Diagnosis-Aware System Design for Automotive E/E Architectures

Peter Waszecki[1], Florian Sagstetter[1], Martin Lukasiewycz[1], Samarjit Chakraborty[2]
[1] TUM CREATE, Singapore, Email: peter.waszecki@tum-create.edu.sg
[2] TU Munich, Germany, Email: samarjit@tum.de

***Abstract*—This paper proposes a schedule synthesis approach taking fault diagnosis and testability into account at design time. Over the last years, the amount of automotive software and hardware has been successively growing. As a consequence, the complexity of present-day Electrical and Electronic (E/E) architectures reached a state where current fault detection mechanisms are often not sufficient or computationally too expensive to guarantee a reliable system functionality. As a remedy, we propose a novel design methodology, optimizing a subsequent fault diagnosis in terms of the necessary detection time as well as the diagnostic resolution. Our approach is based on a time-triggered architecture and aims at a decentralized message-based fault diagnosis solution. In order to increase the system reliability, during schedule synthesis a modified and adapted message distribution is taken into account which additionally considers previously undiagnosable resources. While our approach might lead to a slightly increased bandwidth utilization, it clearly improves the overall diagnosis of faulty resources by a reduced detection time and an increased diagnostic resolution.**

I. Introduction and Related Work

Recent years have seen an unprecedented increase in vehicle functionality based on software and electronics. A large number of these functions, like drive-by-wire or advanced driver assistance systems, are highly safety-critical and, hence, have hard real-time requirements. At the same time, the continuous advancements in the underlying hardware architecture, e.g., shrinking geometries and reduced supply voltages, result in an increased occurrence of faults. By consequence, there is a demand for highly predictable Electrical and Electronic (E/E) architectures with reliable fault tolerance mechanisms as their integral part.

Currently, the automotive industry is slowly shifting from event-triggered towards time-triggered architectures, which innately provide a better predictability. On the other hand, many powerful fault tolerance strategies for distributed systems have been introduced over the past years, for some of which [1, 2] provide a good overview. In our work, these two aspects are regarded as part of a conjunct system design approach where the synthesis of a system schedule is explicitly considering an improved fault diagnosis[1] during system operation.

Moreover, the design of E/E architectures is based on a number of specialized tools which are often incompatible and lacking appropriate interfaces for a coherent development process and, thus, require time-consuming and cost-intensive testing, as described in [3]. As a matter of principle, our approach can be used to improve and reduce testing efforts during the entire development life cycle by providing a defined observation time[2] as well as an increased diagnostic resolution (i.e., a larger number of diagnosable resources).

Related work. As fault tolerance is inevitable for safety-critical systems, there exists a variety of work in the area of reliability-aware system synthesis. The approaches range from component-based strategies [4, 5], enhancing system reliability by the proper selection of hardware, to task-based methods [6–8], where the focus is on software fault detection, process mapping and scheduling optimization. Many of these approaches consider important diagnosis strategies, for instance, in [9] the optimal use of imperfect software fault detectors is analyzed and [10] is generating reliability-aware task graphs based on predefined fault management requirements.

By contrast, our work considers exclusively a message-based system communication as foundation for a subsequent permanent fault diagnosis and therefore attempts to optimize and extend the message scheduling. An example for a diagnosis algorithm using malicious messages to distinguish healthy from unhealthy nodes is given in [11, 12], which is designed as an add-on protocol for a time-triggered communication platform. As the authors do not alternate a given system specification and rely on existing messages, their diagnosis method might benefit from our work, which defines a maximal observation time and inserts auxiliary messages for undiagnosable resources.

Contributions of the paper. We propose a novel approach towards an improved fault diagnosis in a distributed network-oriented system, like an automotive E/E architecture. It is based on time-triggered architectures and uses a framework which was previously introduced in the context of schedule integration [13, 14]. Within our diagnosis-aware system design, this framework is extended for a use in the area of message-based fault diagnosis.

The main contribution of this work is the modification of the time-triggered system schedule in such a way that a subsequent diagnosis of faults at runtime can be performed, *first*, within a fixed and reduced observation time and, *second*, with a higher diagnostic resolution without compromising the previously defined specification constraints. The first goal is achieved by adapting the transmission times of existing messages in the network in order to obtain an optimal distribution for their detection. The second goal uses the available bandwidth of the communication channel to insert auxiliary diagnostic messages for naturally undiagnosable resources, like ones executing reception tasks only. Furthermore, the message insertion can be used to resolve infeasible constraints in terms of a too short observation time.

To the best of our knowledge, this is the first approach where the communication schedule is modified to enhance the message-based diagnosis of permanent faults. Currently, the proposed method mainly considers runtime diagnosis. However, we regard this work as an initial step towards future Design for Testability (DFT) techniques for distributed architectures, which also cover manufacturing and maintenance tests.

II. System Model and Background

Our approach is using a graph-based system specification where software functionality and hardware components are modeled by process graphs and architecture graphs, respectively. An example for

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

978-1-4799-4833-8/14/$31.00 ©2014 IEEE

[1]The term *detection* often relates to the identification of the presence of a fault, whereas *diagnosis* describes the identification of its cause.

[2]*Observation time* and *detection time* are used interchangeably, both describing the duration to detect the manifestation of a fault.

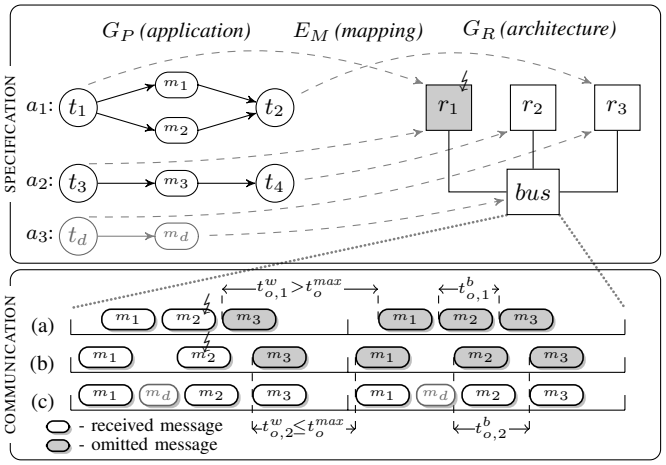

Fig. 1. Specification for a distributed system consisting of several applications mapped to a distributed architecture. The bottom part shows the communication on the bus resource for three different schedules, where t_o^w and t_o^b indicate the worst and best observation time, respectively, whereas t_o^{max} represents its upper bound. A fault ($\lightning$) is meant to occur on r_1 and prevent a further message transmission.

a system specification for a distributed system with three applications $a_i \in A$ is shown in the upper part of Fig. 1. Here, the application graphs $G_P = (P, E_P)$ consist of the vertices $P = T \cup M$ corresponding to tasks $t \in T$ and messages $m \in M$, respectively, and edges E_P representing their data-dependencies. Accordingly, the architecture graph $G_R = (R, E_R)$ connects resources $r \in R$ (e.g., Electronic Control Units (ECUs) and buses) through architectural links $l \in E_R$. Both graphs share the mappings $E_M = (P, R)$ which associate each process with a particular resource.

Time-triggered scheduling. The system model comprises a fully synchronous *time-triggered* schedule where at design-time each process, i.e., task or message, is assigned a start time and applications are defined by their end-to-end delay. Automotive applications define distributed control functions which rely on such maximal end-to-end delays from sensor to actuator rather than on single deadlines for each task. Consequently, given a sufficient bandwidth of the communication channel, the start-times of messages within the end-to-end boundaries might be adapted without affecting the control performance. The main goal of this work is to modify these schedules in order to improve a subsequent fault diagnosis in terms of detection time and diagnostic resolution without compromising the original design constraints.

We assume the required maximum fault detection time to be specified by the system designer depending on specific application requirements as well as the subsequent fault tolerance strategy.

Permanent fault diagnosis. As mentioned before, the fault diagnosis itself is not part of this paper but some of its fundamental properties are crucial for our approach. In order to exclude a single point of failure, we consider a *decentralized* diagnosis mechanism where each resource has knowledge about the system schedule and the ability to monitor the traffic on the bus it is attached to. Furthermore, we regard an *implicit* fault model, showing a fail-silent behavior, where the omittance of expected messages is an indicator for permanent faults.

In this context, while all unmasked faults originating from a physical phenomenon will finally lead to a bit-flip (see [15]), their cause can be natural, i.e., due to environmental conditions, or internal, i.e., due to unstable or marginal hardware. In contrast to the first case, which generally leads to transient faults, the second case results in intermittent and permanent faults with severe consequences, for instance, a partial or complete system failure.

Hence, besides a fast detection time, a reliable fault diagnosis must be able to distinguish between the transient and the more severe intermittent and permanent faults (for the sake of simplicity, in this work we consider only permanent faults). For our system model, we assume that a single message loss can indicate any type of fault, whereas two or more consecutive message losses point to a permanent fault of the corresponding resource.

III. Motivating Example

Fig. 1 depicts different allocations of messages to the bus resource for an exemplary specification. The communication pattern is illustrated for two periodic cycles and will be referred to as schedule (a), (b) and (c), respectively. Schedule (a) represents the unaltered message allocation resulting from the original user-defined system design. The messages m_1 and m_2 are sent from task t_1 to task t_2 and, hence, from r_1 to r_3, whereas m_3 is sent from r_1 to r_2. Now, let us assume that during the transmission of m_2, resource r_1 exhibits a permanent fault ($\lightning$), resulting in the transmission failure of all its messages. As explained before, in order to rule out transient faults, it is necessary to detect at least two consecutively omitted messages from a potentially faulty resource. Thus, for schedule (a) the diagnosis of a faulty r_1 would require a worst case observation time of $t_{o,1}^w$, which is assumed to be higher than a predefined maximum observation time t_o^{max}. By contrast, our approach uses the idle times on the bus to distribute messages from each resource in such a way that t_o^{max} between two messages is not exceeded, as shown in schedule (b). This can, in many cases, not only reduce the observation time itself, but it also enables a better overall system predictability by defining un upper bound for the observation time (compare $t_{o,2}^w \leq t_o^{max}$).

So far, we have been considering existing messages from the original system specification. However, a complex distributed system, like an E/E architecture, can contain a number of resources whose faults might not be detectable with the help of a fully implicit fault diagnosis described above. For instance, in the specification in Fig. 1 the resources r_2 and r_3 are both sinks of the corresponding task communication and, thus, not transmitting messages on their own. For this purpose, to increase the diagnostic resolution, our approach is able to identify the undiagnosable resources and integrate auxiliary processes which can broadcast short diagnostic messages in user defined time intervals (e.g., once a hyper period). Schedule (c) illustrates this for r_3 where a diagnostic message m_d (triggered by task t_d) increases the overall diagnostic resolution of the system from one resource to two resources.

IV. Diagnosis-aware System Design

In this section, we formally describe our diagnosis-aware system design which, besides the graph elements introduced in Section II, also uses the definitions listed below. Additionally, the main parameters are illustrated in Fig. 2 within a hypothetical time-triggered schedule based on the specification from Fig. 1.

- $p \in T \cup M$ – system process $p \in P$ which is either a task $t \in T$ or a message $m \in M$
- $w_{(\tilde{p},p)} \in \mathbb{R}$ – waiting time between the end of $\tilde{p}$ and the start of p for data-dependent processes
- $\mathbf{o}_p, \mathbf{o}_a \in \mathbb{R}$ – variable for the offset interval of a process p or application a
- $h_a \in \mathbb{R}$ – period after which the execution of application a is repeated
- $s_p, e_p, h_p \in \mathbb{R}$ – start time, execution time, and period of process p
- $H_r \in \mathbb{R}$ – hyper period (i.e., least common multiple) of all processes executed on resource r

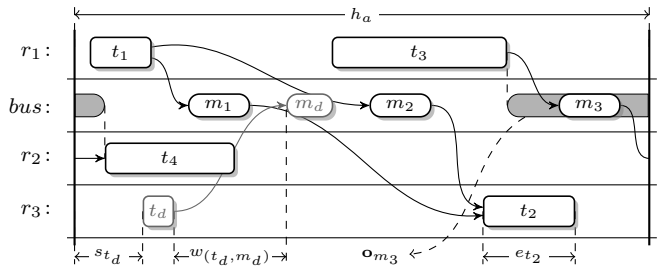

Fig. 2. A hypothetical time-triggered schedule for a system based on the example in Fig. 1. It illustrates the main parameters for our diagnosis-aware system design.

$\alpha(p) : P \to A$ – function returning the respective application a of process p

$\rho(p) : P \to R$ – function returning the respective resource r executing process p

Preventing process overlapping. To adapt a schedule, we shift messages and applications within specified time constraints, ensuring that the defined maximal end-to-end delay is met. Hence, to begin with, we define the boundaries of the application offset $\mathbf{o}_a$ as well as the individual message offsets $\mathbf{o}_m$.

$$\forall a \in A, \forall m \in M, (\tilde{p}, m) \in E_P, (m, p') \in E_P :$$

$$0 \leq \mathbf{o}_a < h_a \tag{1a}$$

$$-w_{(\tilde{p},m)} \leq \mathbf{o}_m \leq w_{(m,p')} \tag{1b}$$

According to Eq. (1a), an application can have an offset of up to its period and this is equivalent to setting an identical offset for all its corresponding processes. In contrast to this, in Eq. (1b) the offset of one single message is bounded by the waiting times to its predecessor and its successor process. For instance, in the schedule in Fig. 2 the message m_3 (whose offset interval $\mathbf{o}_{m_3}$ is indicated by the subjacent gray area) can be maximally shifted by $-w_{(t_3,m_3)}$ towards t_3 and by $w_{(m_3,t_4)}$ towards t_4.

Without loss of generality, we consider a non-preemptive single-threaded execution model where a resource can execute concurrently at most one process. Hence, it must be guaranteed that both tasks and messages are not scheduled on a shared resource at the same time. The constraints assuring non-interference between the corresponding messages and tasks are defined for the Eq. (2a) and (2b), respectively, which, in turn, determine the message offsets $\mathbf{o}_m$ and the application offsets $\mathbf{o}_a$. More precisely, in both cases the two exclusively disjunctive ($\oplus$) inequalities are preventing two processes from overlapping.

$$\forall m, \widetilde{m} \in M, m \neq \widetilde{m}, a = \alpha(m), \tilde{a} = \alpha(\widetilde{m}), \rho(m) = \rho(\widetilde{m}),$$
$$i = \{0, \cdots, \tfrac{3H_{\rho(m)}}{h_m} - 1\}, j = \{0, \cdots, \tfrac{3H_{\rho(\widetilde{m})}}{h_{\widetilde{m}}} - 1\} :$$

$$\begin{aligned} &\mathbf{o}_a + \mathbf{o}_m + i \cdot h_m + s_m + e_m \leq \mathbf{o}_{\tilde{a}} + \mathbf{o}_{\widetilde{m}} + j \cdot h_{\widetilde{m}} + s_{\widetilde{m}} \\ \oplus\ &\mathbf{o}_{\tilde{a}} + \mathbf{o}_{\widetilde{m}} + j \cdot h_{\widetilde{m}} + s_{\widetilde{m}} + e_{\widetilde{m}} \leq \mathbf{o}_a + \mathbf{o}_m + i \cdot h_m + s_m \end{aligned} \tag{2a}$$

$$\forall t, \tilde{t} \in T, t \neq \tilde{t}, a = \alpha(t), \tilde{a} = \alpha(\tilde{t}), a \neq \tilde{a}, \rho(t) = \rho(\tilde{t}),$$
$$i = \{0, \cdots, \tfrac{2H_{\rho(t)}}{h_t} - 1\}, j = \{0, \cdots, \tfrac{2H_{\rho(\tilde{t})}}{h_{\tilde{t}}} - 1\} :$$

$$\begin{aligned} &\mathbf{o}_a + i \cdot h_t + s_t + e_t \leq \mathbf{o}_{\tilde{a}} + j \cdot h_{\tilde{t}} + s_{\tilde{t}} \\ \oplus\ &\mathbf{o}_{\tilde{a}} + j \cdot h_{\tilde{t}} + s_{\tilde{t}} + e_{\tilde{t}} \leq \mathbf{o}_a + i \cdot h_t + s_t \end{aligned} \tag{2b}$$

Adapting message distribution. Having guaranteed that no two processes can utilize a resource at the same time, we can now formulate the main constraints for the diagnosis-aware message distribution, as shown in Eq. (3). As mentioned before, one of our goals is to assure the compliance with an upper bound for the observation time of omitted messages. Considering the subsequent fault diagnosis, the system designer selects a maximum observation time $t_{o,r}^{max}$ for each resource r during which a fault shall be detected. Here, the observation time is defined as the interval between the expected arrival times of two consecutive messages from the same resource.

$$\forall r \in R, \forall m \in M_r, a = \alpha(m), i = \{0, \cdots, \tfrac{H_{\rho(m)}}{h_m}\} :$$

$$\begin{aligned} \bigvee_{\widetilde{m} \in M_r} e_m \leq &\left(\mathbf{x}_{m,i} \cdot h_{\widetilde{m}} + s_{\widetilde{m}} + \mathbf{o}_{\widetilde{m}} + \mathbf{o}_{\widetilde{a}=\alpha(\widetilde{m})}\right) \\ &- (i \cdot h_m + s_m + \mathbf{o}_m + \mathbf{o}_a) \leq t_{o,r}^{max} \end{aligned} \tag{3}$$

However, as the individual process periods may differ between applications, it is not sufficient to compare adjacent messages only. Therefore, an auxiliary integer variable $\mathbf{x}_{m,i}$ is used in such way, that for each iteration i of a message occurrence m, the defined equation finds at least one closest neighbor $\widetilde{m}$ within the analyzed hyper period. In contrast to Eq. (2a) and (2b), where all system messages are considered concurrently, here we regard messages from each particular resource r, denoted as M_r, separately.

Inserting diagnostic messages. Our second goal is to extend the system schedule by inserting diagnostic messages and, hence, include previously undiagnosable resources $r \in R_d$ into our algorithm. The retrieval of these resources is performed with the help of a Depth-First Search (DFS) algorithm on the architecture graph G_R and shall not be explained in detail.

Having determined the set R_d, we can extend the system specification with particular diagnostic tasks t_d which periodically broadcast short diagnostic messages m_d. The maximum observation time serves as constraint according to Eq. (4).

$$\forall r \in R_d, t_d \in T_{d,r}, m_d \in M_{d,r} :$$

$$h_{t_d} = h_{m_d} \leq t_{o,r}^{max} \tag{4}$$

Our algorithm is applied to the extended message and task sets in Eq. (5a) and (5b) where the initial values for the start-times, s_{t_d} and s_{m_d}, are adjusted based on the corresponding application offset $\mathbf{o}_{\mathbf{a_d}}$. The inclusion of the new processes into the existing schedule is performed in compliance with all previous constraints.

$$T' = T \cup T_d \tag{5a}$$

$$M' = M \cup M_d \tag{5b}$$

Although not discussed in detail, the insertion procedure described above makes use of a special conflict refinement. In cases, where the adaptation of the message distribution cannot comply with the specified constraints (e.g., there are not enough messages to guarantee a defined maximal observation time), auxiliary diagnostic messages can be inserted to decrease the observation time. Such a scenario is also used in the case study in Section V. Note, that, depending on the used hardware, the proposed inserting scenario might require adaptations of the respective resources, e.g. in terms of its bus interface, to enable a transmission of messages.

V. Case Study Results

Given the scope of the paper, the focus here is on demonstrating the general feasibility and effectiveness of our approach. For this, the presented case study comprises a hypothetical system specification which is inspired by an automotive E/E architecture with 4 bus-attached ECUs, 25 tasks and 15 messages. The maximal tolerable observation times t_o^{max} for ECU 1 to 4 are set to 2.7, 5.0, 4.0 and 3.0 ms, respectively. The calculations were carried out on an Intel Core i5 with 2.6 GHz and 8 GB RAM. For the methods that required solving a decision problem, *Microsoft Z3* version 4.3.0 as Satisfiability Modulo Theories (SMT) solver has been used.

Fig. 3 depicts the case study outcome for both the unaltered system schedule (a) and the adapted schedule resulting from our diagnosis-aware system design (b). The bus resource in the middle of each schedule contains the entire message communication and the task schedules for ECUs 1 and 2 as well as 3 and 4 are arranged above and below, respectively. For clarity reasons only 13 ms are depicted

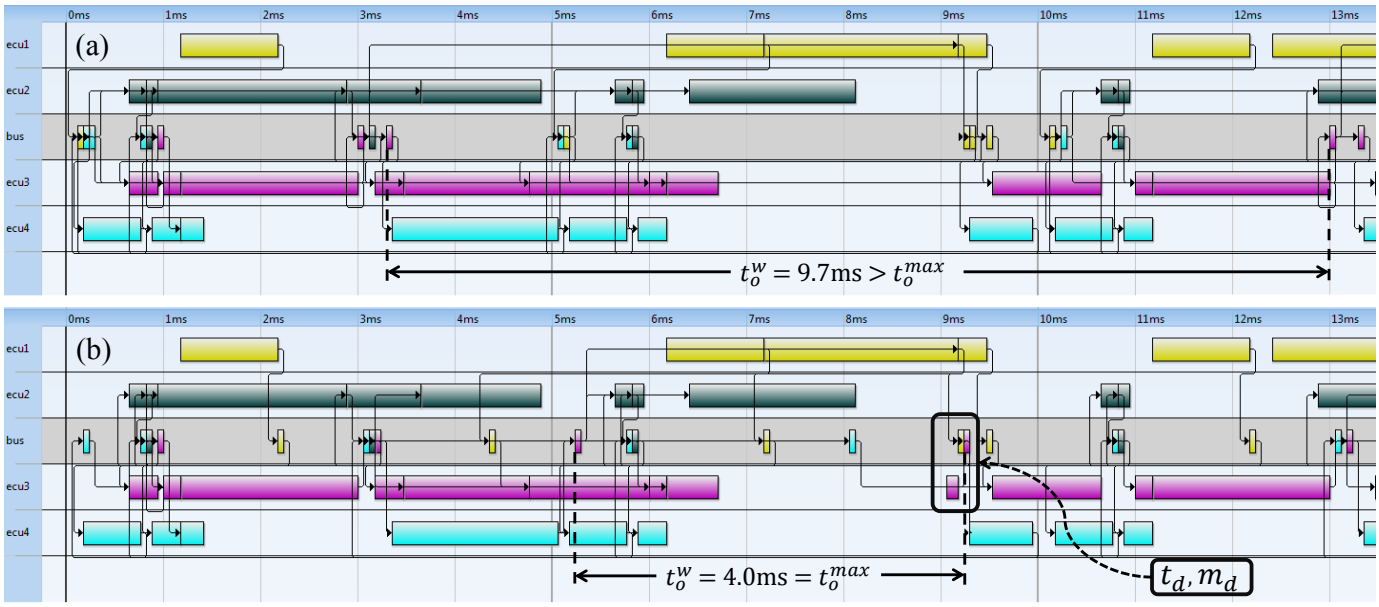

Fig. 3. Screenshot of a schedule visualization tool, showing the resulting schedules for the presented case study. Messages in schedule (a) follow solely the constraints of the initial specification and seem more clustered, whereas our adapted schedule (b) distributes them according to a defined maximum observation time t_o^{max}.

TABLE I
INITIAL AND OPTIMIZED WORST CASE OBSERVATION TIMES.

observation time t_o^w	*ECU 1*	*ECU 2*	*ECU 3*	*ECU 4*
initial	5.0 ms	5.0 ms	9.7 ms	4.5 ms
optimized	2.7 ms	5.0 ms	4.0 ms	3.0 ms

from the entire hyper period of 20 ms. All processes (i.e., tasks and messages) belonging to one ECU are shown in the same color. The worst case observation time t_o^w for ECU 3 is highlighted in both schedules and the processes t_d and m_d, inserted by our algorithm, are framed.

In general, it can be seen that our method distributes the messages more evenly without affecting the end-to-end delays of the individual applications (e.g., the relative positions of single tasks to each other are unaltered), which might be already a hint that longer observation times have been decreased. For instance, it can be clearly seen that the initial worst case observation time of 9.7 ms for messages from ECU 3 could be reduced to the required maximum observation time of 4.0 ms, decreasing the fault detection time by more than 50%. At the same time, also the observation times for the remaining three ECUs could be adapted to comply with the defined maximum times, as shown in Table I. The computation time of our algorithm for the presented case study did not exceed 1.3 seconds promising a good scalability for larger systems whose analysis is part of future work.

VI. CONCLUSION AND FUTURE WORK

This paper presents an approach to enhance the fault diagnosis in safety-critical distributed systems, like automotive E/E architectures. We assume a time-triggered and message-based communication and demonstrate how our approach can decrease the observation time necessary to diagnose permanent faults. Additionally, an insertion method for diagnostic messages is proposed which includes previously undiagnosable resources and provides a conflict refinement for failed schedule adaptations. We consider this work to be a first approach towards a DFT framework, including testing and diagnosis over the entire lifetime of a distributed system. In this context, we are currently working on suitable diagnosis algorithms which could be used in such a framework.

REFERENCES

[1] P. E. Lanigan, S. Kavulya, P. Narasimhan, T. E. Fuhrman, and M. A. Salman, "Diagnosis in automotive systems: A survey," Tech. Rep. CMU-PDL-11-110, Carnegie Mellon University Parallel Data Lab, Tech. Rep., 2011.

[2] A. Girault, C. Lavarenne, M. Sighireanu, and Y. Sorel, "Fault-Tolerant Static Scheduling for Real-Time Distributed Embedded Systems," INRIA, Rapport de recherche RR-4006, 2000.

[3] P. Waszecki, M. Lukasiewycz, A. Masrur, and S. Chakraborty, "How to Engineer Tool-Chains for Automotive E/E Architectures?" *ACM SIGBED Review*, vol. 10, no. 4, pp. 6–15, 2013.

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

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

[6] 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*, 2010, pp. 49–55.

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

[8] P. Waszecki, M. Kauer, M. Lukasiewycz, and S. Chakraborty, "Implicit Intermittent Fault Detection in Distributed Systems," in *Proc. ASP-DAC*, 2014, pp. 646–651.

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

[10] C. Bolchini and A. Miele, "Reliability-Driven System-Level Synthesis of Embedded Systems," in *Proc. of DFT*, 2010, pp. 35–43.

[11] 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*, 2007, pp. 164–174.

[12] M. Serafini, A. Bondavalli, and N. Suri, "On-Line Diagnosis and Recovery: On the Choice and Impact of Tuning Parameters," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 4, pp. 295–312, 2007.

[13] F. Sagstetter, M. Lukasiewycz, and S. Chakraborty, "Schedule Integration for Time-Triggered Systems," in *Proc. of ASP-DAC*, 2013, pp. 53–58.

[14] F. Sagstetter, S. Andalam, P. Waszecki, M. Lukasiewycz, H. Staehle, S. Chakraborty, and A. Knoll, "Schedule Integration Framework for Time-Triggered Automotive Architectures," in *Proc. of DAC*, 2014, pp. 20:1–20:6.

[15] A. Herkersdorf, H. Aliee, M. Engel, M. Glaß, C. Gimmler-Dumont, J. Henkel, V. B. Kleeberger, M. A. Kochte, J. M. Kühn, D. Mueller-Gritschneder *et al.*, "Resilience Articulation Point (RAP): Cross-layer dependability modeling for nanometer system-on-chip resilience," *Microelectronics Reliability*, vol. 54, no. 6, pp. 1066–1074, 2014.