simplify scheduling by while reducing the number of elements that need to be scheduled. When searching for the message, where given signal can be placed, only messages with the same period, which has enough free space are taken into account. That means, that size of the message with the size of the added signal could not be larger than the size of the slot. Furthermore, to fit signal into message, offset of the message must be equal or larger than the offset of the signal and message deadline must be shorter than or equal to the signal deadline.

When the signal is added in the message, offset and deadline of the message must be adjusted to fulfill constrains of all signals in the message. If no convenient message (message without enough free space or message, that doesn't satisfy all the constraints) for specified signal is found, new message with given signal's properties (size C_i, period T_i, release time $\tilde{O}_i$ and deadline $\tilde{D}_i$) is created. As an example of frame packing, see Table I which shows input signals for one node. During this part of the algorithm for the input signals in Table I, signals were converted into messages depicted in the Table II.

	C_i	T_i	$\tilde{O}_i$	$\tilde{D}_i$	signals
m_1	32	1	0	1	s_2, s_6, s_8, s_{12}, s_{13}
m_2	20	1	0	1	s_{20}
m_3	30	2	0	1	s_7, s_{14}, s_{16}, s_{17}
m_4	26	2	0	2	s_1
m_5	16	2	1	2	s_{10}
m_6	12	4	0	4	s_3, s_4, s_{11}
m_7	8	8	3	7	s_5, s_{18}
m_8	32	8	5	8	s_9
m_9	32	16	1	6	s_{15}, s_{19}

Table II
Messages created during frame packing for one node ($messages_{node_7}$)

Frame packing is computed in asymptotic time complexity $\mathcal{O}(s^2)$ where s is number of the signals of the given node.

C. Optimize messages

Most likely, not all of the messages would have exactly the size of the slot, so there is another step, which tries to use the empty space in messages for messages, that can fit into the free space and have lower periodicity. Therefore we reduce the number of messages that need to be scheduled, so it can cause reduction of the number of needed IDs allocated to the node.

This is obtained by iterating over all messages (notice, that previous part of the algorithm was performed separately by periods, but this part works on all messages of the node) sorted from the least occurring message to the most occurring one. Each message is checked, if it couldn't be merged with some other message for current node with preserving it's constrains. Sorting messages from the least occurring to the messages with highest periodicity reduces the amount of old unneeded data (data sent more frequently, than is it's periodicity) sent over the network, in the other words algorithm tries to reach the best possible effectivity. Pseudo code of the described method is depicted in Algorithm 2.

```
Input: $messages_{node_k} = [m_1, ..., m_{number\,of\,messages_{node_k}}]$
Output: $opt\,messages_{node_k} = [m'_1, ..., m'_{number\,of\,optimized\,messages_{node_k}}]$

$for\,(p = 1\,...\,number\,of\,messages_{node_k})\,do:$
    $m'_p = m_p$
    $for\,(q = (p+1)\,...\,number\,of\,messages_{node_k})\,do:$
        $if\,(m'_p\,can\,be\,merged\,with\,m_q)$
            $m'_p = merge\,m'_p\,and\,m_q$
            $delete\,m_q$
        $endif$
```

Algorithm 2: The idea of the message optimization part of the algorithm for node k

See Table III for the output of the message optimization part of the TCFSSS alogirithm.

	C_i	T_i	$\tilde{O}_i$	$\tilde{D}_i$	messages
m'_1	32	1	0	1	m_1
m'_2	20	1	0	1	m_2
m'_3	30	2	0	1	m_3
m'_4	26	2	0	2	m_4
m'_5	28	2	1	2	m_5, m_6
m'_6	32	8	3	6	m_7, m_9
m'_7	32	8	5	8	m_8

Table III
OPTIMIZED MESSAGES CREATED DURING THE MESSAGE OPTIMIZATION FOR ONE NODE ($optimized\,messages_{node_7}$)

To meet all constrains of original signals, latest offset and earliest deadline must have been chosen.

Optimization of the messages is done in asymptotic time complexity $\mathcal{O}(m^2)$ where m is the number of the messages from the previous step of the given node.

D. Create schedule

Messages are being scheduled from the most frequent messages to the messages with the longest period. For each message, the algorithm tries to find the ID, which has enough of the free space (empty frames in the cycles of the hyperperiod) and meets all requirements (offset and deadline of the message).

This process is very similar to the frame packing. The messages are separated for suitable position in those IDs which have enough of free space, in this search we consider only the cycles that meet offset and deadline of the message. If proper position is found, it's place is recorded to the message and corresponding cycles are marked (because message generally occupies more frames thanks to it's periodicity) as occupied. If no suitable ID could be found, new ID is reserved, and message is placed into this one and occupied cycles are marked in the same way.

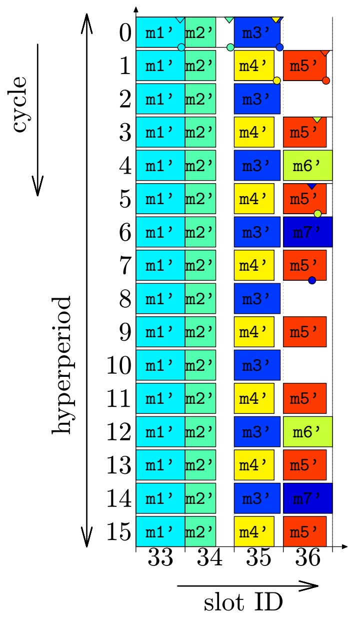

Figure 2. Final computed schedule for one node ($schedule_{node_7}$)

Final computed schedule is shown on the Figure 2. It shows, that 20 signals on the input of the algorithm could be scheduled into 4 slots. At the beginning, signals requested to transfer 1560 bits of information during the hyperperiod. Thanks to the optimization part, in the final schedule, node needs to transfer 1632 bits of the information, so our optimization overhead is in this case 4.6%. In those 4 slots it is possible to transfer at most 2048 bits, that means, that this example use nearly 80% of the available bandwidth.

Creation of schedule is done in asymptotic time complexity $\mathcal{O}\,(max(m' \cdot i \cdot c,\, m' \cdot log(m')))$ where m' is the number of the optimized messages for the given node, i is the number of the IDs associated to the given node and c is the number of cycles in the hyperperiod. $m' \cdot log(m')$ stands for the sort complexity, because optimized messages must be sorted on the input of this part of the algorithm.

VI. PERFORMANCE EVALUATION AND CONCLUSION

Computing time of the algorithm mainly depends on the maximum number of signals for a node. The TCFSSS

# of signals	696	775	973	2667
# of nodes	3	6	1	1
Overhead	5.24%	6.42%	0.54%	0.60%
Utilization	91.74%	82.68%	97.78%	98.05%
Computation	0.84s	0.62s	0.87s	2.10s

Table IV
PERFORMANCE OF TCFSSS ALGORITHM

algorithm was tested on signal sets with three nodes and about 700 signals generated by netcarbench and it is able to compute schedule in about 8 seconds when one node had 400 signals in those signal sets. For signals sets with only one node, but with about 3000 signals per node, takes TCFSSS algorithm about 35 seconds to compute the schedule. Performance was measured on single core Intel 2.4GHz processor and 1GB of RAM, using Matlab R2009B on Windows XP.

Overall time complexity of the TCFSSS algorithm is $\mathcal{O}\left(S + n \cdot (s_n^2 + m_n^2 + max(m'_n \cdot i_n \cdot c, m'_n \cdot log(m'_n)))\right)$ in the worst case scenario, where S is the overall number of the signals on the input, n is the number of the nodes, s_n is the number if the signals of the given node, m_n is the number of the messages of the given node, m'_n is the number of the optimized messages of the given node, i_n is the number of IDs assigned to the given node and c is the number of the cycles in hyperperiod.

Table IV illustrates time complexity of the algorithm for different number of signal. The values of utilization illustrate efficiency of this algorithm. The computation time leaves the room for further extensions of the algorithm.

In order to verify the schedule shown in this paper, a simple experiment have been conducted with real FlexRay HW based on Freescale MC9S12XF and Tektronix oscilloscope MSO/DPO4000B Series. Part of one communication cycle (separate lines represent separate signals) is shown on Figure 3.

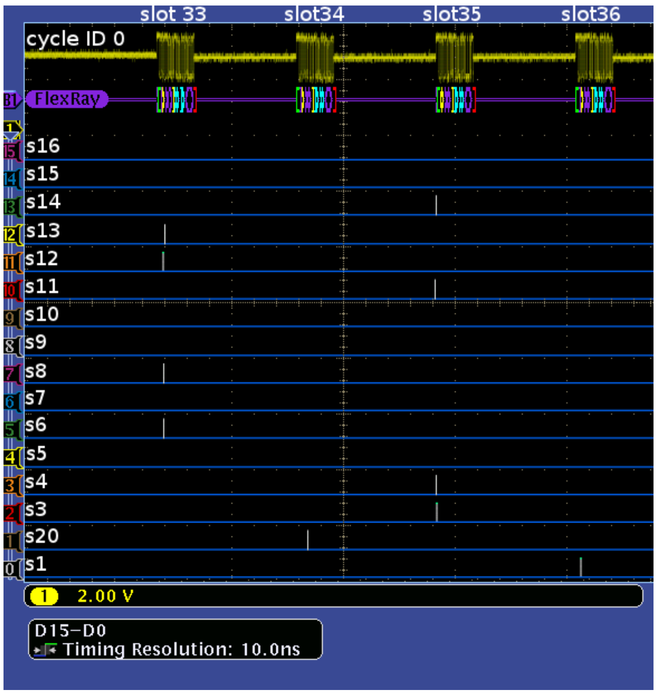

Figure 3. Experiment on real HW

REFERENCES

[1] Amos Albert. Comparison of event-triggered and time-triggered concepts with regard to distributed control systems. In *Embedded World 2004*, pages 235–252, 17.–19.02.2004 2004.

[2] Josef Berwanger, Martin Peteratzinger, and Anton Schedl. Flexray startet durch – flexray-bordnetz får fahrdynamik und fahrerassistenzsysteme. *In Elektronik automotive: Sonderausgabe 7er BMW*, 2008.

[3] Christelle Braun, Lionel Havet, and Nicolas Navet. Netcarbench: A benchmark for techniques and tools used in the design of automotive communication systems. In *The 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems (FeT'2007)*, November 2007.

[4] FlexRay Consortium. Flexray protocol specification v2.1 rev. a, 2005.

[5] Mathieu Grenier, Lionel Havet, and Nicolas Navet. Configuring the communication on FlexRay: the case of the static segment. In *Embedded Real Time Software*, France, 2008.

[6] H. Kopetz. A solution to an automotive control system benchmark. In *Real-Time Systems Symposium, 1994., Proceedings.*, pages 154 –158, 7-9 1994.

[7] AUTOSAR Development Partnership. Specification of flexray transport layer. Version 2.0.1, June 2006.

[8] AUTOSAR Development Partnership. Specification of module flexray interface. Version 2.0.0, June 2006.

[9] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei. Timing analysis of the flexray communication protocol. *Real-Time Systems, 2006. 18th Euromicro Conference on*, pages 11 pp.–216, 2006.

[10] E. G. Schmidt and K. Schmidt. Message scheduling for the flexray protocol: The dynamic segment. *Vehicular Technology, IEEE Transactions on*, 58; 58(5):2160–2169, 2009.

[11] K. Schmidt and E. G. Schmidt. Message scheduling for the flexray protocol: The static segment. *Vehicular Technology, IEEE Transactions on*, 58; 58(5):2170–2179, 2009.

[12] Byungseok Seo and Dongik Lee. Determining the length of static message for efficient use of flexray network. *SICE Annual Conference 2010, Proceedings of*, pages 563 –566, aug. 2010.

[13] Haibo Zeng, M. Di Natale, A. Ghosal, and A. Sangiovanni-Vincentelli. Schedule optimization of time-triggered systems communicating over the flexray static segment. *Industrial Informatics, IEEE Transactions on*, 7(1):1 –17, 2011.