FlexRay Static Segment Scheduling

Martin Lukasiewycz, Michael Glaß, Jürgen Teich, and Paul Milbredt

1 Introduction

The FlexRay protocol was introduced by an international consortium including several car manufacturers to cope with growing real-time requirements of advanced driver assistance functions and safety functions in the automotive domain. The FlexRay protocol offers a static and dynamic segment with a high data rate of 10 Mbit/s. While the event-triggered dynamic segment is used mainly for diagnosis, maintenance, and calibration data, the time-triggered static segment might be used for critical data with strict real-time requirements. In addition to standard linear bus and star topologies, the FlexRay bus allows hybrid topologies including a dual channel mode to increase the reliability. However, in contrast to the prevailing CAN bus [4] in the automotive domain, the configuration of the FlexRay bus is significantly more complex: It requires a large set of parameters and a predefined schedule. This chapter introduces a scheduling concept for the static segment of the FlexRay based on the transformation to a two-dimensional bin packing problem.

Martin Lukasiewycz
TU Munich, Germany, e-mail: martin.lukasiewycz@rcs.ei.tum.de

Michael Glaß
University of Erlangen-Nuremberg, Germany, e-mail: michael.glass@cs.fau.de

Jürgen Teich
University of Erlangen-Nuremberg, Germany, e-mail: juergen.teich@cs.fau.de

Paul Milbredt
AUDI AG, Germany, e-mail: paul.milbredt@audi.de

1.1 FlexRay Protocol

The FlexRay communication is organized in *cycles*, as illustrated in Figure 1. Since each frame has exactly 6 cycle count bits, the cycles are numerated from 0 to 63 and subsequently start over with 0 again. Each cycle is divided into four segments of configurable duration:

(1)The *static segment* enabling a guaranteed real-time transmission of critical data,
(2)the *dynamic segment* (optional) for low-priority and event-triggered data,
(3)the *symbol window* (optional) used to transmit special symbols, and
(4)the *network idle time* used to perform a clock synchronization.

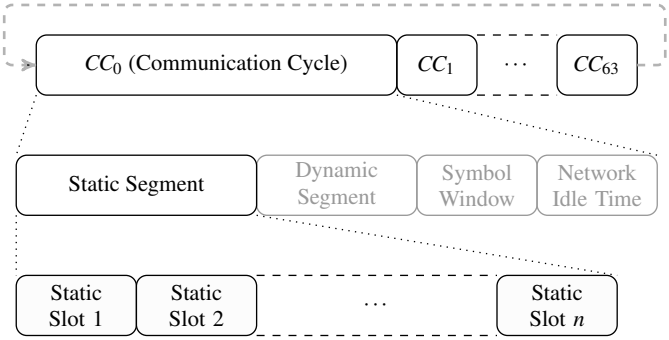


Fig. 1 The FlexRay communication protocol consisting of 64 communication cycles with a detailed illustration of the static segment.

The focus of this chapter is put on new techniques for scheduling the static segment. The static segment is made up of n equally sized *slots* where each one is uniquely assigned to one *node* (or none). One node, however, may occupy more than one slot. Each slot consists of a header and trailer segment and a payload segment that is statically configured to carry between 0 and 254 bytes. By a predefined schedule, each slot is filled with the communication data of the applications. As illustrated in Figure 1, the FlexRay protocol uses a Time Division Multiple Access (TDMA) to multiplex the communication into different slots.

Figure 2 gives an overview of data transmission on the FlexRay bus in the automotive domain. The basic unit are the *signals*, e.g., physical data like the vehicle speed. The signals are packed to *messages* which are measured in bytes as the basic unit. The messages are packed to *frames*. Since all participants of a FlexRay bus are aware of the current cycle number, the cycle is used for a second multiplexing dimension as suggested in the AUTomotive Open System ARchitecture (AUTOSAR) FlexRay Interface Specification [1]. A *slot* may contain different different frames for specific cycles. Here, exactly one (or no) frame is transmitted in a *slot* at one specific cycle to increase the utilization of the bus. Finally, the slots are added to the *schedule*.

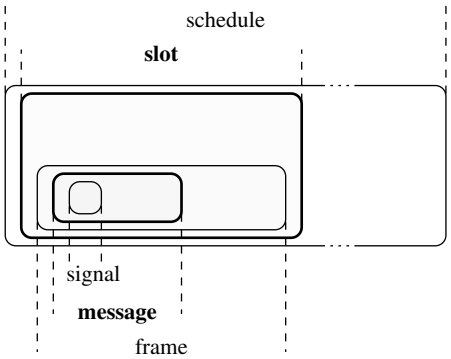


Fig. 2 Overview and terminology regarding the data transmission on the FlexRay bus in the automotive domain.

In the work at hand, it is assumed that the basic communication units are messages, thus, the signals are already packed into messages. This is a common scenario, since messages are predefined by Electronic Control Unit (ECU) and gateway packing strategies. Thus, the goal is to pack messages into slots implicitly defining the frame packing. Note that a two-step approach where messages are first packed to frame and frame to slots might lead to suboptimal solutions with respect to the number of required slots.

1.2 Scheduling Requirements

In real-world implementations of the FlexRay bus, the periodic and safety-critical data is scheduled on the static time-triggered segment while the dynamic segment is mainly used for maintenance and diagnosis data [3, 17]. Though, in the first generation of the FlexRay bus in series-production vehicles, the static segment is not used at the full capacity [17], it is projected that the data volume on FlexRay buses will increase significantly in the future. Therefore, a schedule optimization that minimizes the number of used slots is necessary to allow a high flexibility for incremental schedule changes[1] and for future automotive networks with a higher data volume. Hence, an efficient schedule optimization of the static segment is the key to the success of the FlexRay bus.

The configuration of the FlexRay bus is defined by a large set of parameters. In particular, these parameters allow a configuration of the number and size of the slots in the static segment. Nevertheless, these values are mostly predefined by the manufacturer guided by existing data. For instance, the duration of a communication cycle is usually 5 milliseconds (ms) due to the periods of the messages in the present automotive networks that are predominantly a multiple of 5 ms. For each

[1] incremental changes are common in the automotive area to decrease the testing exposure

messsage that is routed on the FlexRay bus, a fixed size in bytes is given and the minimal repetition is deduced from the period of the communication cycle and its own period, cf. [8]. In order to efficiently improve the tunable FlexRay parameters, fast scheduling techniques are necessary to allow for an effective parameter exploration.

1.3 AUTOSAR Interface Specification

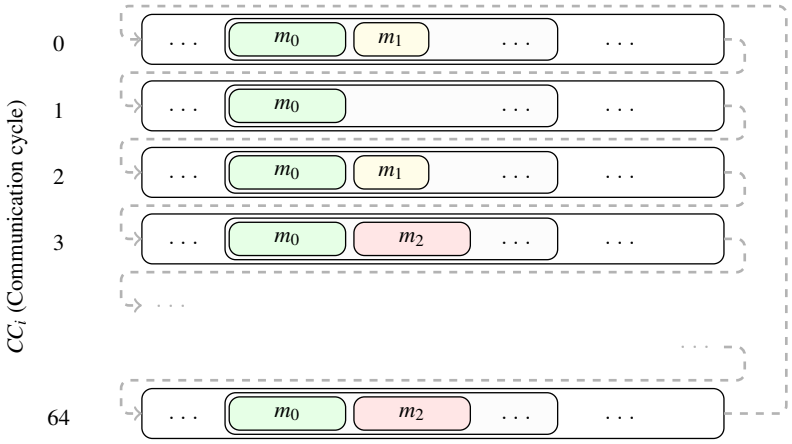
As suggested in the AUTOSAR FlexRay Interface Specification [1] that is currently applied in all series-production vehicles, *cycle multiplexing* is used to increase the utilization of the FlexRay bus. An example of this *cycle multiplexing* for a single slot is illustrated in Figure 3. The cycle multiplexing of messages is defined by the *base cycle* and the *cycle repetition*: The base cycle defines the offset in cycles for the first occurrence of the respective message. The cycle repetition denotes the frequency of a message among the communication cycles. The value of the cycle repetition is always a power of two $2^n, n \in \{0,...,6\}$ to allow a periodic occurrence in the 64 cycles. Thus, for a given base cycle b and repetition r, a message is existent in each communication cycle CC_i with

$$i = (b + r \cdot n)\%64 \text{ with } n \in \mathbb{N}_0. \tag{1}$$

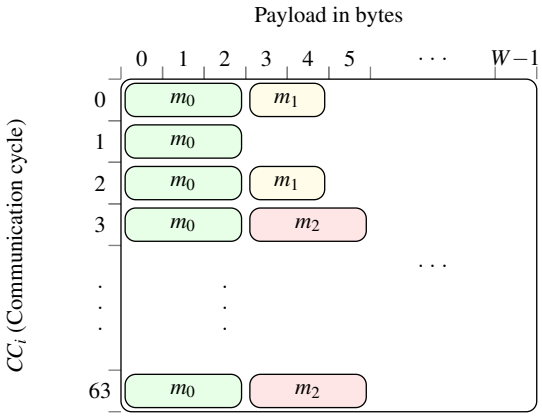
Here, % is the modulo operation. An example of scheduling three messages m_0, m_1, and m_2 is given in Figure 3. The base cycle values are 0 for m_0, 0 for m_1, and 3 for m_2. The repetition values are 1 for m_0, 2 for m_1, and 4 for m_2. Given a common duration of a single communication cycle of 5 ms, the message m_0 is sent each cycle with a period of 5 ms, the message m_1 each second cycle with a period of 10 ms, and the message m_2 each fourth cycle with a period of 20 ms. The cycle multiplexing technique maximizes the utilization of the static segment in compliance with the high requirements for reliability and robustness and, therefore, is integrated into real-world automotive implementations of the FlexRay bus based on the AUTOSAR specification.

2 Related Work

The FlexRay specification [6] is under development by the *FlexRay Consortium* including *BMW*, *Daimler*, *General Motors*, and *Volkswagen*. Currently, the series-production vehicles using FlexRay are the *BMW X5*, *X6* and *7* series [2] and the *Audi A8* [10]. All these series-production vehicles are compliant with the FlexRay AUTOSAR Interface Specification [1]. Thus, this AUTOSAR specification is the de-facto industrial standard for the software specification of the FlexRay nodes.



(a) Cycle multiplexing of three messages into a single slot in the context of the 64 communication cycles.



(b) Cycle multiplexing of three messages into a single slot in a two-dimensional representation.

Fig. 3 FlexRay cycle multiplexing of a single static slot. In order to achieve a high utilization, cycle multiplexing allows each slot to have an individual message scheduling in each communication cycle.

Recent papers cover diverse FlexRay related topics. An introduction of the FlexRay protocol and the operating mode in real-world automotive systems is presented in [1, 3, 20]. In [16], a timing and performance analysis of FlexRay embedded systems is given, mostly focused on the dynamic segment. The determination and optimization of FlexRay schedules for the dynamic segment is discussed in [15]. The work in [11] presents a scheme for the acknowledgment and retransmission of data that is implemented on top of an existing FlexRay schedule in order to increase the reliability of FlexRay-based applications.

An approach that optimizes the static segment with a Genetic Algorithm (GA) is proposed in [7]. The approach in [21] introduces an Integer Linear Programming (ILP) approach for a proposed custom software architecture. In [24], the authors present a Mixed Integer Linear Programming (MILP) approach for scheduling of messages and tasks in a synchronous architecture. However, these papers do not consider the AUTOSAR [1] software specification for the FlexRay bus and are, therefore, not applicable to solve current FlexRay scheduling problems in the automotive domain. Moreover, in the case of the exact ILP and MILP approaches, the scalability might form an obstacle for the applicability since the relevant papers only present relatively small case studies.

A recent work on scheduling the static segment is given in [8] that considers the optimization of the schedule with respect to cycle multiplexing. However, a predefined frame packing as described in [18, 19] is assumed. This approach is rather restrictive since a two-level packing of signals to messages and messages to frames as provided by AUTOSAR is common in the automotive domain. In contrast, the work at hand enables the scheduling of the messages directly into slots including the frame packing. Available tools for solving the scheduling problem are TTX PLAN [22] based on an heuristic approach and DAVINCI NETWORK DESIGNER FLEXRAY [23], a graphical user interface that only allows to build schedules manually. The scheduling algorithm of TTX PLAN is not published, but the experimental results in this work give evidence of an inferior behavior of TTX PLAN regarding runtime and quality of results.

One of the main contributions of the work at hand is a transformation scheme for the FlexRay scheduling problem into a special two-dimensional bin packing problem and its efficient solution. The general two-dimensional bin packing problem has been researched thoroughly, a brief summary is presented in [12]. Since an exact solution based on ILP results in a huge number of variables, cf. [5], heuristics are usually favored for unconstrained problems. However, in the presence of constraints like the level [13] or guillotine packing [17], an ILP can be formulated and solved efficiently. To the best of our knowledge, the special two-dimensional bin packing problem with individual level constraints, as presented in the work at hand, has not been topic of any research so far.

3 Schedule Optimization

The optimization flow as proposed in this work is illustrated in Figure 4. The main

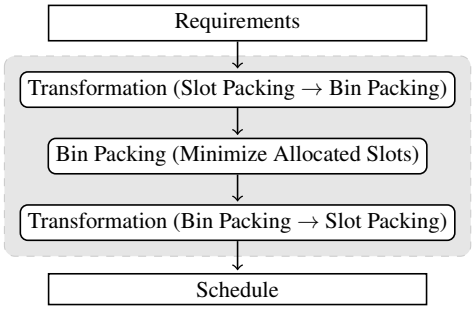


Fig. 4 Schedule optimization flow.

goal is to minimize the number of used slots in order to maximize the utilization of the bus. Unused slots are still part of the final schedule, but these slots can be assigned to any ECU if the schedule is extended incrementally in further development. Moreover, a fast scheduling approach is advantageous, e.g., for the exploration of specific bus parameters.

First, the transformation of the original slot packing problem into a special *bin packing* problem is performed using the proposed transformation scheme. The bin packing is carried out in order to minimize the number of allocated slots. The work at hand introduces a fast heuristic and an exact ILP approach for this special bin packing problem. Finally, the transformation is inverted to convert the solution of the bin packing to a feasible FlexRay schedule. Since each slot is assigned to at most one ECU, the scheduling for each ECU is done independently and the slots are put together in the final schedule.

3.1 Problem Transformation

This section describes a one-to-one transformation between the slot packing problem that arises from the FlexRay cycle multiplexing and a special form of a two-dimensional bin packing problem. Since each slot corresponds to one bin, the transformation is presented for a single slot to a single bin and vice versa.

First, the general conditions for a feasible FlexRay slot packing are introduced: Each slot is defined by the payload size W (without the reserved load for the AUTOSAR specific update bits) and the number of cycles H which is 64 for the FlexRay bus. The set of messages is denoted M.
Each message $m \in M$ is defined by the following two values:

- $w_m \in \mathbb{N}$ - byte-length with $w_m \leq W$.
- $r_m \in \{2^n | n \in \{0,...,6\}\}$ - repetitions in the powers of two, defining the step-size for the multiplexing over the cycles. It holds that $r_m \leq H$.

For a feasible slot packing, two values for each message have to be determined:

- $x_m \in \mathbb{N}_0$ - the offset in bytes on the x-axis.
- $b_m \in \mathbb{N}_0$ - the base cycle that defines the offset on the y-axis. It holds that $b_m < r_m$.

A message is not allowed to exceed the slot ($x_m + w_m \leq W$) and no intersection between two messages is possible. The task of the transformation of the slot packing into a bin packing is to convert each message into a rectangular *element* and determine its position such that each feasible slot packing results in a feasible bin packing and vice versa. The bin size is the same as the slot size with the width W and height H. Also the position on the x-axis x_m and the width w_m for each element m correspond to the position and width of the message in the slot packing. Therefore, the main task is to find a transformation that obtains the following two values for each element m:

- $y_m \in \mathbb{N}_0$ - the offset on the y-axis.
- $h_m \in \mathbb{N}_0$ - the height of an element.

The transformation for the height h_m is related to the repetition r_m:

$$h_m = \frac{H}{r_m} \tag{2}$$

$$r_m = \frac{H}{h_m} \tag{3}$$

Thus, the height of an element equals the number of appearances of the corresponding message in the H cycles.

Given $b_m < r_m$, it follows that in the bin packing problem, the position y_m is restricted to r_m individual levels, depending on the height of the element. It holds that the *level* of an element $l_m = y_m / h_m$ has to be in $\mathbb{N}_0$. This arises from the fact that two messages with the same repetition but different base cycles will never intersect each other. The same holds for elements of the same height but different levels.

Consider the following transformation function $t : \mathbb{N}_0 \times \mathbb{N} \to \mathbb{N}_0$:

$$t(x,y) = \begin{cases} 0, & x = 0 \\ t(\frac{x}{2}, \frac{y}{2}), & x \text{ is even} \\ t(\frac{x-1}{2}, \frac{y}{2}) + \frac{y}{2}, & x \text{ is odd} \end{cases} \tag{4}$$

with

$$y \in \{2^n | n \in \mathbb{N}_0\} \tag{5}$$

$$0 \leq x < y \text{ and } x \in \mathbb{N}_0 \tag{6}$$

It holds:

$$t(t(x,y),y) = x \text{ and } t(x,y) = t^{-1}(x,y) \tag{7}$$

The transformation function t directly transforms the level of an element l_m to the base b_m and vice versa, such that the following holds:

$$l_m = t(b_m, r_m) \text{ and } b_m = t(l_m, r_m) \tag{8}$$

and, thus,

$$y_m = l_m \cdot h_m = t(b_m, r_m) \cdot \frac{H}{r_m} \tag{9}$$

and

$$b_m = t(\tfrac{y_m}{h_m}, r_m) = t(\tfrac{y_m}{h_m}, \tfrac{H}{h_m}). \tag{10}$$

Thus, a transformation from the slot packing problem to a bin packing problem with individual level constraints based on the height of the elements is done by applying Equation (2) and (9) for each message. For the opposite direction, the transformation is performed by Equation (3) and(10) for each element. An example for the transformation of a single slot is given in Figure 5. A detailed proof for the

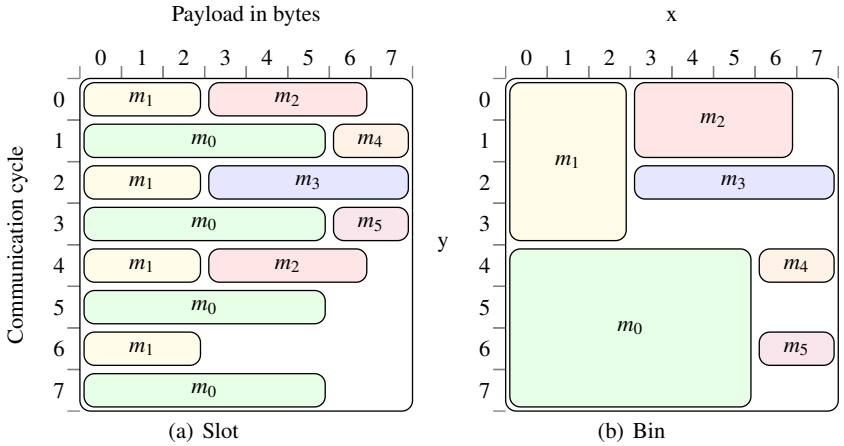


Fig. 5 Example of a transformation from a slot packing problem (a) to a bin packing problem (b) and vice versa.

correctness of this transformation is given in the following.

Theorem 1. *Two elements in the generated bin packing intersect if and only if the corresponding messages in the slot packing problem are conflicting, i.e., intersecting.*

Proof. For the x-transformation, this is trivial since the x-position and width w of the messages or elements, respectively, are identical. Thus, this has to be solved for the y-transformation as given by the Equations (2), (3), (9), and (10).

The function $t(x,y)$ from Equation (4) performs a bitwise flip operation of the value x given in *Little-endian* encoding on its $\log_2 y$ bits. This satisfies the properties of function t in Equation (7) under the assumptions in Equation (5) and (6).

Two elements in a bin intersect if

$$\neg((y_m + h_m \leq y_{\tilde{m}}) \vee (y_m \geq y_{\tilde{m}} + h_{\tilde{m}})) \tag{11}$$

or by applying *De Morgan's law*

$$(y_m + h_m > y_{\tilde{m}}) \wedge (y_m < y_{\tilde{m}} + h_{\tilde{m}}). \tag{12}$$

Applying Equation (2) to Equation (12) results in

$$y_m - y_{\tilde{m}} > -\frac{H}{r_m} \text{ and } y_m - y_{\tilde{m}} > \frac{H}{r_{\tilde{m}}}. \tag{13}$$

A further transformation with Equation (9) results in the following equations:

$$r_m \cdot t(b_{\tilde{m}}, r_{\tilde{m}}) < r_{\tilde{m}} \cdot (t(b_m, r_m) + 1) \tag{14}$$

$$r_{\tilde{m}} \cdot t(b_m, r_m) < r_m \cdot (t(b_{\tilde{m}}, r_{\tilde{m}}) + 1) \tag{15}$$

If Equation (14) and (15) are both satisfied, the two elements are intersecting. On the other hand, if either Equation (14) or (15) is violated, the two elements are not intersecting.

($\Rightarrow$) If the slot packing for two messages m and $\tilde{m}$ is conflicting, the corresponding elements in the bin packing intersect:

Without loss of generality, it is assumed that $r_m \leq r_{\tilde{m}}$. Two messages are conflicting in the slot packing if

$$\exists n \in \mathbb{N}_0 : b_m + r_m \cdot n = b_{\tilde{m}}. \tag{16}$$

This means, that the $\log_2 r_m$ least significant bits of b_m and $b_{\tilde{m}}$ are equal and, thus,

$$t(b_{\tilde{m}}, r_{\tilde{m}}) = \frac{r_{\tilde{m}}}{r_m}(t(b_m, r_m) + a) \tag{17}$$

holds with

$$0 \leq a \leq 1 - \frac{r_m}{r_{\tilde{m}}} < 1. \tag{18}$$

Here, a is the potential remainder by applying a shift of $\log_2 \frac{r_{\tilde{m}}}{r_m}$ bits. The upper bound 1 holds due to the problem-specific constraint $r_m \geq 1$ and the given assumption $r_{\tilde{m}} \geq r_m$.

Equation (17) is transformed to the following two equations:

$$r_m \cdot t(b_{\tilde{m}}, r_{\tilde{m}}) = r_{\tilde{m}} \cdot (t(b_m, r_m) + a) \tag{19}$$

$$r_{\tilde{m}} \cdot t(b_m, r_m) = r_m \cdot (t(b_{\tilde{m}}, r_{\tilde{m}}) - a \cdot \tfrac{r_{\tilde{m}}}{r_m}) \tag{20}$$

Given Equation (19) with $a < 1$, Equation (14) holds. At the same time, Equation (15) holds due to Equation (20) and $a \geq 0$. Thus, the elements intersect as required.

($\Leftarrow$) If the slot packing for two messages m and $\tilde{m}$ is not conflicting, the corresponding elements in the bin packing do not intersect:

Without loss of generality, it is assumed that $r_m \leq r_{\tilde{m}}$. Two messages are not conflicting in the slot packing if

$$\forall n \in \mathbb{N}_0 : b_m + r_m \cdot n \neq b_{\tilde{m}}. \tag{21}$$

This means, that the $\log_2 r_m$ least significant bits of b_m and $b_{\tilde{m}}$ are not equal and, thus, either

$$t(b_{\tilde{m}}, r_{\tilde{m}}) \leq \frac{r_{\tilde{m}}}{r_m}(t(b_m, r_m) - 1 + a) \tag{22}$$

or

$$t(b_{\tilde{m}}, r_{\tilde{m}}) \geq \frac{r_{\tilde{m}}}{r_m}(t(b_m, r_m) + 1 + a) \tag{23}$$

hold, both with a in the bounds from Equation (18). From Equation (22) it follows

$$r_{\tilde{m}} \cdot t(b_m, r_m) \geq r_m \cdot (\tfrac{r_{\tilde{m}}}{r_m}(1-a) + t(b_{\tilde{m}}, r_{\tilde{m}})) \tag{24}$$

that violates Equation (15) due to $\frac{r_{\tilde{m}}}{r_m}(1-a) \geq 1$ that holds since $a \leq 1 - \frac{r_m}{r_{\tilde{m}}}$ as stated in Equation (18). Equation (23) equals

$$r_m \cdot t(b_{\tilde{m}}, r_{\tilde{m}}) \geq r_{\tilde{m}} \cdot (t(b_m, r_m) + 1 + a) \tag{25}$$

that violates Equation (14) due to $a \geq 0$. Thus, either Equation (14) or (15) is violated and the elements do not intersect as required.

This proves Equation (2) and Equation (9). Equation (3) holds due to Equation (2) and Equation (10) holds due to Equation (9) with the inverse properties of the t function given in Equation (7).

3.2 Bin Packing

The task of a two-dimensional bin packing problem is to pack rectangular elements of different sizes defined by an individual width w and height h into a minimal number of rectangular bins without any intersection. Each bin has the fixed length W and height H. The transformation of the slot packing into a special two-dimensional bin packing problem determines a rectangular element m with the width w_m and height h_m for each message $m \in M$. The width of the bin equals the payload of a slot W and the height is the number of cycles which is 64 for FlexRay.

In contrast to the common two-dimensional bin packing, the transformed problem from the previous section contains two constraints:

(1) Each element $m \in M$ has a height h_m that is a power of two, i.e., 2^n with $n \in \mathbb{N}_0$ and the bin height is at least the maximal height of all elements.
(2) Each element $m \in M$ can be placed everywhere on the x-axis but only on a multiple of its height on the y-axis, i.e., $y_m = l \cdot h_m$ with the level $l \in \{0, ..., \frac{H}{h_m} - 1\}$.

This section introduces two optimization approaches for this specially constrained two-dimensional bin packing problem. The first approach is a fast greedy heuristic, the second is an efficient encoding as an ILP that allows to find the optimal solution.

3.2.1 Fast Greedy Heuristic

The presented bin packing problem can be solved by a fast greedy heuristic comparable to the approach presented in [17]. This heuristic is outlined in Alg. 1.

Algorithm 1 Fast greedy heuristic for bin packing.

```
1:  S = {} //set of bins
2:  for m ∈ M do
3:      for s ∈ S do
4:          if place(m,s) then
5:              continue with next m
6:          end if
7:      end for
8:      create new s and add it to S
9:      place(m,s)
10: end for
```

The algorithm starts with an empty set of bins S. Each element $m \in M$ is tried to be placed subsequently in a bin $s \in S$. Here, the function $place(m,s)$ is problem dependent and returns *true* if the placing is successful, and *false* otherwise. If an element is not placed in any of the allocated bins in S, a new bin s is allocated, added to S, and the element m is placed into this new empty bin.

Applied to the proposed special bin packing problem, the order of M influences the quality of the results. The elements in M are ordered first by their height h_m such that high elements are ordered to the front. The second criterion for the ordering of elements of the same height is the width w_m such that wide elements are ordered to the front. The function $place(m,s)$ tries to place each element m the most left void space in the bin s considering the individual level constraints of the proposed bin packing problem. This strategy tends to avoid the waste of void space of the bins. The complexity of this heuristic is polynomial.

3.2.2 Integer Linear Programming

Basic ILP. Solving a general two-dimensional bin packing problem with an ILP results in a high number of variables and constraints [5, 13]. However, the fact that each element has a height of a power of two and can only be placed on levels depending on its height can be exploited to deduce a compact and efficient ILP formulation with relatively few variables and constraints. The ILP formulation relies on the following binary variables:

- $\mathbf{m_{s,l}}$ - element m is placed at level l in bin s
- $\mathbf{s}$ - bin s is allocated (used)

The ILP is formulated as follows:

$$min \sum_{s \in S} \mathbf{s} \tag{26}$$

$$\forall\, m \in M : \sum_{s \in S} \sum_{l=0}^{\frac{H}{h_m}-1} \mathbf{m_{s,l}} = 1 \tag{27}$$

$$\forall\, s \in S, \{y = 0, ..., H-1\} : \sum_{m \in M} w_m \cdot \mathbf{m}_{\mathbf{s},\left\lfloor \frac{\mathbf{y}}{\mathbf{h_m}} \right\rfloor} \leq W \tag{28}$$

$$\forall\, m \in M, s \in S, \{l = 0, ..., \tfrac{H}{h_m} - 1\} : \mathbf{s} - \mathbf{m_{s,l}} \geq 0 \tag{29}$$

The objective function (26) of the ILP minimizes the number of allocated bins. Here, the set S has to contain a minimal number of bins that are necessary to solve the problem. This number is deduced from the presented fast heuristic approach. The constraints (27) state that each element m is placed in exactly one bin s at the specific level l. By adding the widths of the elements and restricting this sum by the width of a bin, the constraints (28) ensure that the size of each bin is not exceeded. The constraints (29) state that a bin s has to be allocated if at least one element m is placed in it.

Solving this ILP provides a bin s and level l for each element m (exactly one variable $\mathbf{m_{s,l}}$ is true). Placing the elements starting from the highest element to the most left void space in the bin s at the level l results in a feasible solution of the bin packing problem. This holds since the individual level constraints induce that each element that is sorted to the most left void space has at most one contact element on its left, cf. Figure 5(b). Thus, the constraints (28) are sufficient to determine a feasible bin packing.

Though this is a very efficient ILP encoding in terms of the number of variables, one has to keep in mind that the complexity of an ILP is exponential in general. Moreover, in contrast to the heuristic approach, the presented ILP cannot be used incrementally, i.e., an already allocated bin cannot be filled with additional elements without moving the old elements.

Enhanced ILP. The stated ILP can be further improved by reducing the search space by applying domain-specific knowledge. First, the set of S is reduced by one bin, simplifying the ILP by omitting several variables and constraints. In case there exists no feasible solution of this simplified ILP, there also exists no feasible bin packing for $|S|-1$ bins and, thus, the reference solution obtained by the heuristic is already optimal.

Furthermore, a lower bound for the objective is deduced by domain-specific knowledge to improve the runtime of the ILP. This lower bound is calculated as follows:

$$lb(M) = \frac{\sum_{m \in M} w_m \cdot h_m}{W \cdot H}. \tag{30}$$

The additional constraint

$$\sum_{s \in S} \mathbf{s} \geq \lceil lb(M) \rceil \tag{31}$$

sets the lower bound for the objective function. This constraint improves the runtime of the ILP: If the optimal solution is reached and equals the lower bound, the optimization process terminates immediately.

Regarding the problem of bin packing, a so-called *symmetry breaking* is applicable to reduce the search space. Consider the following one dimensional bin packing example: Given two elements m_1 and m_2 and two bins s_1 and s_2, there exist four possible distributions of the elements to the bins:

1. m_1, m_2 in s_1
2. m_1 in s_1 and m_2 in s_2
3. m_1, m_2 in s_2
4. m_1 in s_2 and m_2 in s_1

Since all bins have the same size and their order is negligible, there exists a symmetry between s_1 and s_2 regarding (1),(3) and (2),(4): Either (1) or (3) state that both elements are in the same bin, and correspondingly (2) or (4) state the both elements are in different bins. If the element m_2 is prohibited to be packed to bin s_2, (2) and (3) become invalid and the symmetry is broken. Thus, the search space is effectively reduced.

In order to generalize the symmetry breaking for the presented ILP formulation for the two-dimensional bin packing problem, two order functions for the elements and bins are used:

$$o: S \to \mathbb{N} \tag{32}$$

$$o: M \to \mathbb{N} \tag{33}$$

These functions assign to each element and bin, respectively, a unique integer value starting from 1 to $|M|$ and $|S|$, respectively. Given these functions, the symmetry breaking between bins is performed by adding the following constraints to the ILP formulation:

$$\begin{aligned} &\forall m \in M, s, s' \in S (s \neq s'), l = \{0, ..., \tfrac{H}{h_m} - 1\} \\ &\text{with } o(m) = o(s'), o(s') < o(s) : \mathbf{m_{s,l}} = 0 \end{aligned} \tag{34}$$

This ensures that an element is not allowed to be placed in a bin with a higher order.

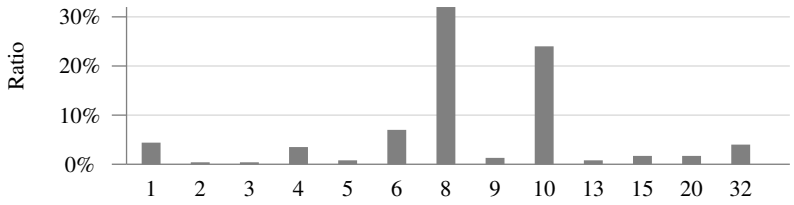
Additionally, the two-dimensional bin packing leads to a possible horizontal symmetry through the middle of each bin. The symmetry breaking inside a bin is performed by adding the following constraints:

$$\begin{aligned} &\forall m \in M, s \in S, l = \{0, ..., \left\lceil \tfrac{H}{2 \cdot h_m} - 1 \right\rceil\} \\ &\text{with } o(m) = o(s) : \mathbf{m_{s,l}} = 0 \end{aligned} \tag{35}$$

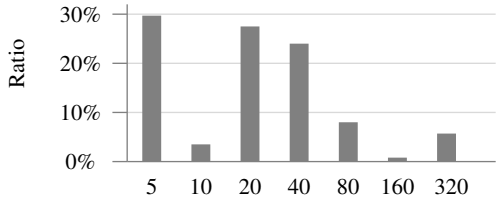
This ensures that all elements with the same order value as the order value of a bin can only be placed in the lower half of this bin. Note that each symmetry breaking effectively accelerates the ILP solving without excluding optimal solutions.

4 Case Study

A real-world example consisting of a FlexRay bus with 8 ECUs and an overall number of 220 messages is carried out as a case study to show the applicability of the proposed methods. The distribution of the sizes and periods of the messages is illustrated in Figure 6. The messages are highly heterogeneous in terms of their period and size. The parameters of the FlexRay bus are predefined such that the static segment consists of 62 slots with each slot carrying a payload of 42 bytes. Effectively, only 41 bytes are used since one byte is reserved for the update bits. The duration of the communication cycle is 5 milliseconds. The experiments were



(a) Distribution of the message sizes in bytes.



(b) Distribution of the message periods in milliseconds.

Fig. 6 Distributions for message sizes and periods of the real-world case study.

carried out on an Intel Pentium 4 3.20 GHz machine with 512 MB RAM. The ILP solver for the bin packing was the CPLEX solver in the version 10.5 [9]. Currently, the only available automatic scheduling approach compliant with the AUTOSAR specification is the commercial tool TTX PLAN [22]. For the introduced case study, a reference solution obtained by TTX PLAN is available.

The time for the transformation into a bin packing problem and vice versa might be omitted since it is negligibly small (less than 1 millisecond). The results for the case study are given in Table 1. The runtimes for the heuristic and ILP approach for

Method	runtime [s]	slots
Heuristic	**0.065**	**27**
ILP* (CPLEX)	25.7	**27**
ILP (CPLEX)	0.080	**27**
TTX Plan	360	29

Table 1 Results for the case study.

the case study is a fraction of a second. The row ILP* in Table 1 shows the results that were obtained by the ILP approach without the presented enhancements. Here, the runtime is significantly higher with 25.7 seconds. In fact, the ILP enhancements are always advantageous since there arises no additional overhead. The heuristic and ILP approach both deliver solutions with 27 allocated slots. Moreover, the ILP approach proves that 27 allocated slots is the optimal solution.

The proposed algorithms were compared to a result obtained by the commercial available scheduling tool TTX PLAN [22] that is based on an undisclosed heuristic approach with a prospected polynomial scalability of the runtime. While the commercial tool returns a schedule with 29 allocated slots, the heuristic and the ILP improve this value by two slots, which is significant for a real-world application. These results show that the commercial tool delivers an inferior result in a comparatively large amount of time, in particular, in 6 minutes. The runtime of the commercial tool TTX PLAN and the presented approaches differs by four orders of magnitude. Since scheduling is typically just one of several tasks in a complete design flow, this enables a Design Space Exploration (DSE) [14] with reasonable runtimes using the proposed algorithms.

5 Summary

This chapter presented a scheduling optimization scheme for the static segment of the FlexRay bus in compliance with the AUTOSAR specification. First, the problem is transformed into a special two-dimensional bin packing problem using a proposed one-to-one transformation scheme. This constrained bin packing problem is solved either with a presented heuristic approach, delivering good results in a relatively small amount of time, or an introduced efficient ILP approach that delivers the optimal solution. The results of the case study show that the heuristic and ILP approach are superior to a commercial tool in runtime and quality. The scalability analysis studies the applicability of the proposed methods.

References

1. AUTOSAR: Specification of the FlexRay Interface Version 3.0.2 (2008). Http://www.autosar.org
2. Berwanger, J., Peteratzinger, M., Schedl, A.: FlexRay startet durch. FlexRay-Bordnetz für Fahrdynamik und Fahrerassistenzsysteme (in German). In: Elektronik Automotive: Sonderausgabe 7er BMW (2008). Available at http://www.elektroniknet.de/home/automotive/bmw-7/flexray-startet-durch/
3. Broy, J., Müller-Glaser, K.D.: The Impact of Time-triggered Communication in Automotive Embedded Systems. In: Proceedings of the International Symposium on Industrial Embedded Systems (SIES 2007), pp. 353–356 (2007)
4. CAN: Controller Area Network. Http://www.can.bosch.com/
5. Christofides, N., Hadjiconstantinou, E.: An Exact Algorithm for Orthogonal 2-D Cutting Problems using Guillotine Cuts. European Journal of Operational Research **83**(1), 21–38 (1995)
6. FlexRay Consortium: FlexRay Communications Systems - Protocol Specification Version 2.1 Rev. A. (2005). Http://www.flexray.com
7. Ding, S., Murakami, N., Tomiyama, H., Takada, H.: A GA-based Scheduling Method for FlexRay Systems. In: Proceedings of the International Conference on Embedded Software (EMSOFT 2005), pp. 110–113 (2005)
8. Grenier, M., Havet, L., Navet, N.: Configuring the Communication on FlexRay: The Case of the Static Segment. In: Proceedings of the 4th European Congress on Embedded Real Time Software (ERTS 2008) (2008)
9. ILOG: CPLEX. Http://www.ilog.com/products/cplex/, Version 10.5
10. Kötz, J., Poledna, S.: Making FlexRay a Reality in a Premium Car. In: Proceedings of the Society of Automotive Engineers International 2008 (SAE 2008) (2008)
11. Li, W., Di Natale, M., Zheng, W., Giusto, P., Sangiovanni-Vincentelli, A., Seshia, S.: Optimizations of an Application-Level Protocol for Enhanced Dependability in FlexRay. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2009), pp. 1076–1081 (2009)
12. Lodi, A., Martello, S., Vigo, D.: Recent Advances on Two-dimensional Bin Packing Problems. Discrete Applied Mathematics **123**(1-3), 379–396 (2002)
13. Lodi, A., Martello, S., Vigo, D.: Models and Bounds for Two-Dimensional Level Packing Problems. Journal of Combinatorial Optimization **8**(3), 363–379 (2004)
14. Lukasiewycz, M., Glaß, M., Milbredt, P., Teich, J.: FlexRay Schedule Optimization of the Static Segment. In: Proceedings of the 7th IEEE/ACM International Conference on Hardware/software Codesign and System Synthesis (CODES+ISSS 2009), pp. 363–372 (2009)
15. Pop, T., Pop, P., Eles, P., Peng, Z.: Bus Access Optimisation For FlexRay-based Distributed Embedded Systems. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2007), pp. 51–56 (2007)
16. Pop, T., Pop, P., Eles, P., Peng, Z., Andrei, A.: Timing Analysis of the FlexRay Communication Protocol. In: Proceedings of the 18th Euromicro Conference on Real-Time Systems (ERTS 2006), pp. 203–216 (2006)
17. Puchinger, J., Raidl, G.R.: Models and Algorithms for Three-stage Two-dimensional Bin Packing. European Journal of Operational Research **127**(3), 1304–1327 (2007)
18. Saket, R., Navet, N.: Frame Packing Algorithms for Automotive Applications. Journal of Embedded Computing **2**(1), 93–102 (2006)
19. Sandstrom, K., Norstom, C., Ahlmark, M.: Frame Packing in Real-Time Communication. In: Proceedings of the Seventh International Conference on Real-Time Computing Systems and Applications (RTCSA 2000), pp. 399–403 (2000)
20. Schedl, A.: Goals and Architecture of FlexRay at BMW. In: Slides presented at the Vector FlexRay Symposium (2007). Available at https://www.vector-worldwide.com/
21. Schmidt, K., Guran Schmidt, E.: Message Scheduling for the FlexRay Protocol: The Static Segment. IEEE Transactions on Vehicular Technology **58**(5), 2170–2179 (2009)
22. TTTech: TTX Plan. Http://www.tttech-automotive.de/

23. Vector: DaVinci Network Designer FlexRay. Http://www.vector.com/
24. Zeng, H., Zheng, W., Di Natale, M., Ghosal, A., Giusto, P., Sangiovanni-Vincentelli, A.: Scheduling the FlexRay Bus Using Optimization Techniques. In: Proceedings of the 46th Conference on Design Automation (DAC 2009), pp. 874–877 (2009)