Decentralized Diagnosis of Permanent Faults in Automotive E/E Architectures

Peter Waszecki
TUM CREATE, Singapore
peter.waszecki@tum-create.edu.sg

Martin Lukasiewycz
TUM CREATE, Singapore
martin.lukasiewycz@tum-create.edu.sg

Samarjit Chakraborty
TU Munich, Germany
samarjit@tum.de

***Abstract*—This paper presents a novel decentralized approach for the diagnosis of permanent faults in automotive Electrical and Electronic (E/E) architectures. Both, the safety-critical real-time requirements and the distributed nature of these systems make fault tolerance in general and *fault diagnosis* in particular a crucial and challenging issue. At the same time, high unit numbers in manufacturing add cost efficiency as an important criterion during system design, which is conflicting with the use of often expensive explicit fault diagnosis hardware. To address these challenges, we propose a diagnosis framework that consists of two stages. In the first *diagnosis determination* stage, potential fault scenarios, such as defective Electronic Control Units (ECUs), are investigated to obtain a set of diagnosis functions. Specific diagnosis functions are used for each component in the network at runtime to determine whether a certain fault scenario is present. In the second *diagnosis optimization* stage, an optimization of diagnosis functions is proposed to determine trade-offs between diagnosis times and the number of monitored message streams. Experimental results based on 100 synthetic test cases give evidence of the feasibility and efficiency of the presented framework. Finally, an automotive case study demonstrates the practicability and details of our fault diagnosis approach.**

I. Introduction

Today, modern cars often comprise more than 100 Electronic Control Units (ECUs) connected via several buses and gateways. In these automotive Electrical and Electronic (E/E) architectures, the growing use of software and hardware for safety-critical applications, like Advanced Driver Assistance Systems (ADAS), makes strict guarantees on reliability and fault tolerance inevitable. In case of a fault, the defective resource, like ECU, bus or gateway, must be identified as quickly as possible in order to apply appropriate countermeasures. These can range from restarting the affected tasks on other resources to a safe shutdown of the entire system. At the same time, given the high number of units in the automotive industry, costs for electronics are a crucial optimization criterion restricting the use of explicit fault tolerance strategies. The topic gains further in importance as the shrinking dimensions and decreasing supply voltages in modern integrated circuits lead to an increased susceptibility to faults [1]. Consequently, novel reliable and efficient *fault diagnosis*[1] methods for advanced fault tolerance approaches are becoming necessary. In the automotive area, however, a new diagnosis method will only be accepted if the integration effort is minimal and all real-time and safety guarantees are met in terms of correctness and diagnosis time.

As a matter of principle, the proposed fault diagnosis monitors and analyzes existing message streams in a distributed architecture in order to identify defective resources.

This work was financially supported in part by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

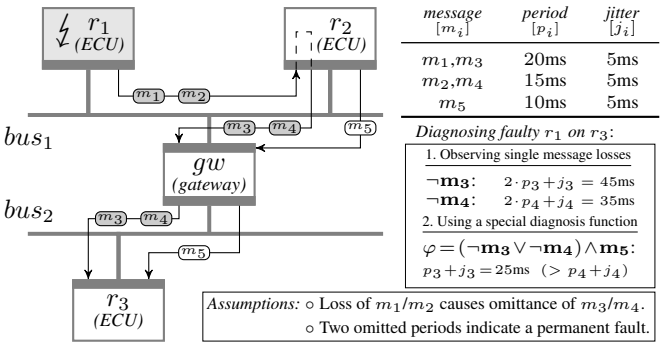


Fig. 1: Automotive E/E architecture consisting of three ECUs, r_1 to r_3, two buses and one gateway. ECU r_3 is an observing resource capable of diagnosing permanent faults ($\lightning$) by monitoring patterns of omitted ($\neg\mathbf{m_i}$) and received ($\mathbf{m_i}$) messages. Often, these *diagnosis functions* allow a reduction of the diagnosis time.

For this purpose, it uses explicit patterns of received and omitted messages, so-called *diagnosis functions* generated by the presented framework. The approach does not rely on a single observer, but rather each message-processing resource in the architecture, like an ECU or a gateway, has the capability of diagnosing the entire system or parts of it. Consequently, this decentralized approach removes the single point of failure and, thus, improves the reliability of the fault diagnosis itself. By using only existing system messages, the induced diagnosis overhead for each observing resource is kept minimal.

The general concept of the proposed fault diagnosis is illustrated in Fig. 1, showing a simplified automotive architecture with three ECUs, two buses and one gateway. In this example, ECU r_3 can diagnose a permanent fault at ECU r_1, by detecting the omittance of the messages m_3 or m_4 (which depend on m_1 and m_2, respectively) for two consecutive periods. By using explicit diagnosis functions which consider both omitted ($\neg\mathbf{m_i}$) and received ($\mathbf{m_i}$) messages, we are able to pinpoint the source of fault (in this case by excluding r_2) and reduce the diagnosis time. A more detailed explanation of this example is given in Section III.

Contributions of the paper. In this work, we present a novel approach towards the diagnosis of permanent faults in automotive E/E architectures based on the analysis of the monitored network traffic. Modern in-vehicle networks use different buses, such as Controller Area Network (CAN), Local Interconnect Network (LIN) or FlexRay, operating in

[1] In this work, the term *detection* mainly relates to the identification of the presence of a fault, while *diagnosis* describes the identification of its source (i.e., defective resource).

different domains and connected via gateways [2]. This can impede a message-based fault diagnosis especially when the observing and defective resources are attached to different buses, as in Fig. 1. By regarding the entire distributed system at *network-level*, including all buses and gateways (i.e., considering all system messages and their dependencies) our diagnosis extends the number of detectable resources and enables a low diagnosis time.

Moreover, the abandonment of a single diagnosis entity increases the overall system reliability and, hence, the safety-critical aspects of automotive E/E architectures. At the same time, by using only existing network communication, there is no need for special diagnostic messages which would involve additional computation overhead and possibly extend the diagnosis time. Such a decentralized and implicit fault diagnosis is especially (but not solely) beneficial in the cost-driven automotive industry where an early diagnosis of defective hardware is crucial, but each dedicated monitoring and testing device entails increased hardware costs due to a high number of units.

In particular, we propose a fault diagnosis definition which determines the effect of potential permanent faults on the existing message-based communication. For a given system architecture, we generate unique diagnosis functions for each possible fault scenario which later will be implemented on all suitable ECUs. In order to find trade-offs between diagnosis times (i.e., observation times necessary to detect particular fault scenarios) and the number of monitored message streams, we propose an optimization of the diagnosis functions.

In summary, our contribution is twofold:

1) Determination of diagnosis functions for an implicit and decentralized real-time fault diagnosis in distributed systems.
2) Optimization of diagnosis functions in terms of diagnosis times and number of message streams.

We address fault diagnosis as a self-contained topic and the system behavior subsequent to it is not in the focus of this paper. However, our approach is not limited to a specific fault recovery method and, hence, applicable to any fault tolerance strategy in a distributed system which considers a fail-silent behavior of its resources. To the best of our knowledge, this is the first approach towards an implicit and decentralized permanent fault diagnosis for automotive E/E architectures, including a trade-off analysis of the real-time and performance aspects.

II. Related Work

For safety-critical automotive systems it is crucial to diagnose faults early and provide appropriate countermeasures. As consequence, there exists a variety of work in the area of reliability-aware system design. The approaches comprise, on the one hand, component-oriented strategies which try to enhance reliability by the proper selection of system components, taking into account constraints like their area consumption and performance, e.g., [3, 4]. On the other hand, there are task-oriented methods where a better reliability is achieved through specific process allocation, additional tasks or schedule optimization, e.g., [5, 6]. Many of these approaches include differently elaborated fault diagnosis strategies.

For instance, in [7] the use of imperfect software fault detectors is investigated. The paper presents a design optimization which considers both detectable and undetectable faults, assuming that, although imperfect detectors might not prevent all silent data corruptions, they may provide more capacity for fault tolerance methods due to a lower resource utilization. An enhanced reliability technique for transient faults is presented in [8] (considering single faults) and [9] (considering multiple faults) where fault management requirements for task-graph elements can be specified at design time together with different hardware reliability features. The subsequent design space exploration is applied according to the extended application and architecture graphs. Regarding the actual fault detection methods, the three works above consider well-known approaches like task duplication with concurrent error detection units or other checking and voting mechanisms.

By contrast, our work presents a diagnosis of permanent faults which analyzes the message-based communication in order to identify defective resources. In this context, a detection algorithm using corrupted messages to discriminate functioning from defective resources is introduced in [10, 11]. It is designed as an add-on protocol for a time-triggered communication platform and to some extent omitted messages are also used to detect faults. However, while these detection methods are applied at bus-level, our approach is considering the entire system at network-level, including different buses and gateways. Furthermore, the detection time is bound by the sending slot cycles of the Time Division Multiple Access (TDMA)-architecture, while our approach supports event-triggered communication as well.

The idea of monitoring conjoint event arrivals has been used in [12] in the context of mixed-criticality systems. In this work the activation of a group of low priority tasks is modeled through the sums of their workload arrival functions to bound their influence on high priority tasks. While the concept of combined workload arrival functions is similar to our approach of summing up message arrival curves, it is used for integrating applications aiming at increasing the system utilization. On the contrary, in our work the arrival curves are used to determine minimal observation times for the diagnosis of permanent faults.

A distributed fault diagnosis for automotive architectures is presented in [13]. Here, single nodes send the outcome of a local fault diagnosis to neighboring nodes in order to achieve a global fault diagnosis. Although the paper highlight the high communication costs and delays of a centralized diagnosis, the approach is still relying on diagnosis status broadcasts and diagnosis times are not explicitly discussed. Our method uses solely existing messages and is based on a decentralized principle to reduce system overhead as well as remove a possible single point of failure. It also puts an emphasis on a minimal diagnosis time.

In [14] the authors present an on-line/off-line diagnosis framework motivated by the automotive domain. The approach uses propositional logic to describe the relationship between specific system properties and potential faults. However, in contrast to our work which provides a concrete massage-based detection method, the presented framework is a purely formal specification depending on user-defined tests and monitors to produce a valid diagnosis outcome.

Finally, fault diagnosis has been also addressed in the area of Discrete Event Systems (DESs), for instance in [15–17]. Such diagnosis methods often apply a model-oriented approach using logical models like Petri nets and timed automata. Although we use a graph-based system description

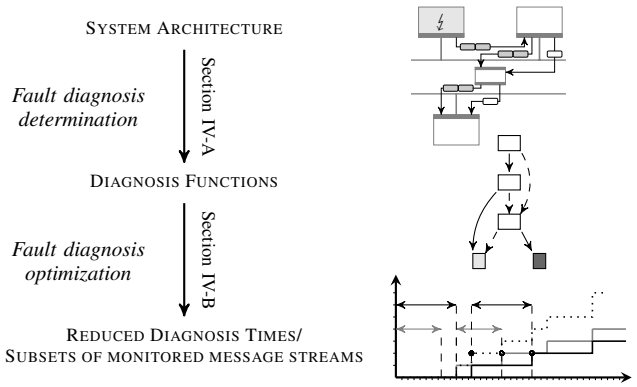


Fig. 2: The presented fault diagnosis framework includes two main parts. At design time, a given system specification is analyzed in order to generate diagnosis functions. In a second step, these functions are used to determine and optimize diagnosis times for permanent faults.

to determine diagnosis functions, at runtime the fault diagnosis is operating model-free by monitoring the network traffic to keep the computational and memory overhead low. Therefore, model-oriented methods can be hardly applied or compared in our work.

III. Framework

In this section, we define the permanent fault diagnosis problem and outline our diagnosis framework.

A. Problem Definition

As shown in Fig. 1, we regard automotive E/E architectures as distributed systems where the communication between ECUs is message-based and can comprise several different gateway-connected buses. Each message stream[2] m_i is defined by its nominal period p_i and a jitter j_i which depicts the variation of the actual period from the nominal period caused by task delays. For the proposed fault diagnosis, message streams are analyzed in order to identify permanent faults and distinguish them from transient faults.

Fault model. Faults in electronic components originate from physical phenomena and usually manifest themselves as bit flips. The cause of a fault can be both natural (e.g., a temporary change in the environmental conditions) and platform-based (e.g., unstable or marginal hardware) where the first case leads to transient faults and the second case results in intermittent and/or permanent faults. In our approach, intermittent faults are considered severe as they might cause the omittance of several messages and, hence, fall into the category of permanent faults.

In general, we assume a fail-silent behavior where a fault will result in an error leading to a transmission failure of messages. This might be the effect of a total breakdown of a resource. However, it is possible that a fault will not automatically lead to a message loss but, for example, will entail a message corruption or other erroneous behavior. To guarantee a complete and correct fault diagnosis, we assume that in such cases an internal fault detection mechanism prevents a resource from sending faulty messages. Although, this could require an extension of the current ECU functionality, the gain would be an improved and simplified system-wide fault diagnosis. In this manner, our approach only needs to consider existing communication without the necessity for special diagnostic messages. Nevertheless, a system that already uses diagnostic messages can further benefit from the proposed method by a reduction of the diagnosis time.

In particular, we discriminate between a functioning resource, which exhibits transient faults only, and a defective resource, which is affected by permanent faults, on the basis of the number of consecutive message losses in a message stream. We assume that the loss of a single message indicates any fault (also transient) while several consecutively omitted messages in one message stream indicate a permanent fault. However, depending on its specific manifestation within the system, a transient fault might also cause more than one message loss such that each additional omitted message increases the certainty that a permanent fault has been diagnosed. Thus, in a trivial case, when regarding merely one single message stream, the minimum fault diagnosis time[3] is $N \cdot p_i + j_i$ with $N \geq 2$. This means, for $N = 2$ one needs to detect an interruption in the message stream m_i for at least two periods p_i (plus the jitter j_i) in order to conclude that a permanent fault occurred at the corresponding resource.

Motivating example. The automotive E/E architecture in Fig. 1 illustrates the general concept of our fault diagnosis. In this example, ECU r_1 is periodically sending messages m_1 and m_2 to ECU r_2 where they are processed and forwarded to ECU r_3 as messages m_3 and m_4, respectively. In this way, there is a dependency between m_1 and m_3 as well as m_2 and m_4, such that when the transmission of the former message fails the latter message will be omitted as well. Furthermore, r_2 sends another periodic message to r_3, namely m_5. As the ECUs r_1 and r_2 use a different bus than r_3, the communication between them is routed via a gateway.

Based on the periods and jitters of the messages listed in the table in Fig. 1, we are able to diagnose faulty ECUs and minimize the diagnosis time not only on a common bus but also across the gateway borders. For instance, a permanent fault of r_1 will cause an interruption of the message streams m_1 and m_2 as well as m_3 and m_4, due to the aforementioned dependency (see shaded messages). Now, when r_3 monitors m_3 and m_4 separately, it needs at least 45 ms ($2 \cdot p_3 + j_3$) or 35 ms ($2 \cdot p_4 + j_4$), respectively, in order to decide if a permanent fault occurred. Besides, it still cannot decide if the fault stems from r_1 or r_2.

Here, a concurrent analysis of all message streams on bus_2 can reduce the diagnosis time and correctly pinpoint the defective ECU. With the help of a specific diagnosis function, $(\neg \mathbf{m_3} \vee \neg \mathbf{m_4}) \wedge \mathbf{m_5}$, we only need to wait until the first loss of the message m_3 as it has the longer period and m_4 must have been inevitably omitted as well during this time. Hence, the corresponding diagnosis time decreases to 25 ms, as $(p_3 + j_3) > (p_4 + j_4)$. At the same time, by detecting a received m_5, we can exclude r_2 as potential defective ECU.

B. Diagnosis Framework

Fig. 2 illustrates our diagnosis framework which consists of a fault diagnosis determination stage and an optimization stage. A brief outline of the framework is given below and will be discussed in detail in Section IV.

[2]Depending on the context, m_i denotes a *message* as well as the corresponding *message stream* (i.e., the periodic occurrence of the message).

[3]In the following, diagnosis time can be also denoted as *observation time*.

1) Fault diagnosis determination (*Section IV-A*). As foundation for the permanent fault diagnosis, our framework determines diagnosis functions of message observations that are derived from a given system specification and can be generated at design time. The system specification is a graph-based representation of a distributed system, where applications consisting of tasks and messages are mapped onto architecture resources, such as ECUs and buses. With the help of a graph-oriented search algorithm, the system is inspected for all potential sources of failure in order to define various *fault scenarios*. These are represented as sets which can consist of a single resource or a group of resources with a common potential source of fault, such as a shared power supply. For each fault scenario a unique pattern of received and omitted messages per *observing resource* is determined that allows to indicate the occurrence of a permanent fault. It is assumed, that each bus-attached resource can act as observing resource.

In order to efficiently integrate our fault diagnosis in a runtime system, the observation patterns are encoded as Binary Decision Diagrams (BDDs) from which, with the help of a *minimum cut* algorithm, *diagnosis functions* $\varphi_{r_o,s}(M_{r_o,s})$ are obtained. The observing resources r_o use these functions to diagnose a particular fault scenario s on the basis of received and omitted messages $m \in M_{r_o,s}$, and thereby identify the defective system components (Eq. (1)).

$\forall r_o \in R, s \in S:$

$$\varphi_{r_o,s}(M_{r_o,s}) = \begin{cases} 1, & s \text{ diagnosed by } r_o \\ 0, & s \text{ not diagnosed by } r_o \end{cases} \quad (1)$$

Within the presented decentralized approach, the diagnosis functions for one fault scenario s can vary for different observing resources r_o.

2) Fault diagnosis optimization (*Section IV-B*). In the second stage, our framework uses the diagnosis functions to investigate trade-offs between the fault diagnosis times and the number of message streams to be monitored. Here, the message periods and jitters are modeled based on event arrival curves. At the same time, the Boolean representation of $\varphi_{r_o,s}(M_{r_o,s})$ enables to determine message streams in $M_{r_o,s}$ which can be excluded from the diagnosis function *without* decreasing the diagnosis capability. While this exclusion is done without affecting the diagnosis itself, it may increase the corresponding diagnosis time.

This stage applies Integer Linear Programming (ILP) with the objective to minimize the observation time for each diagnosis function. The optimization problem is repeated multiple times where the cardinality of the set $M_{r_o,s}$ is successively reduced.

IV. Methodology

This section provides the detailed approach towards the determination of diagnosis functions as well as their subsequent optimization.

A. Fault Diagnosis Determination

Our framework uses a system specification that maps message-based task communications onto a system network consisting of distributed resources, as shown in the architecture in Fig. 1. For a better understanding of the presented method, this architecture will be used as an illustrative example.

The formal definitions in this section are mainly based on the sets and functions listed below.

- R set of all resources in the system, where $r_o \in R$ denotes an observing resource
- S set of all fault scenarios $s \in S$, each representing a set of defective resources (i.e., $s \subseteq R$)
- M set of all messages in the system, where $M_{r_o} \subseteq M$ contains all messages observable by r_o and $M_{r_o,s} \subseteq M$ contains messages observable by r_o for a particular fault scenario s
- $c: R \times M_{r_o} \to 2^R$ *cause function* determining resources which can cause the loss of a message m observable by r_o
- $o: R \times S \times M_{r_o} \to \{0,1\}$ *observation function* determining if a message m observable by r_o will be omitted (0) or detected (1) for a fault scenario s
- $b: R \times S \to 2^{M_{r_o}}$ *BDD function* constructing a BDD $b(r_o, s)$ for an observing resource r_o and a fault scenario s

System analysis. For each message $m \in M_{r_o}$, all resources $r \in R$ shall be identified which will cause a loss of m when affected by a permanent fault. Within our framework, *cause functions* $c(r_o, m)$ are used to assign these resources. In particular, the function performs a Depth-First Search (DFS) on the graph representing the system architecture, starting at the observing resource r_o and taking all data dependencies of the application into account. For instance, in the architecture in Fig. 1, the cause function $c(r_3, m_3)$ will result in the set $\{r_1, bus_1, r_2, gw, bus_2\}$.

Based on the cause functions, it shall be determined which messages will be received and omitted, respectively, when diagnosing a particular fault scenario $s \in S$. For this, we use *observation functions* $o(r_o, s, m)$, formally defined in Eq. (2) and explained below.

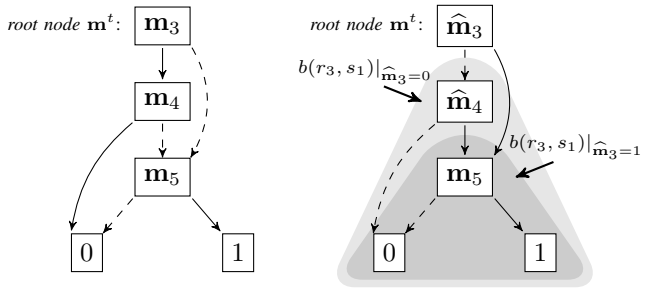
$\forall r_o \in R, s \in S, m \in M_{r_o}:$

$$o(r_o, s, m) = \begin{cases} 1, & \text{if } c(r_o, m) \cap s = \emptyset \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

An observation function indicates if any of the resources which can cause the loss of particular message m are also contained within a fault scenario s. It delivers 1 if a fault scenario s and a cause function $c(r_o, m)$ have no common intersection, else it returns 0. In other words, $o(r_o, s, m) = 1$ indicates that message m will not be affected by a permanent fault in s while $o(r_o, s, m) = 0$ points to a scenario causing the loss of m. For example, in the specification in Fig. 1, an observation function for the fault scenario $s_1 = \{r_1\}$ and $r_o = r_3$ will result in $o(r_3, s_1, m_3) = 0$, $o(r_3, s_1, m_4) = 0$ and $o(r_3, s_1, m_5) = 1$, as only m_3 and m_4 are affected by a defective ECU r_1.

Although, each observation function unambiguously defines received and omitted messages for a particular fault scenario and observing resource, it comprises all detectable messages from M_{r_o}. However, our goal is to find representations of diagnosis functions $\varphi_{r_o,s}$ which not only provide low diagnosis times but are also efficient in terms of the number of monitored message streams. For this, we use BDDs to encode the observations determined by $o(r_o, s, m)$.

A BDD depicts Boolean functions in the form of propositional directed acyclic graphs and enables an efficient manipu-



(a) original BDD (b) modified BDD

Fig. 3: Illustration of the original (a) and the modified (b) BDD $b(r_3, s_1)$ representing the diagnosis function for a faulty r_1 and an observing r_3 in the specification in Fig. 1. A solid line denotes a received message, a dashed line denotes an omitted message. Note, that the edges for auxiliary variables $\widehat{\mathbf{m}}$ are inverted.

lation and analysis of the underlying data structure [18]. Each node in a BDD represents a binary variable and the terminal nodes describe the function result 0 or 1. To obtain one (of possibly many) variable assignment for a positive result of the Boolean function, one has to follow a path from the root node to the 1-terminal and set each variable according to the outgoing edge (1 for a solid line, 0 for a dashed line).

Eq. (3) shows the definition of a BDD $b(r_o, s)$ for an observing resource r_o and an observed fault scenario s. Essentially, the BDD encodes observation differences between the analyzed scenario s and all other fault scenarios. In this context, messages are modeled one-to-one by the binary variables $\mathbf{m}$. More precisely, $\mathbf{m} = 1$ indicates a *positive observation* and $\mathbf{m} = 0$ indicates a *negative observation* representing a received and an omitted message, respectively. An illustration of $b(r_3, s_1)$ from the system specification in Fig. 1 is shown in Fig. 3a.

$\forall r_o \in R, s \in S:$

$$b(r_o, s) = \bigwedge_{\tilde{s} \in S, s \neq \tilde{s}} \Big(\big(\bigvee_{\substack{m \in M_{r_o} \\ o(r_o,s,m)=0 \\ o(r_o,\tilde{s},m)=1}} \neg\mathbf{m} \big) \vee \big(\bigvee_{\substack{m \in M_{r_o} \\ o(r_o,s,m)=1 \\ o(r_o,\tilde{s},m)=0}} \mathbf{m} \big) \Big) \quad (3)$$

Diagnosis function generation. The immediate way to retrieve a Boolean function from a BDD is to disjunctively combine all possible paths towards the 1-terminal. As this procedure may result in non-minimal functions, we use a *minimum cut* algorithm for the generation of diagnosis functions which excludes redundant paths within the BDD.

The recursive computation of the algorithm is defined in Eq. (4), where variable $\mathbf{m}^t$ denotes the root node of $b(r_o, s)$, and its two sub-BDDs on the 1-path and 0-path are referred to as $b(r_o, s)|_{\mathbf{m}^t=1}$ and $b(r_o, s)|_{\mathbf{m}^t=0}$, respectively (see shaded areas in Fig. 3b). Basically, redundant paths are removed by the set-theoretic difference in Eq. (4). It eliminates nodes in $MinCut(b(r_o, s)|_{\mathbf{m}^t=0})$ residing on paths towards the 0-terminal in $b(r_o, s)|_{\mathbf{m}^t=1}$.

$$\begin{aligned} &MinCut(b(r_o, s)) = \\ &\{C \cup \{\mathbf{m}^t\} | C \in MinCut(b(r_o, s)|_{\mathbf{m}^t=0}) \\ &\setminus Cut(b(r_o, s)|_{\mathbf{m}^t=1})\} \cup MinCut(b(r_o, s)|_{\mathbf{m}^t=1}) \end{aligned} \quad (4)$$

An essential prerequisite for the minimal cut algorithm is the *monotonicity* of the analyzed Boolean function, which, simply said, restricts the function to solely using non-negated variables. We can achieve monotonicity by substituting negative observations with positive auxiliary variables $\widehat{\mathbf{m}}$ and switch them back after the calculation is finished. The auxiliary variables invert the edges of their nodes, as can be seen in the modified BDD in Fig. 3b.

In principle, the minimal cut algorithm provides sets of Boolean variables representing *minterms* (i.e., conjunctions of variables) in a Disjunctive Normal Form (DNF) function. However, as our goal is to analyze the *sums* of message arrival curves, it is more beneficial to have diagnosis functions in a Conjunctive Normal Form (CNF). To circumvent this issue, we apply *De Morgan's laws*, which describe the transformation between disjunctive and conjunctive expressions via negations of terms. More precisely, we modify the algorithm to follow paths to the 0-terminal instead of the 1-terminal and interpret the resulting minterms as *maxterms* (i.e., disjunctions of variables).

Finally, the algorithm results in sets of nodes corresponding to maxterms in a CNF function which represents the desired diagnosis function $\varphi_{r_o,s}$. This is shown in Eq. (5), where $\Phi_i(M_i)$ defines a *maxterm* in $\varphi_{r_o,s}$ containing positive and negative message observations (Eq. (5b) and (5c)).

$\forall r_o \in R, s \in S:$

$$\varphi_{r_o,s}(M_{r_o,s}) = \bigwedge_i \Phi_i(M_i) \quad , \quad i \in \mathbb{N} \quad (5a)$$

with $$\Phi_i(M_i) = \big(\bigvee_{m \in M_i^-} \neg\mathbf{m} \big) \vee \big(\bigvee_{m \in M_i^+} \mathbf{m} \big) \quad (5b)$$

and $$M_i = M_i^- \cup M_i^+ \quad , \quad \bigcup_i M_i = M_{r_o,s} \quad (5c)$$

Returning to the example in Fig. 1, the diagnosis function for the observing resource r_3 and a faulty r_1 is $\varphi_{r_3,s_1} = (\neg\mathbf{m}_3 \vee \neg\mathbf{m}_4) \wedge \mathbf{m}_5$.

B. Fault Diagnosis Optimization

Our diagnosis functions are the foundation for an efficient decentralized diagnosis of permanent faults. Within the runtime system, each observing resource will be provided with a set of diagnosis functions for different fault scenarios. By monitoring the bus with respect to the message arrival times and updating the message observations within the diagnosis functions accordingly, an observing resources can reliably detect defective resources.

However, as the number of monitored message streams affects the computational performance of the system, we propose an ILP-based determination of the optimal diagnosis times for different message stream numbers. The diagnosis time is defined by the periods and jitters of the corresponding messages in $M_{r_o,s}$ and its determination and optimization is based on the sets and variables listed below.

$\mathbf{t}_{\varphi_{r_o,s}} \in \mathbb{R}$	variable defining the diagnosis time of $\varphi_{r_o,s}(M_{r_o,s})$
$\Phi \in \varphi_{r_o,s}$	maxterm (logical OR) of a set of variables $\mathbf{m}$ within $\varphi_{r_o,s}(M_{r_o,s})$
$N \in \mathbb{N}$	number of consecutively omitted messages in a message stream (i.e., omitted periods)
$R_F \subseteq R$	set of all faulty resources

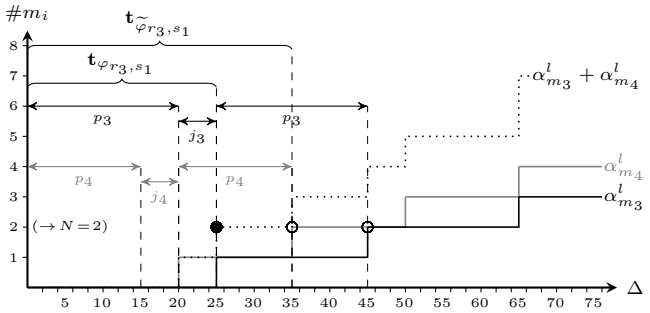


Fig. 4: Lower arrival curves (α^l_m) of the messages m_3 and m_4 from Fig. 1. They define the minimum number of messages (y-axis) observed during an arbitrary time interval Δ (x-axis). Diagnosis times for the single message observations (○) as well as the reduced diagnosis time (●), defined by the sum of the curves (dotted line) for $N=2$, are highlighted.

M^-, M^+	sets of omitted (M^-) and received (M^+) messages in $M_{r_o,s}$
α^l_m	lower arrival curve of a message $m \in M_{r_o,s}$

Diagnosis time determination. To determine the arrival times of messages, we use arrival curves based on an event model introduced earlier in [19]. As a matter of principle, this model represents the earliest and latest possible arrival times of a message as an upper (α^u_m) and a lower (α^l_m) arrival curve, respectively. Fig. 4 illustrates the arrivals of m_3 and m_4 from Fig. 1 where, for reasons of clarity, only lower arrival curves α^l_m are depicted. Considering, that for each message $m \in M_{r_o,s}$ the arrival curves are known through the corresponding periods and jitters, then the set M^+ contains all messages whose arrival times are above their lower arrival curve α^l_m and M^- contains all messages with arrival times below α^l_m. Intersections are excluded for reasons of consistency of the diagnosis (Eq. (6)).

$$M^- \cap M^+ = \emptyset \tag{6}$$

Now, the general diagnosis condition can be formulated, which implies that each faulty resource must result in omitted messages while, at the same time, there can be no message loss from a functioning resource (Eq. (7)).

$\forall r_o \in R, \breve{m} \in M^-, \forall \widehat{m} \in M^+ :$

$$\Big(c(r_o,\breve{m}) \cap R_F \neq \emptyset\Big) \wedge \Big(c(r_o,\widehat{m}) \cap R_F = \emptyset\Big) \tag{7}$$

Based on this general premise, we can formulate an ILP to determine the diagnosis time of a diagnosis function $\varphi_{r_o,s}(M_{r_o,s})$. The objective of the optimization problem is to minimize the variable $\mathbf{t}_{\varphi_{r_o,s}}$ (Eq. (8a)) and the corresponding constraints are defined in Eq. (8b) through (8g).

$\forall m \in M_{r_o,s},\ \mathbf{s}^k_m \in \{0,1\},\ \mathbf{t}_{\varphi_{r_o,s}} \in \mathbb{R} :$

$$\textit{minimize} \quad \mathbf{t}_{\varphi_{r_o,s}} \tag{8a}$$

subject to:

$$\forall \Phi \in \varphi_{r_o,s} : \sum_{m \in M^-} \alpha^l_m \geq N \tag{8b}$$

$$\forall \Phi \in \varphi_{r_o,s} : \sum_{m \in M^+_N} \alpha^l_m \geq 1 \tag{8c}$$

$$\alpha^l_m = \sum_{k=1} \mathbf{s}^k_m \tag{8d}$$

$$\mathbf{s}^k_m \geq \mathbf{s}^{k+1}_m \tag{8e}$$

$$\mathbf{t}_{\varphi_{r_o,s}} \geq (p_m + j_m) \cdot \mathbf{s}^1_m + \sum_{k=2}^{N} p_m \cdot \mathbf{s}^k_m \tag{8f}$$

$$\mathbf{t}_{\varphi_{r_o,s}} < (p_m + j_m) + p_m \cdot \mathbf{s}^1_m + \sum_{k=2}^{N} p_m \cdot \mathbf{s}^k_m \tag{8g}$$

Eq. (8b) and (8c) state that the sum of lower arrival curves must add up to at least N for omitted and at least 1 for received messages in order to identify the corresponding observations as negative and positive, respectively. For this, each maxterm Φ of the diagnosis function is regarded separately. In Eq. (8d) the lower arrival curve itself is defined as step function, where each step is described by a binary variable $\mathbf{s}^k_m$. It delivers 1 for the latest arrival time of the k^{th} instance of the message m and returns 0 otherwise. Eq. (8e) guarantees that all previous steps are set before a new step can be added to the arrival curve. The last two constraints, Eq. (8f) and (8g), define the boundaries for the diagnosis time.

The consideration of multiple message streams when monitoring omitted messages in the diagnosis function $\varphi_{r_o,s}$ is essential to guarantee a low diagnosis time. This is illustrated in Fig. 4, where the sum of the arrival curves $\alpha^l_{m_1}$ and $\alpha^l_{m_3}$ results in a reduced diagnosis time $\mathbf{t}_{\varphi_{r_3,s_1}} = 25$ ms for $N=2$.

Trade-off analysis. Finally, trade-offs between the diagnosis time and the number of monitored message streams shall be analyzed. Here, low values are preferable for both parameters in order to guarantee a fast diagnosis of a fault scenario and a low computational overhead. However, a reduced message set may result in an increase of the diagnosis time (Eq. (9)).

$$\mathbf{t}_{\widetilde{\varphi}_{r_o,s}(\widetilde{M}_{r_o,s})} \geq \mathbf{t}_{\varphi_{r_o,s}(M_{r_o,s})}\,, \quad \text{with} \quad \widetilde{M}_{r_o,s} \subseteq M_{r_o,s} \tag{9}$$

The trade-off analysis is performed by successively reducing the cardinality of the message set $M_{r_o,s}$ and repeating the diagnosis time optimization for each reduced set $\widetilde{M}_{r_o,s}$. For this, the ILP is extended with an additional binary variable $\mathbf{c}_m$ which determines whether a message m shall be considered for the optimization (1) or not (0). This is shown in Eq. (10), where all constraints defined in Eq. (8) hold.

$\forall k \in \{|M_{r_o,s}|, \ldots, 1\} :$

$\forall \widetilde{M}_{r_o,s} \subseteq M_{r_o,s},\ |\widetilde{M}_{r_o,s}| = k,\ \mathbf{c}_m, \mathbf{s}_m \in \{0,1\},\ \mathbf{t}_{\widetilde{\varphi}_{r_o,s}} \in \mathbb{R} :$

$$\textit{minimize} \quad \mathbf{t}_{\widetilde{\varphi}_{r_o,s}(\widetilde{M}_{r_o,s})} \tag{10a}$$

subject to:

$$\mathbf{c}_m - \mathbf{s}^1_m \geq 0 \tag{10b}$$

$$\sum_{m \in M_{r_o,s}} \mathbf{c}_m = |\widetilde{M}_{r_o,s}| \tag{10c}$$

This stage of our framework results in diagnosis functions with a reduced number of monitored message streams still capable of diagnosing the corresponding fault scenario. Regarding the example in Fig. 1, the optimization removes m_3 and, hence, results in a more compact diagnosis function $\widetilde{\varphi}_{r_3,s_1} = \neg \mathbf{m}_4 \wedge \mathbf{m}_5$ with a higher diagnosis time $\mathbf{t}_{\widetilde{\varphi}_{r_3,s_1}} = 35$ ms for $N=2$ (see Fig. 4).

V. Experimental Results

To provide a significant evaluation of the proposed approach, a selection of 100 synthetics test cases and a case study, both inspired by real automotive E/E architectures, are used. The test cases comprise systems with 2 to 23 ECUs and up to 4 gateway-connected buses where applications contain

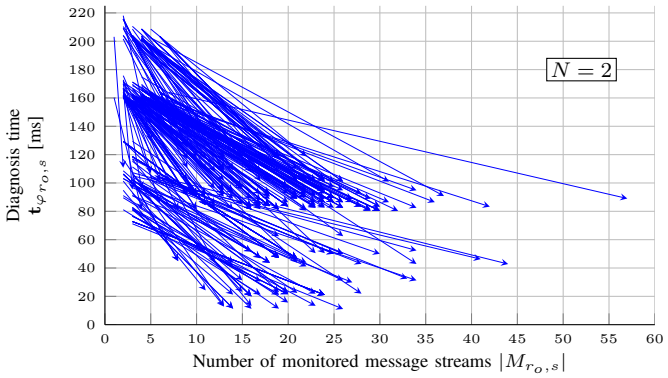


Fig. 5: Diagnosis time as function of the monitored message streams. Each vector represents the decrease from a maximum to a minimum diagnosis time with the two corresponding message numbers. The graph comprises results for $N = 2$ with a time reduction of more than 50 ms.

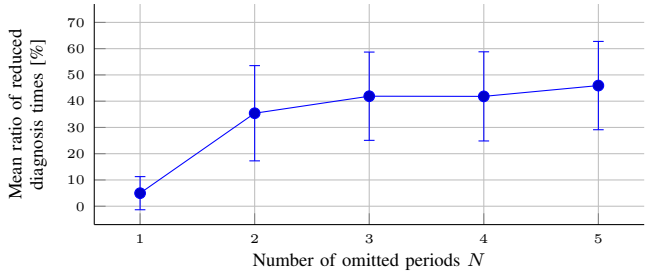


Fig. 6: Mean ratio (and standard deviation) of diagnosis functions with reduced diagnosis times for different numbers of omitted periods.

10 to 132 tasks and 7 to 78 messages. Overall, up to 24 fault scenarios per test case have been determined and each of them has been analyzed for all possible observing resources and for 1 to 5 omitted message periods. For the presented results we use fault scenarios consisting of one resource ($s_i = \{r_i\}$). All experiments including the system analysis, diagnosis function generation and optimization were carried out on an Intel Core i5 with 2.6 GHz and 8 GB RAM. For the ILP-based optimization, GUROBI version 5.6 was used as solver [20].

Synthetic test cases. To demonstrate the general feasibility of the presented method, Fig. 5 illustrates diagnosis times as function of monitored message streams for the diagnosis functions of all analyzed test cases. Results with a time reduction of less than 50 ms have been omitted to maintain legibility. As defined in Sec. IV-B, the time reduction represents the diagnosis time difference between the diagnosis function $\widetilde{\varphi}_{r_o,s}$ with a reduced number of considered messages and the original function $\varphi_{r_o,s}$ for a specific fault scenario.

The resulting graph indicates that diagnosis functions can be optimized either towards a minimal diagnosis time or a lower computational performance by monitoring fewer message streams. Hence, during system design, emphasis can be put on hard real-time requirements and energy/cost efficiency, respectively. The amount of diagnosis functions allowing a diagnosis time reduction depends on the system specification of a particular test case. Nevertheless, results with an irreducible diagnosis time provide important information about which messages can be removed from the diagnosis function without affecting the fault diagnosis itself. Such results would be depicted as horizontal vectors in the graph and have been left out for the sake of clarity.

Fig. 5 depicts the outcome where two message losses imply a permanent fault, i.e., $N = 2$. To investigate the influence of different numbers of omitted periods, Fig. 6 illustrates the mean ratios of reducible to irreducible diagnosis times for up to 5 omitted periods. While a diagnosis time improvement is very improbable when only one omitted period is regarded, the amount of reducible diagnosis functions strongly increases for two and more omitted periods. This results from the summation of message arrival curves where for a higher period number more messages might contribute to a potential diagnosis time reduction. For the same reason, an increased number of omitted periods does not necessarily proportionally increase the diagnosis time.

Overall, the outcome shows that considering a higher number of omitted periods might be an important design criterion, especially, as the certainty for detecting a permanent fault increases with the number of consecutively observed message losses, as mentioned in Section III.

The results in Fig. 5 and 6 are based on approximately 8000 distinct diagnosis functions for each N. Altogether, based on 100 test cases, our framework delivers a total of 40380 distinct fault observations from which in 12551 cases (31%) a reduction of the diagnosis time can be achieved by increasing the number of monitored messages.

As the fault diagnosis determination and optimization are performed offline and the target system only stores the diagnosis functions $\varphi_{r_o,s}(M_{r_o,s})$, a fast and memory-efficient computation for our framework is not directly required. Nevertheless, both the BDD-based diagnosis function determination and their ILP-based optimization exhibit an exponential complexity with the number of used variables, hence, it is worth to analyze the corresponding performance.

In this context, the upper graph in Fig. 7 shows the computation times of our framework divided in system analysis (i.e., the generation of diagnosis functions) (o-marks) and the diagnosis function optimization depicted (x-marks). The timeout for each test case is set to 3600. It can be seen, that even for systems beyond 20 ECUs both runtime values do not exceed 100 s and 600 s, respectively. Together with BDD sizes in a lower three-digit range (see center graph) this indicates a good scalability for larger systems.

The lower graph shows the maximum sizes of non-optimized diagnosis functions. Although, for some large systems the numbers are in the range 1000 variables, in the most cases our optimization can reduce this values by a factor of 10 or more without affecting the diagnosis time. It can be reduced even further when taking an increase of the diagnosis time into account. Consequently, an implementation of the diagnosis functions in today's automotive hardware is feasible.

Case study. Finally, a realistic case study shall demonstrate the usability of our method for an in-depth reliability analysis. It is based on a state-of-the-art automotive E/E architecture comprising 106 tasks and 61 messages which are distributed on 22 ECUs. The system uses 4 buses connected to a gateway (gw) where resources r_1 to r_4, r_5 to r_{12}, r_{13} to r_{15} and r_{16} to r_{22} are sharing a bus, respectively. Message periods can range from 5 to 80 ms and their jitters are determined according to the priority and delay calculation for event-triggered systems proposed in [21].

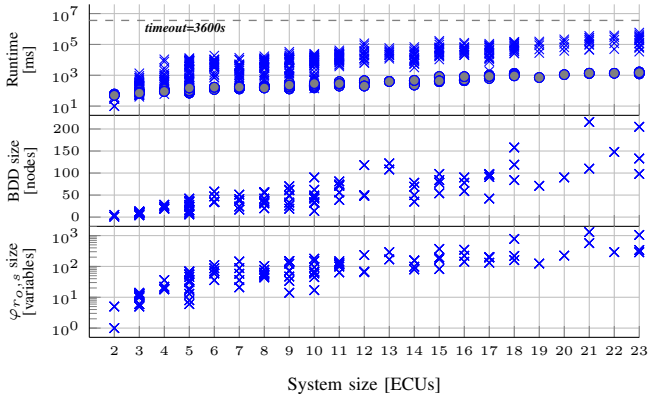


Fig. 7: Performance analysis of the presented approach. The figure illustrates how different system sizes (ECU number) influence the runtime (top), BDD size (middle) and the maximum size of the diagnosis functions (bottom).

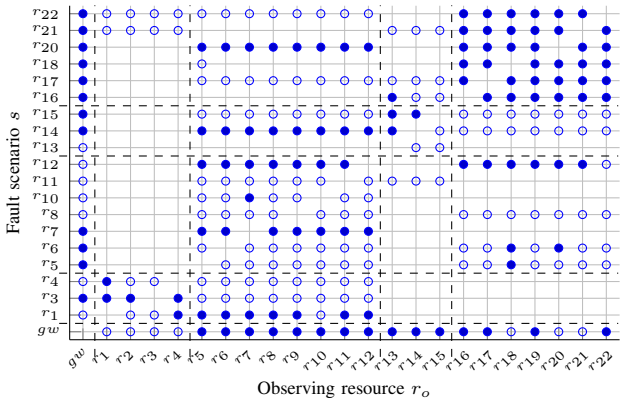


Fig. 8: Assignment of the determined diagnosis functions to different fault scenarios (y-axis) and the corresponding observing resources (x-axis) for our case study with $N = 3$. Diagnosis times can be reduced for 119 diagnosis functions (●) while the remaining 149 cases (○) are irreducible.

Based on the system specification, our framework generated 268 explicit diagnosis functions, from which 119 allow a diagnosis time reduction (●) and the remaining ones cannot be minimized (○). The resulting assignment of observing resources to fault scenarios is illustrated in Fig. 8. Here, each scenario consists of one potentially faulty ECU and the dashed lines delimit ECUs from different buses. For instance, the four groups lying on the diagonal represent cases where the observing resource and the fault scenario are attached to the same bus. Thus, we can clearly recognize the ability of our method to diagnose faulty resources (as well as optimize the diagnosis) beyond the limits of the own bus, namely, at network-level.

In summary, a system analysis as presented in the case study allows the system designer to implement the permanent fault diagnosis in a suitable and efficient way.

VI. Concluding Remarks

This paper presents a framework for a decentralized and implicit diagnosis of permanent faults in automotive E/E architectures. It comprises a fault diagnosis determination where diagnosis functions are generated based on BDD-encoded observations of fault scenarios. Additionally, during an optimization stage, the diagnosis functions are used to determine trade-offs between diagnosis times and the number of monitored message streams. The feasibility and efficiency of our approach is demonstrated based on an experimental evaluation of 100 synthetic test cases as well as a case study. **Future work.** An implementation of the permanent fault diagnosis on a research platform for automotive E/E architectures is part of the future work. The results of the implementation will, amongst other things, allow us to evaluate the diagnosis performance on a real embedded platform. Furthermore, the presented approach will be tested in terms of additional computational overhead and memory footprint.

References

[1] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," *Micro, IEEE*, vol. 25, no. 6, pp. 10–16, 2005.

[2] A. Sangiovanni-Vincentelli and M. Di Natale, "Embedded system design for automotive applications," *IEEE Computer*, vol. 40, no. 10, pp. 42–51, 2007.

[3] S. Tosun, N. Mansouri, E. Arvas, M. Kandemir, and Y. Xie, "Reliability-centric high-level synthesis," in *Proc. of DATE*, pp. 1258–1263, 2005.

[4] M. Glaß, M. Lukasiewycz, T. Streichert, C. Haubelt, and J. Teich, "Reliability-aware system synthesis," in *Proc. of DATE*, pp. 409–414, 2007.

[5] O. Héron, J. Guilhemsang, N. Ventroux, and A. Giulieri, "Analysis of on-line self-testing policies for real-time embedded multiprocessors in DSM technologies," in *Proc. of IOLTS*, pp. 49–55, 2010.

[6] Y. Xie, L. Li, M. Kandemir, N. Vijaykrishnan, and M. J. Irwin, "Reliability-aware co-synthesis for embedded systems," in *Proc. of ASAP*, pp. 41–50, 2004.

[7] J. Huang, K. Huang, A. Raabe, C. Buckl, and A. Knoll, "Towards fault-tolerant embedded systems with imperfect fault detection," in *Proc. of DAC*, pp. 188–196, 2012.

[8] C. Bolchini and A. Miele, "Reliability-driven system-level synthesis of embedded systems," in *Proc. of DFT*, pp. 35–43, 2010.

[9] C. Bolchini and A. Miele, "Reliability-driven system-level synthesis for mixed-critical embedded systems," *Computers, IEEE Transactions on*, vol. 62, no. 12, pp. 2489–2502, 2013.

[10] M. Serafini, N. Suri, J. Vinter, A. Ademaj, W. Brandstatter, F. Tagliabo, and J. Koch, "A tunable add-on diagnostic protocol for time-triggered systems," in *Proc. of DSN*, pp. 164–174, 2007.

[11] M. Serafini, A. Bondavalli, and N. Suri, "On-line diagnosis and recovery: On the choice and impact of tuning parameters," *Dependable and Secure Computing, IEEE Transactions on*, vol. 4, no. 4, pp. 295–312, 2007.

[12] M. Neukirchner, P. Axer, T. Michaels, and R. Ernst, "Monitoring of workload arrival functions for mixed-criticality systems," in *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, pp. 88–96, 2013.

[13] J. Luo, K. Choi, K. R. Pattipati, L. Qiao, and S. Chigusa, "Distributed fault diagnosis for networked, embedded automotive systems," in *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, pp. 1226–1232, 2006.

[14] S. Tripakis, "A combined on-line/off-line framework for black-box fault diagnosis," in *Runtime Verification*, pp. 152–167, 2009.

[15] I. Fliss and M. Tagina, "Multiple fault diagnosis of discrete event systems using petri nets," in *Communications, Computing and Control Applications (CCCA), 2011 International Conference on*, pp. 1–6, 2011.

[16] C. Mahulea, C. Seatzu, M. Cabasino, and M. Silva, "Fault diagnosis of discrete-event systems using continuous petri nets," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 42, pp. 970–984, July 2012.

[17] B. Suiphon, Z. Simeu-Abazi, and E. Gascard, "Implementation of a fault diagnosis method for timed discrete-event systems," in *Industrial Engineering and Systems Management (IESM), Proceedings of 2013 International Conference on*, pp. 1–8, 2013.

[18] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *Computers, IEEE Transactions on*, vol. 100, no. 8, pp. 677–691, 1986.

[19] S. Chakraborty, S. Kunzli, and L. Thiele, "A general framework for analysing system properties in platform-based embedded system designs," in *Proc. of DATE*, pp. 190–195, 2003.

[20] Gurobi Optimization, "Gurobi Optimizer version 5.6." http://www.gurobi.com/.

[21] M. Lukasiewycz, S. Steinhorst, and S. Chakraborty, "Priority assignment for event-triggered systems using mathematical programming," in *Proc. of DATE*, pp. 982–987, 2013.