2021 IEEE Intelligent Transportation Systems Conference (ITSC)
Indianapolis, USA. September 19-21, 2021

A Model-Based Approach to Facilitate Design of Homogeneous Redundant E/E Architectures

Hadi Askaripoor, Morteza Hashemi Farzaneh, and Alois Knoll
Chair of Robotics, Artificial Intelligence and Real-time Systems
Technical University of Munich
Boltzmannstr. 3, 85748 Garching bei München
Email:{hadi.askari,morteza.hashemi}@tum.de, {knoll}@in.tum.de

2021 IEEE International Intelligent Transportation Systems Conference (ITSC) | 978-1-7281-9142-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/ITSC48978.2021.9565115

***Abstract*—Designing the new generation of electronic and electrical architectures, guaranteeing reliable communication is a prerequisite. The fulfillment of this requirement is a challenging undertaking that requires advanced expertise and is time-consuming. This paper presents a novel model-based approach for automating the generation of optimized network architectures in the design phase, supporting homogeneous redundant routings. Our results indicate a linear growth of architecture synthesis time if the number of applications is growing while the number of nodes remains constant. Conversely, increasing the number of nodes results in exponential growth of the architecture synthesis time. We believe that the proposed approach contributes to facilitating the design of safe E/E architectures.**

***Index Terms*—Functional Safety, E/E Architectures, Model-Based Development, Homogeneous Redundant Routings, Architecture Optimization.**

I. Introduction

In the past decade, there has been significant development of Electrical and Electronic (E/E) architectures in the automotive industry. Advanced driver assistance systems and in-car entertainment require a considerable amount of data transmission over the vehicle network architecture. The reliability of the data involved in safety-critical applications must be guaranteed. Redundancy is one approach for improving the reliability that is also defined in the automotive functional safety standard ISO 26262 [1]. Therefore, the redundancy path for critical data must be specified and ensured in the early design phase of the architecture.

The design of reliable data routes in compliance with ISO 26262 is an elaborate engineering task, that is time consuming and also requires domain-specific knowledge.

According to ISO 26262, redundancy for the provision of fault-tolerance comprises two types: homogeneous and heterogeneous redundancy. Homogeneous redundancy requires mandatory duplication of the elements, either hardware components or software processes. In addition, it concentrates on the outcomes of the hardware transient faults as well as random faults in the design phase. On the other hand, heterogeneous redundancy uses the software processes to provide the redundancy for the hardware components.

This paper presents a model-based approach for automating the creation of reliable E/E network architectures in order to generate single routes and Homogeneous Redundant (HR) routes for safety-critical applications, based on predetermined optimization objectives (e.g., reducing cost by minimizing the number of allocated network links). In our approach, we collect architectural requirements such as the number of nodes, criticality level of the applications, and link cost. After analysis of these requirements, they are automatically transformed into constraints that represent the mathematical formulation (using Integer Linear Programming (ILP)) of the defined routing requirements. This transformation is performed by the Model Driven Development (MDD) approach [2]. The constraints are then solved based on predefined optimization goals (e.g., by reducing the total number of network links used). The result is an optimized E/E architecture supporting reliability by redundant routes.

Since architecture synthesis time (the time required for the generation of the architecture based on the predefined requirements) plays a significant role in accelerating the design and the development process, we present the synthesis time measurements using three various scenarios.

This paper is structured as follows: the next section discusses the related work. Section III describes our methodology including the system model, constraints encoding, optimization goals, and implementations. Section IV presents the run time evaluation and finally section V presents further steps and the conclusion.

II. Related Work

The reliability of a heterogeneous automotive system using Automotive Safety Integrity Level (ASIL) decomposition has been studied by the authors in [3]. They presented two heuristic algorithms to improve the reliability goal while minimizing the development cost for each ASIL decomposition scheme. This work calculates the reliability based on tasks mapped on the ECUs and their failure rates. The authors of [4] and [5] have focused on meeting the real-time requirement of distributed automotive applications while optimizing the development cost. However, none of these studies created any network architecture and routings. Lukasiewycz *et.al* [6] proposed a graph-based model and a constraint system to create routing of messages generated by an application which is mapped on an ECU in the architecture. The same author [7] presented a binary encoding strategy for resource allocation, binding the tasks, and routing the messages using Satisfiability (SAT)-encoding for the one-linear objective

optimization. The new Time-Sensitive Networking (TSN) standards improve the safety (e.g. by introducing mechanisms for deterministic timing and high network reliability) of E/E architectures. Optimization of routings in TSN have been explained in [8]. These papers concentrate on the relation between the schedule of the time-triggered traffic and the message routing while, redundant routing is not discussed. An automated approach, for the creation of low-redundant and single routings in automotive architectures while considering predefined optimization goals using SAT-Decoding, was explained in [9]. In this work, mapping of applications on the resources, given to the system as inputs. In addition, the generated routes were optimized based on the number of allocated links and mean-time-to-failure (MTTF) (i.e., defined as the individual failure rate for all architecture components).

The above approaches create routings for network architectures; however, they consider a predesigned architecture as an input for the constraint system and do not create HR routing. Moreover, none of these studies utilized a model-based development approach.

III. Methodology

A. System Model

The graph-based model from [6], [9], which is an extension of the model presented in [7], has been used. Our model consists of three independent graphs, namely an application graph, architecture graph, and mapping graph which are explained in the following.

Architecture graph $G_{Arch}(N, \vec{L})$ comprises nodes N which are connected by directed links $\vec{L}$. Each node can either be a processor, a switch, or an ECU. A node can be connected to one or multiple other nodes.

The application graph $G_{App}(A, D)$ includes applications $A = A_s \cup A_{ns}$ as well as their data D_s and D_{ns} respectively, where the applications are either safety or non-safety critical. Each datum is a directed edge between a defined application as a source and another specified application as a destination. An application always contains a datum which is safety-critical or non-safety-critical. Each safety-critical application as a source, in order to transmit its data to its relevant destination, must meet the predefined safety requirements (e.g., high-redundancy or low- redundancy, reliability, etc.) while non-safety critical applications are exempted from these requirements.

The mapping graph $G_M(A, N, M)$ contains applications, which act either as senders A_{sen} or receivers A_{rec}, nodes, and mapping edges. This graph illustrates on which node a certain application (A_{sen} or A_{rec}) can be implemented. In case of the mapping of A_{sen} on a node, the related node is interpreted as a source node ($N = N_s$) with mapping definition m_s while the node, which is allocated to A_{rec}, becomes a destination node ($N = N_d$) with mapping declaration m_d ($N = N_s \cup N_d$, $m = \{m_s, m_d\}, m \in M$).

In our system model, each datum, $d = \{d_{in}, d_{out}\}, d \in D$, is sent by an application A_{sen} as the sender and is received by another application A_{rec} as the receiver. For each datum d, we define d_{in} (if it is received by a node) and d_{out} (if it is sent from a node). Each datum can also be routed over any link, and any node. Moreover, it is assumed that all links can be utilized by all existing data in the architecture graph.

The routing of each safety critical or non-safety critical datum, is indicated by the routing graph G_R (a subgraph of G_{Arch}). G_R consists of directed links $\vec{L}$ which include routed data. Each $\vec{L}$ starts from the source node (N_s), where the A_{sen} is mapped, and ends at the destination node, N_d, where A_{rec} is implemented.

B. Constraints Encoding

In this subsection, we illustrate an approach for formulating constraint sets explaining the system model with the possibility of generating valid single routes and HR routes.

Similar to [6], [9], in this approach, the routing decisions are explained by encoding a binary variable for each d_{in} and d_{out} so that these variables indicate whether the corresponding d_{in} and d_{out} are utilized for transmitting the related datum D over a link $\vec{L}$. As each d_{in} and d_{out} belongs to a unique link in our model, by using the relevant d_{in} or d_{out}, the related link is activated in order to route the corresponding datum. To implement our approach, we encoded d_{in}, d_{out}, m_s, and m_d, as binary variables in our constraint system.

For instance, d_{in} is translated as an incoming datum which is allocated in the implementation system as a binary variable. In this respect, if the relevant datum d enters into a node over a link, its related $d_{in} = 1$; otherwise, $d_{in} = 0$.

1) Single Route or Non-Safety Critical Route Constraint Set: to encode non-redundant routings or single routings, the following constraints are utilized:

$\forall A_{sen} \in A, \forall N_s \in N$:

$$\sum_{\boldsymbol{m_s} \in \boldsymbol{m}} \boldsymbol{m_s} = \boldsymbol{1} \tag{1}$$

$\forall A_{rec} \in A, \forall N_d \in N$:

$$\sum_{\boldsymbol{m_d} \in \boldsymbol{m}} \boldsymbol{m_d} = \boldsymbol{1} \tag{2}$$

$\forall d_{out} \in D, m \in M$:

$$\boldsymbol{m_s} - \boldsymbol{m_d} - \sum_{\boldsymbol{d_{out}} \in \boldsymbol{d}} \boldsymbol{d_{out}} <= \boldsymbol{0} \tag{3}$$

$\forall d_{in} \in D, m \in M$:

$$\boldsymbol{m_s} + \sum_{\boldsymbol{d_{in}} \in \boldsymbol{d}} \boldsymbol{d_{in}} <= \boldsymbol{1} \tag{4}$$

$\forall d_{in} \in D, m \in M$:

$$\boldsymbol{m_d} - \boldsymbol{m_s} - \sum_{\boldsymbol{d_{in}} \in \boldsymbol{d}} \boldsymbol{d_{in}} <= \boldsymbol{0} \tag{5}$$

$\forall d_{out} \in D, m \in M$:

$$\boldsymbol{m_d} + \sum_{\boldsymbol{d_{out}} \in \boldsymbol{d}} \boldsymbol{d_{out}} <= \boldsymbol{1} \tag{6}$$

$\forall d_{in}, d_{out} \in D$, $m \in M$:

$$\boldsymbol{m_s - m_d + \sum_{d_{in} \in d} d_{in} - \sum_{d_{out} \in d} d_{out} = 0} \quad (7)$$

$\forall \vec{L_1}, \vec{L_2}, \forall d = \{d_{out}, d_{in}\}, d \in D$:

$$\boldsymbol{d_{out} + d_{in} = 0} \quad (8)$$

This constraint indicates that each application as a sender can be mapped on a node just once (1). Accordingly, (2) states that the application as a receiver can be bounded just once as well. Constraint (3) enforces at least one out-going datum (d_{out}) over a link for the source node ($N_s \Rightarrow m_s = 1$). The source node, in the event of running an A_{sen} on an arbitrary node ($N_s \Rightarrow m_s = 1$), is obliged to block all in-coming data (d_{in}) which are coming from the other nodes, according to constraint (4); as a result, there is no d_{in} for the relevant node. Similarly to (3), but this time for a node as the destination ($N_d \Rightarrow m_d = 1$), at least one in-coming datum (d_{in}) over a link must be applied, according to constraint (5). Constraint (6) expresses that the destination must not have any activated out-going data (d_{out}); in other words, if $(N_d \Rightarrow m_d = 1) \Rightarrow \sum d_{out} = 0$. Constraint (7) consists of several explanations. If a node gets allocated for A_{sen} as well as A_{rec} simultaneously, so that the node becomes a source and a destination, it must not comprise any activated data (d_{in}, d_{out}) according to (7), while drawing attention to constraints (4),(6) as well. Moreover, a node, allocated as the source not the destination (i.e., $m_s = 1, m_d = 0$), must have exactly one activated out-going datum (d_{out}) in respect of (7). While, a node, assigned to A_{rec}, must include exactly one triggered in-coming datum (i.e., $\sum d_{in} = 1$) w.r.t. constraint (7). In addition, based on (7), the nodes are enforced, which are neither a source nor a destination for routing a unique datum, to either have no triggered data (i.e., $\sum d_{out} = 0, \sum d_{in} = 0$) or to have precisely one activated in-coming datum and one activated out-going datum (i.e., $\sum d_{out} = 1, \sum d_{in} = 1$). These constraints encode single routings while avoiding routing cycles. In order to connect the d_{in} to the d_{out} while both nodes are connected by a directed link, the constraint (8) is used. For instance, if the d_{out} routing out from N_1, as the starting point of link $\vec{L_1}$, over the link $\vec{L_1}$ becomes activated, the d_{in} routing into N_2, as the finishing point of the link $\vec{L_1}$, over the link $\vec{L_1}$ must be activated too in respect of (8). The same concept is applied for $\vec{L_2}$.

2) HR Route Constraint Set for High Safety-Critical Purposes: the constraints below encode the HR route, where the generated topology for a specified datum includes at least $n_{hmr} \in \mathbb{N}$ HR routings, with duplication of the entire routing elements including the nodes (except for the source and destination nodes), and the links.

$n_{hmr} \in \mathbb{N}$ is defined as a constant variable, which determines the minimum number of devoted HR routings for the generated topology corresponding to the selected application graph.

$\forall d_{out} \in D_s$, $D_s \in D$, $m \in M$, $n_{hmr} \in \mathbb{N}$:

$$\boldsymbol{m_s - m_d - \sum_{d_{out} \in d, d \in D_s} d_{out} <= -n_{hmr}} \quad (9)$$

$\forall d_{in} \in D_s$, $D_s \in D$, $m \in M$, $n_{hmr} \in \mathbb{N}$:

$$\boldsymbol{m_d - m_s - \sum_{d_{in} \in d, d \in D_s} d_{in} <= -n_{hmr}} \quad (10)$$

$\forall d_{out} \in D_s$, $D_s \in D$, $m \in M$:

$$\boldsymbol{m_s - m_d + \sum_{d_{out} \in d, d \in D_s} d_{out} <= 1} \quad (11)$$

$\forall d_{in} \in D_s$, $D_s \in D$, $m \in M$:

$$\boldsymbol{m_s - m_d + \sum_{d_{in} \in d, d \in D_s} d_{in} <= 1} \quad (12)$$

$\forall d = \{d_{in}, d_{out}\} \in D_s, D_s \in D$, $m \in M$:

$$\boldsymbol{m_s - m_d + \sum_{d_{out} \in d, d \in D_s} d_{out} - \sum_{d_{in} \in d, d \in D_s} d_{in} = 0} \quad (13)$$

$\forall d = \{d_{in}, d_{out}\} \in D_s, D_s \in D$:

$$\boldsymbol{\sum_{d_{out} \in d, d \in D_s} d_{out} + \sum_{d_{in} \in d, d \in D_s} d_{in} <= 1} \quad (14)$$

For generating topology meeting HR routings, the constraints (4) and (9) are considered for a node in the constraint system if the node is allocated as the source ($m_s = 1, m_d = 0$). Constraint (9) states that the source node must have at least n_{hmr} activated out-going data (d_{out}) while it must not have any triggered d_{in} based on (4). Conversely, the constraints (6) and (10) are applied to a node if the related node is determined as the destination ($m_d = 1, m_s = 0$). The destination is enforced to have at least n_{hmr} activated in-coming data (d_{in}) by the constraint (10); at the same time, it must not have any operated d_{out} in respect of (6). The Constraints (11), (12), and (13) are utilized to encode the HR routings by only a node which is not assigned for the destination as well as the source ($m_d = 0, m_s = 0$). According to (11), the node, which is neither a destination nor a source, can have at most one activated d_{out} while constraint (12) states that the same node can have at most one operated in-coming datum d_{in}; moreover, the number of d_{out} routing through a link, must be equal to the number of in-coming data (d_{in}) for the same node based on (13). The constraint (14) is used to avoid cycling between two nodes (i.e., invalid routing). A cycle occurs when a datum visits a node more than once. This constraint forces a desired node to have either one activated d_{out}, one activated d_{in}, or no activated datum with another node.

C. Optimization Goals for Routings

In order to generate optimized topologies including the routes, two general optimization objectives are applied to the single as well as the HR routings constraint sets. These optimization goals focus on minimizing cost and the number of used links.

$\forall d_{in} \in d$:

$$\boldsymbol{min} \sum_{\boldsymbol{d_{out} \in d}} \boldsymbol{d_{out}} \tag{15}$$

$\forall d_{out} \in d, d \in D, c_l^{d_{out}} \in c_l, c_l \in \mathbb{W}$:

$$\boldsymbol{min} \sum_{\boldsymbol{d_{out} \in d}} \boldsymbol{d_{out} * c_l^{d_{out}}} \tag{16}$$

$\forall d_{in} \in d, d \in D, c_l^{d_{in}} \in c_l, c_l \in \mathbb{W}$:

$$\boldsymbol{min} \sum_{\boldsymbol{d_{in} \in d}} \boldsymbol{d_{in} * c_l^{d_{in}}} \tag{17}$$

According to (15), the minimization of out-going data (d_{out}) routing over the links ($\vec{L}$) resulting in mitigation of the used links in the entire created topology, is set as the optimization objective in our constraint system. Furthermore, objectives (16) and (17) describe reducing the cost of the generated topology, based solely on the cost of each link ($c_l \in \mathbb{W}$). This, in such a way that the sum of the multiplication between d_{out} and the cost of its related link ($c_l^{d_{out}}$), is reduced regarding (16). The same concept is applied for d_{in} based on (17).

D. Implementations

1) System Architecture: to generate E/E topologies based on predefined requirements, the approach presented in [10] is followed. According to Fig. 1, our system architecture collects application requirements (i.e., safety criticality, application assignment, number of required nodes, and the cost of the topology components such as the link) as the inputs to the system. After analysis of the requirements based on the inputs and defining variables for the constraint system, the constraints are generated depending on the safety-criticality level of applications and they are formulated into an ILP method. Moreover, the optimization objectives (i.e., the cost and links optimization) can be set in our constraint system. Finally, the generated constraint system is solved using a Gurobi Optimization Solver [11] in order to create either single or HR routings based on the defined constraints.

In order to implement our topology generation framework, we use a Model-Driven Development (MDD) approach to specify how the software system should work before the code is generated.

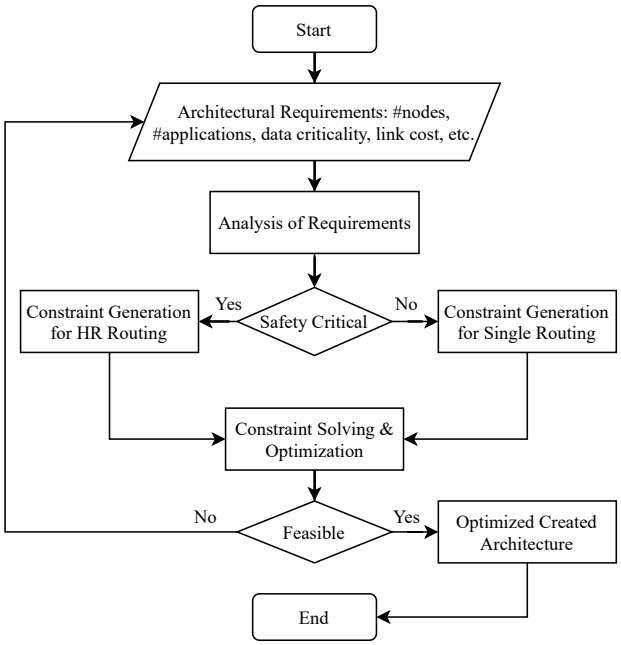

Fig. 1. The system architecture to create homogeneous redundant E/E architectures.

2) Model-Driven Development: using MDD, software developers can build elaborate applications in a visualized manner because it utilizes pre-construct application components and graphical models. Automation of the challenging programming tasks is defined as the main goal of MDD. Furthermore, particularly when developing several applications, it speeds up the redeployment, rebuilds, and tests procedures compared with traditional approach [2].

There are various types of MDD tools for creating models for software design purposes. In this work, an open-source graphic modeling tool using Unified Modeling Language (UML) from *Eclipse Foundation* was utilized. Using a similar approach, the authors of [12] present a model-based framework to facilitate schedule synthesis in TSN. The framework automates the creation and solving of scheduling constraints for the safety-critical network traffic.

According to Fig. 2, the designed metamodell (i.e., using a model to describe another model as an instance) of our framework is visualized. This metamodell consists of eight elements. The Topology element plays a role as the main class of the metalmodell where all other classes are subordinate to it. Here we have five classes as the main elements in our system model including Node, Application, Link, Data, Data-in, and Data-out; in addition, there is a data type (i.e., GRBTopology) used by the elements of the system model. To explain the relation of each element to the system model mentioned in III-A, Node class plays the same role as the node in the G_{Arch} and includes several attributes. Here, each Node can include *one-to-many* links and *zero-to-many* applications as the sender and the receiver. While Application, similarly presented in G_{App}, either as the sender or the receiver, can only be mapped on one Node; moreover,

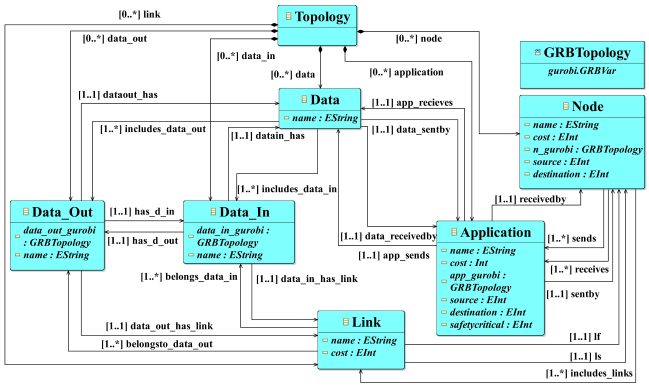

Fig. 2. The metamodell containing elements used for creating topology instances, generating constraints, and solving architectural optimization problem.

the Application can only send one Data based on *one-to-one* reference relation with Data. Furthermore, each pair of applications are considered (A_{sen}, A_{rec}) as one application in our run-time evaluation. In our designed metamodell (Fig. 2), a Link (with the same role as explained in G_{Arch}) can be referenced to a Node as its starting point and its finishing point. In addition, *one-to-many* Data-In and Data-Out can be routed over a Link. In this class diagram, Data, which is described in G_{App} as our system model, references to *one-to- many* Data-In and Data-Out while it can be either sent or received by only one Application. Eventually, according to Fig. 2, Data-In or Data-Out (described in G_{App} as well as III-B) can be routed over only one Link; moreover, each Data-In or Data-Out belongs to a Data. Also, each Data-In is referenced to each Data-out and conversely.

Algorithm 1: ADL

Input: $N = \{N_1, N_2, ..., N_p\}, A = \{A_1, A_2, ..., A_p\}$, $D = \{\{d_{in_1}, .., d_{in_p}\}, \{d_{out_1}, .., d_{out_p}\}\}$, and $M = \{\{m_{s_1}, .., m_{s_p}\}, \{m_{d_1}, .., m_{d_p}\}\}$, $p \in \mathbb{N}$

Output: Created routings only for the destination node running a safety-critical application

```
1  for i ← 0 to A do
2      for j ← 0 to N do
3          if safety-criticality of A becomes true then
4              if m_d = 1∧ m_s = 0 then
5                  for k ← 0 to d_in do
6                      expr_1 · add(d_in · gurobi);
7                  end
8                  expr_2 · addConstr(m_d − m_s − expr_1 <= −1);
9                  model · add(expr_2);
10             end
11         end
12     end
13 end
```

We presented the ADL algorithm, as shown in Algorithm 1, to ensure that at least two incoming data (d_{in}) enter only to the destination node to create the HR routings. The details of the ADL algorithm are explained as follows. For each application (line 1), all nodes pass through conditions (line 2) including the safety-criticality level of each application (line 3) and also if the receiver application is mapped on a node (line 4). After passing the $if condition$, the binary variable of all data entering the node (d_{in}), is added to the expression 1 (line 5-7). The defined constraint is applied and added to the expression 2 (line 8), and finally, it is added to our system model (line 9).

IV. Run-Time Evaluation

Our proposed approach facilitates the design process for system architects. Nonetheless, high computation times for generation and solving the constraints will have a negative effect on practical usage. Therefore, this section focuses on measuring the run time of constraint generation and solving for three specific scenarios.

In the first scenario, the number of HR routes is increased while the number of applications and nodes is constant. In the second scenario, the number of applications increases whereas the number of nodes and HR routes remains constant. In the last scenario, the number of nodes is grown and the number of HR routes and applications remains unchanging. Each measurement consists of two parts: the time for generation of ILP constraints and the time for solving.

To analyze the role of HR routes in the constraint generation run time in the first scenario, we generated topologies with the same number of nodes and applications, fixed to 100 and 20 respectively, while increasing the number of routes from one to six. As Fig. 3(a) shows, the number of HR routes does not significantly influence the constraint generation time.

For the second scenario, the number of applications is increased from 20 to 100 while the number of nodes and HR routes are fixed at 100 and 6 respectively. The goal is, to observe the effect of the increase in the number of applications on the constraint generation time. As Fig. 3(b) shows, the run time for constraint generation rises linearly (approximately from 4500 to 27000 milliseconds) in proportion to the increase in the number of applications, once the number of applications grows from 20 to 100.

In our last scenario, we keep the number of applications and HR routes constant, 2 and 6 respectively, whereas the number of nodes increases to 160. As shown in Fig. 3(c), the generation time grows as the number of nodes increases. However, the number of nodes has less impact on constraint generation time than the number of applications based on Fig. 3(b). Furthermore, it can be seen from Fig. 3(c) that the run time exhibits a non-linear increase.

In the second part of this experiment, the run time to solve the created ILP constraints using the Gurobi Solver is measured for the three aforementioned scenarios. Moreover, we optimized the number of allocated network links (defined in 15). Fig. 3(d) shows the measured solving time for creating

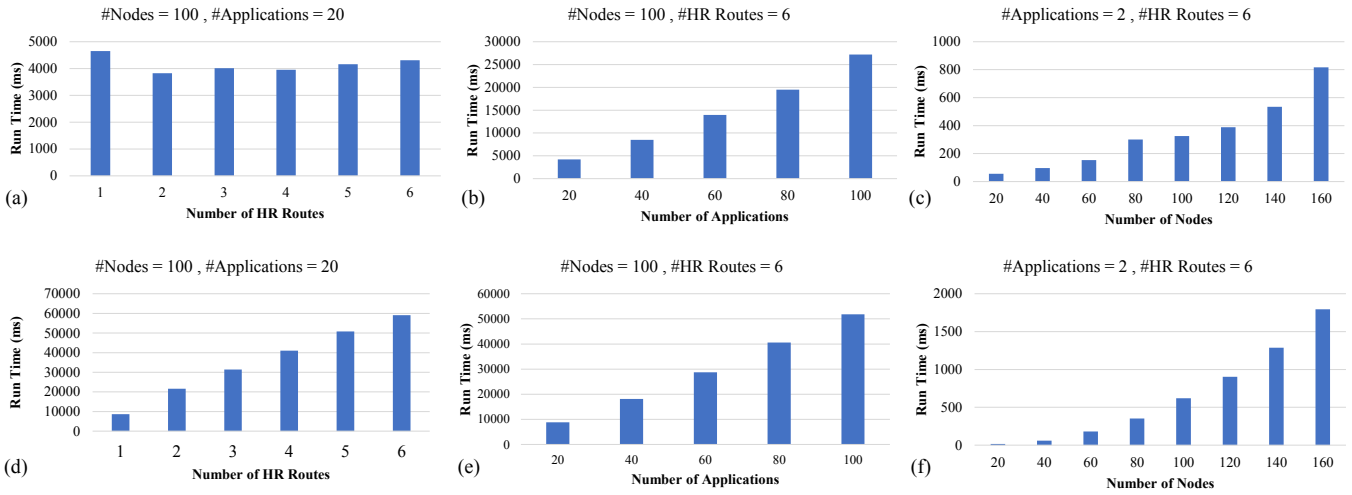

Fig. 3. The result of architectural synthesis time of the defined experimental scenarios for constraint generation (a, b, and c) and constraint solving (d, e, and f).

HR routes with the number of nodes and applications are fixed at 100 and 20 respectively. It shows that the run time grows linearly with the increase of the number of HR routes. The computation time indicates a linear increase as the number of applications grows and the nodes and number of HR routes remain constant (Fig. 3(e)). For example, to solve the constraints of an architecture that consists of 100 nodes and 20 applications supporting six HR routes, the required time is roughly 10 seconds while the architecture comprising 100 applications and the same number of nodes as well as HR routes, requires about 50 seconds. In the last scenario, we increase the number of nodes from 20 to 160 while the number of applications and HR routes are equal to 2 and 6 respectively. In contrast to Fig. 3(d) and Fig. 3(e), Fig. 3(f) exhibits exponential growth in the solving time as the number of nodes increases. As an example, the solving time for 80 nodes is roughly 400 milliseconds while it increases to 1700 milliseconds for 160 nodes.

V. Conclusion

In this paper, we have proposed a novel model-based approach to automate the creation of automotive E/E architectures in the design phase while calculating homogeneous redundant routings for safety-critical applications. We also optimized the generated architectures based on predefined optimization goals such as reducing the number of used network links. Integer Linear Programming (ILP) was utilized to implement and solve the routing constraints. We evaluated the performance of our presented approach by measuring the run time of constraint generation as well as solving for three different scenarios. The solving time increased exponentially as the number of nodes was risen. In addition, we observed approximately a linear growth as the number of safety-critical applications and homogeneous redundant routes increased with the number of nodes constant.

In this paper, we concentrated on improving the redundancy aspect in the design of E/E architectures. However, this approach needs to be extended by a measurable metric (e.g., MTTF) to assess architectural safety improvements. Moreover, we will design industrial feasibility studies to further evaluate and enhance the presented approach.

References

[1] ISO 26262:2018 - Road Vehicles - Functional Safety, International Organization for Standardization in ISO 26262, Sep. 2020. [Online].

[2] Atkinson, C. and Kuhne, T., 2003. Model-driven development: a metamodeling foundation. IEEE software, 20(5), pp.36-41.

[3] Xie, G., Chen, Y., Liu, Y., Li, R. and Li, K., 2017. Minimizing development cost with reliability goal for automotive functional safety during design phase. IEEE Transactions on Reliability, 67, pp.196-211.

[4] Gan, J., Pop, P. and Madsen, J., 2014. Tradeoff analysis for dependable real-time embedded systems during the early design phases. Dissertation, DTU Compute.

[5] Tamaş-Selicean, D. and Pop, P., 2011. Optimization of time-partitions for mixed-criticality real-time distributed embedded systems. In 2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops (pp. 1-10).

[6] Lukasiewycz, M., Shreejith, S. and Fahmy, S.A., 2014. System simulation and optimization using reconfigurable hardware. In 2014 International Symposium on Integrated Circuits (ISIC) (pp. 468-471). IEEE.

[7] Lukasiewycz, M., Streubuhr, M., Glaß, M., Haubelt, C. and Teich, J. Combined system synthesis and communication architecture exploration for MPSoCs. In 2009 Design, Automation & Test in Europe Conference Exhibition (pp. 472-477). IEEE.

[8] Nayak, N.G., Dürr, F. and Rothermel, K., 2016. Time-sensitive software-defined network (TSSDN) for real-time applications. In Proceedings of the 24th International Conference on Real-Time Networks and Systems (pp. 193-202).

[9] Smirnov, F., Reimann, F., Teich, J., Han, Z. and Glaß, M., 2018. Automatic optimization of redundant message routings in automotive networks. In Proceedings of the 21st International Workshop on Software and Compilers for Embedded Systems (pp. 90-99).

[10] Askaripoor, H., Farzaneh, M.H. and Knoll, A. Considering Safety Requirements in Design Phase of Future E/E Architectures. In 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA) (Vol. 1, pp. 1165-1168).

[11] Gurobi Optimization, I., 2020. Gurobi optimizer reference manual. URL http://www. gurobi. com.
Available: https://www.iso.org/standard/68383.html

[12] Farzaneh, M. H., Kugele, S., and Knoll, A. (2017, September). A graphical modeling tool supporting automated schedule synthesis for time-sensitive networking. In 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA).