Evolutionary Exploration of E/E-Architectures in Automotive Design

Ralph Moritz, Tamara Ulrich, and Lothar Thiele

Abstract In this work we design an evolutionary algorithm for the exploration of the electric and electronic (E/E-)architectures in a car. We describe a novel way to simultaneously optimize the assignments of function components to electronic control units (ECU) and of ECUs to busses. To this end we present a suitable representation as well as corresponding variation operators. We also provide heuristics for the optimization of the communication infrastructure, i.e. how busses are connected to each other. Preliminary results show that the approach is able to produce architectures similar to those which are used nowadays, as well as promising alternatives.

1 Introduction

In the last decade, the electric and electronic systems (E/E-architectures) of a car have become increasingly complex. Additional functions such as brake assistance, skidding control, or parking aid have been integrated, which in turn lead to an explosion of the number of electronic control units (ECUs) and their communication. As a result, the design of E/E-architectures gets more and more involved and time consuming. Many communication links and dependencies between single functions prohibit a simple decomposition of the architecture design task into independent subtasks. Instead, even minor individual decisions may have a global influence on the whole system evaluation in terms of objective functions and constraints.

In this work, we present a high-level optimization framework which considers all major tasks, from setting up the ECUs and placing them in the car up to connecting them to busses and define the cable routing. It is not the purpose of the proposed approach to fully automate the design of future E/E-architectures. Rather, it should

Ralph Moritz
Robert Bosch GmbH, Schwieberdingen, Germany, e-mail: Ralph.Moritz@de.bosch.com

Tamara Ulrich and Lothar Thiele
Computer Engineering and Networks Laboratory, ETH Zurich, 8092 Zurich, Switzerland e-mail: firstname.lastname@tik.ee.ethz.ch

provide decision support for design engineers by presenting alternatives to current designs.

2 The Design of an E/E-Architecture

Figure 1 shows an example of an abstract E/E-architecture design. We assume that the E/E-architecture will have to implement a number of composite functions, which in turn consist of a number of individual components that communicate via signals. Such components can either be sensors, processing components, or actuators. In order to incorporate a function into the car, its components have to be assigned to assembly units, in general ECUs, which in turn have to be placed in an appropriate location in the car and assigned with additional hardware (intelligent semiconductors), depending on the components requirements. Next, the ECUs need to be assigned to busses of a specific type (LIN, CAN, MOST, FlexRay,...), which need to be connected via gateways, such that all digital signals can be routed using the constructed inter-bus topology, i.e. over a sequence of busses connected by gateways. Beside the inter-bus topology, for each bus an intra-bus topology (e.g. linear or star) needs to be defined, that states which ECUs are physically connected.

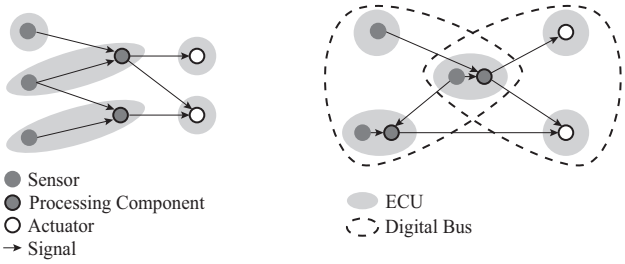

Fig. 1 E/E-Architecture Design Example: A given function containing 7 components and 7 signals is assigned to ECUs (left) and the corresponding ECUs are assigned to digital busses (right), where the middle ECU acts as a gateway between the two busses. Intra-bus topology is not shown.

The resulting architectures are evaluated according to two objectives which are to be minimized. The first one is cost, which is governed by the cable and ECU cost. The cables in turn depend on the signals that have to be routed between the ECUs, on the used communication structure as well as on the placement of the ECUs. The second objective is ECU complexity defined as the average number of different functions assigned to an ECU. A low complexity increases the reliability of a car as the number of functions which are affected by a single ECU failure is minimized.

3 Optimizing E/E-Architectures

Due to the size of the search space and the presence of several objective functions, we propose to design a multiobjective evolutionary algorithm to optimize the E/E-architecture design problem. Evolutionary algorithms (EAs) are considered to be black box optimization algorithms and therefore, they are used mostly for problem classes where classical standard methods fail or can hardly be applied [9]. To make the search most efficient, the EA needs to be adapted to the problem. Problem-specific knowledge enters EAs through (a) embedded local heuristics, (b) an appropriate representation of solutions and (c) corresponding variation operators. The use of local heuristics reduces the search space size of the EA but the available diversity of solutions in the population might be lost [3]. We decided to do the following split, based on the observation that many decisions only affect one objective, namely the cost, and the cost-optimal solution can be found in a straight-forward manner.

EA-optimized decisions :

- Task 1: Assignment of components to ECUs
- Task 2: Physical placement of ECUs
- Task 3: Assignment of ECUs to digital busses

Decisions made by local heuristics :

- Intelligent semiconductor selection for each ECU (simply take the cheapest that satisfies the memory requirements of the processing components of that ECU)
- Inter-bus topology(described in Section 3.3)
- Bus type selection (choose the cheapest type that satisfies the data-rate requirements of the signals that have to be routed over the bus)
- Intra-bus topology (choose such that the cable cost is minimized)

3.1 Representation

Now that we have reduced the number of tasks that have to be optimized by the EA, we need to design a suitable representation to optimize these tasks simultaneously. This is in contrast to the two-stage approach of Limam [6], who first optimizes Task 1 and then takes the best assignment and optimizes Task 2. It also differs from the work of Hardung [4], where physical placement and cable routing is fixed and only the assignment of components to the given ECUs is optimized. Furthermore, the representation must be able to express all possible solutions and allow the design of operators that have a direct and controllable effect [8].

The assignment of components to ECUs corresponds to a partitioning of the components into k clusters, where k, i.e. the number of clusters, is a parameter to be optimized. The assignment of ECUs to busses starts with partitioning the ECUs to busses and later on defining gateways to connect the busses (see section 3.3). This partitioning differs from the first because the number of ECUs as well as the number

of busses is not fixed. Therefore the second partitioning depends on the first. One could handle the tasks separately and require additional repair strategies to make both partitioning fit together [1]. In our approach this dependency is handled by the data structure of the representation to reduce the number of infeasible assignments and avoid the bias that may be induced by a repair strategy. This data structure is in form of a hierarchical partitioning, where components are first partitioned into ECUs, and the ECUs are then partitioned to busses, see Figure 2. This data structure is implemented as a tree which allows a direct manipulation by the operators. Representing the placement of ECUs in the car is also straight-forward. We assume that we are given a set of possible mounting spaces and simply add a mounting space field for each ECU node in the representation.

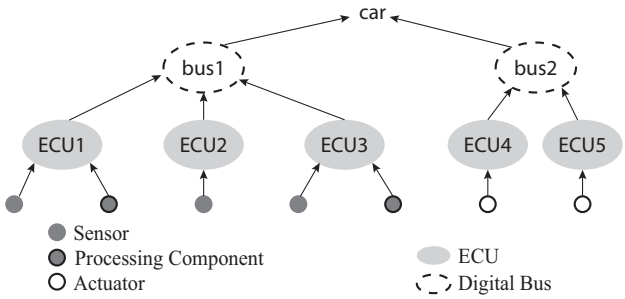

Fig. 2 EA Representation: Hierarchical Partitioning (Component to ECU and ECU to Bus)

3.2 Variation Operators

In general, a mutation operator should perform a slight and random change. Different operators can be characterized according to their exhaustiveness, locality and bias, for which we gave definition and measures in [7]. Focusing on partitioning problems, Falkenauer [2] did extensive research. He found that a mutation operator must be able to merge and split clusters, and optionally to move single elements from one cluster to another. We extended his ideas for the hierarchical partitioning mutation operator as follows: First, a partitioning level (i.e. either the component to ECU or the ECU to bus level) is selected at random. Then, either merge, split or move is selected at random. If a merge operation is selected, two random clusters are merged. In the case of a split operation, a random cluster is split into two random parts. Finally in case of a move operation, a random element is moved to a random cluster. The mutation of the mounting space assignment is straight forward, a random ECU is selected and its mounting space field is randomly changed.

The role of recombination is to improve a solution by adding a certain property of an already good one. Therefore, these properties must be reachable and exchange-

able by the recombination operator. If not, a recombination may just perform large changes, without a driving force to improve a solution. For the recombination operator we also make use of Falkenauers work. We again select a partitioning level at random. On this level we randomly select a cluster of the first solution and copy it to the partitioning of the second solution. As the elements of the new cluster now occur twice in this partitioning, the occurrences originating from the second solution are deleted. In the same way, we also add a cluster of the second to the first solution. The proposed recombination therefore creates two new solutions by directly exchanging ECUs or whole busses.

3.3 Gateway Selection Heuristic

The busses resulting from the hierarchical partitioning are not yet connected via gateways and therefore do not ensure that ECUs which implement components that need to communicate to each other and which reside on different busses actually can transmit signals via communication paths on the intra-bus topology. We decided that the busses should be connected in a tree-like manner, such that the routing of signals is unambiguous. This is in accordance with architecture examples from the industry, in which most busses are also connected in a tree shape.

To generate a tree, we first construct a graph where the nodes are the busses and the edges represent the signals that have to be transmitted between the two corresponding busses. Each edge is weighted with the total data rate of all signals that have to be transmitted between sink and source bus of that edge. We then apply a standard algorithm to find a maximum spanning tree of that graph [5]. This spanning tree is unique only if no two edges have the same weight. To remove ties, a given ordering of the signals is used to identify the edge with the lower rank. After calculating the spanning tree, a gateway is created for each edge in the spanning tree by adding an ECU from one bus to the other. This is done in such a way that the cost for additional cables is minimized.

The use of this heuristic has several advantages. First, the resulting topologies assure that for each signal a unique path from sender to receiver exists. This is given because a spanning tree must preserve the connectivity of the initial graph, which is defined by the signals. It is further worth mentioning, that if a bus does not exchange signals with the other busses, the heuristic will not create a gateway. Therefore the resulting topologies are connected as much as needed and as less as possible. Second, the distance of two busses in the topology is related to the data rate of the exchanged signals. Therefore large communication needs are handled as direct as possible. All in all this heuristic reduces the size of the search space by omitting infeasible topologies and further is biased on preferred topologies.

3.4 Handling Infeasibility

There are several situations in which a generated E/E-architecture is infeasible. These architectures could be repaired using heuristics. These repairs may introduce an unintended bias and require extensive problem-specific knowledge. Therefore, we decided to use a generic repeat-strategy. If a mutated architecture is infeasible, the parent is mutated again until a feasible one is found. If no feasible architecture can be found after a fixed number of trials, the original architecture is returned.

4 Conclusions

We presented an evolutionary algorithm that optimizes E/E-architectures in a holistic manner. We first decided which optimization tasks should be optimized by the EA and which by local heuristics. Then, we propose to use a hierarchical partitioning in order to represent the assignment of components to ECUs on the first level and of ECUs to digital busses on the second level. We also define suitable variation operators, as well as heuristics to optimize the communication between digital busses. Preliminary results on a real-world problem with 20 functions and a total of 150 components show that our algorithm is able find both existing architectures which are used in practice, as well as new designs that can be used to improve existing designs. Future work contains the evaluation of operators and selection strategies as well as a more problem specific constraint handling strategy.

References

1. Blickle, T.: Theory of Evolutionary Algorithms and Application to System Synthesis. PhD Thesis, ETH Zurich (1997)
2. Falkenauer, E.: Genetic Algorithms and Grouping Problems. Wiley, New York (1998)
3. Grosan, C. and Abraham, A.: Hybrid Evolutionary Algorithms: Methodologies, Architectures, and Reviews. In: Hybrid Evolutionary Algorithms, Studies in Computational Intelligence Springer 2007
4. Hardung, B.: Optimisation of the Allocation of Functions in Vehicle Networks. PhD Thesis, University of Erlangen-Nürnberg (2006)
5. Kruskal, J.B.: On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In: Proceedings of the American Mathematical Society, vol. 7, number 1, pp. 48-50, (1956)
6. Limam, M: A New Approach for Gateway-Based Automotive Network Architectures. PhD Thesis, Dresden University of Technology (2009)
7. Moritz, R., Ulrich, T., Thiele, L., Buerklen, S.: Mutation Operator Characterization: Exhaustiveness, Locality, and Bias. In: Proceedings of the 2011 IEEE Congress on Evolutionary Computation, IEEE Press 2011
8. Rothlauf, F.: Representations for Evolutionary Algorithms. In: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation, ACM 2010
9. Yu, X. ; Gen, M.: Introduction to Evolutionary Algorithms. Springer (2010)