An Optimization Approach for Automotive Systems Architecture Driven by Safety and Cost

Mohamed Slim Dhouibi, Jean-Mars Perquis, Laurent Saintis, Mihaela Barreau

An Optimization Approach for Automotive Systems Architecture Driven by Safety and Cost

Méthode d'optimisation de l'architecture des systèmes automobiles Dirigée par le coût et la sûreté de fonctionnement

S. DHOUIBI et J. PERQUIS
VALEO
76 Rue Auguste Perret
94000 Créteil

L. SAINTIS et M. BARREAU
LARIS , Université d'Angers
62 avenue Notre Dame du Lac
49000 Angers

Summary

Safety critical systems are present, today, almost in every car. They ensure different functionalities such as braking, steering and airbag deployment etc… The failure of these systems could lead to hazardous situations. To ensure that the risk in these systems is reduced to an acceptable level, the automotive industry refers to ISO-26262. It is the functional safety standard for electrical and electronic systems in road vehicles. It focuses on the requirements, processes and methods to deal with the effects of systematic failures and unsystematic hardware failures. Reaching a compliant design is, often, challenging particularly for high safety constraints systems. It has been also noted that, sometimes, due to safety constraints a design could lead to a cost derive. Ensuring that the design remains competitive in terms of cost is vital. With the growing complexity in functionalities and in size, the system design cycle can benefit from an approach that can help the designers make the best architectural choices to reach an optimal design. In this paper, we propose an approach for system design architecture optimization driven by the safety and cost constraints. It consists of an architecture synthesis and mapping approach that takes into account the safety constraints in the ISO 26262 context. It allows, at one hand, to reach a system preliminary architecture by choosing the best component that reduce the overall cost. On the other hand, it leads to a mapping that respects the safety constraints related to safety levels or to dependant failures.

Résumé

Les systèmes critiques sont aujourd'hui présents dans la grande majorité des voitures. Ils assurent des fonctions diverses telle que le freinage, la direction et les airbags etc. La défaillance de ces systèmes peut mener à des situations dangereuses. Pour assurer que le risque dans ces systèmes est à un niveau acceptable, les acteurs de l'industrie automobile se basent sur le standard ISO 26262. Il s'agit du standard en vigueur pour les systèmes électriques et électroniques embarqués dans les véhicules routiers. Ce standard fournit les méthodes et les procédures pour traiter les défaillances systématiques et non systématiques. Atteindre une architecture respectant les contraintes de sécurité est de plus en plus compliqué, particulièrement pour les systèmes à un haut niveau de sécurité. Satisfaire ces contraintes a mené parfois à une dérive du coût au moment où la compétitivité du design au niveau coût est critique.
Avec la multiplication des fonctionnalités critiques et la croissance de leur taille et leur complexité, une approche qui assiste le processus de conception pourrait aider à atteindre une architecture optimale. Dans cet article, on propose une approche d'optimisation de l'architecture guidée par le coût et la sureté de fonctionnement. Elle s'agit d'une approche de génération d'architecture et d'allocation tenant en compte les contraintes de sureté de fonctionnement dans le contexte de l'ISO 26262. Elle vise, d'une part à atteindre une architecture préliminaire à travers un choix des composants réduisant le cout total. D'autre part, elle permet de trouver une allocation sur ces composants qui respecte les contraintes liés aux niveaux de sureté.

Introduction

The design cost is a crucial parameter for any project particularly for automotive embedded systems. During the design process, the architectural choices and decisions are, often, guided by their impact on the overall cost. For critical systems, safety constraints are having an increasing influence on the design and cost. In the context of ISO 26262, the safety levels (ASILs) allocations and decompositions choices have a considerable impact on the architecture. These choices can be translated into functional decompositions at functional level and as a mapping and redundancies choices at physical level. These choices could lead to lowering the development cost or to incurring unnecessary extra-costs. It is necessary, today, to take, efficiently, these constraints into consideration during the design process to reach a compliant and competitive solution.
Currently, finding the solution that make a compromise between the different constraints (functional, safety and cost) is based on the engineer expertise. Since such approach remains error prone and does not guarantee the optimality of the retained solution, an automated approach is, in our opinion, needed to reach alternative cost optimal solutions.
In this article, we propose an optimization approach driven by safety and cost. It aims at reaching a compliant design without incurring unnecessary costs.

The paper is organized as follows: first, we discuss the state of the art and the related works. In the second section, we give a description of the notation and definitions of the different elements used along the paper. We fix the objective of the approach presented in this paper in the third section followed by an overview of its steps approach. A use case example is presented in the fifth section. The paper ends with a conclusion and the future works.

Related Works

The design optimization motivated multiple works. Multiple approaches have been proposed to reach optimal design taking into account various parameters such as cost, reliability and safety. These approaches target different industries.
Some targeted the System-On-Chip and they are referred to as Electronic System-Level (ESL) synthesis approaches. They aim at providing and supporting with tools a design process leading to generating SOC architecture: "The task of ESL synthesis is then the process of selecting an appropriate platform architecture, determining a mapping of the behavioral model onto that architecture, and generating a corresponding implementation of the behavior running on the platform". We may cite for example Daedalus(Nikolov and Thompson 2008), SystemCoDesigner (Keinert et al. 2009), System-On-chip Environment (Dömer et al. 2008) . Based on a functional model and using design exploration approach, these approaches reach a mapping of the functional architecture on a platform. The platform is consisting of processing elements, communication elements and bus databases. The reader can refer to (Gerstlauer and Haubelt 2009) for a detailed description of these approaches and detailed comparison between them. As far as we are concerned, these approaches cannot be used for automotive systems. The approach remains domain specific, mainly for SOC or MPSOC.
In the automotive domain context, Archeopterix (Aleti and Bjornander 2009) adopted a slightly similar approach to ESL synthesis methodologies. It aims at finding a cost optimal mapping of Software functions on a network of ECUs. The approach takes into account multiple parameters such as CPU load, network bandwidth, reliability. The safety constraints are, though, not taken into account. Co-localization constraints, that could be used to express a safety constraint, were introduced in the approach amongst the mapping constraints. But we are convinced that using a Co-localization will not be enough to express all the safety constraints. For example, it is not possible to verify the conditions for mixed ASIL levels cohabitation requirements or independency requirements in respect to the standard.
While the approaches presented above focus on optimizing the mapping, the approach proposed by HIP-HOPS (Papadopoulos et al. 2011) focus on the redundancies introduction and the ASIL allocation as an optimization approach. It is more safety oriented than the previously presented works through automatic Fault tree analysis (FTA). But, it can be applied at a single level: functional or physical. Since, it does not support the link between these levels; it is often used at the physical level. ASIL are allocated to the failures modes of the components and different redundancy patterns are used. Unfortunately, redundancy at this level is not often the favorite choice for automotive systems designers. Many alternative solutions may also be missed where the redundancy is only at functional level.
The approach we are proposing in this paper takes efficiently the safety constraints and cover the functional and physical level. It is inspired from the works on ESL approaches where an architecture is generated and a mapping is specified. It differs by taking into account the safety levels. On the other hand, while it may still introduce redundancies in the architecture, it remains different from HIP-HOPS approach since the redundancies is not systematically introduced but results from the combination of functional decomposition, ASIL allocation and mapping onto an element.

Basic Notations and Definitions

To ease the reading, this section introduces definitions of the notations that are used in the rest of the paper.

A functional architecture of a critical system can be described as several functions interacting between each other to guarantee a set of services which are the functionalities of the system in question. A Functional architecture can, thus, be described as:
FA= $\{S_1, S_2,\ldots,S_n\}$, a set of services ensured by the system.

Each Service is described using a Data Flow Graph (DFG) graph: G= {V,E} where,

- V = {f1…fn} is a set of functions
- E = {fl1…flm} is a set of flows connecting the functions

We chose to use DFG because it is particularly adapted for this case. The main advantage of data flow graphs over other models is their compactness and the easiness of the interpretation. That is, the translation from the conceived system to a DFG is straightforward and, once accomplished, it is equally straightforward to determine by inspection which aspects of the system are represented.

The flows connect the functions to describe the data dependency. A flows is defined as FL = (T,FS,FT) where T is the type of flow exchanged, FS is the function source of the flow, FT is the function target of the flow.

The Functions represents the elementary tasks of the system. They are defined as a tuple F =(A,H,SS) where A is the ASIL allocated to the function, H is the set of possible hosts of the functions, SS is the set of subsystems that could implement the function.

The hosts are the physical components that implement the functions. A sensor for example is a possible host for an acquisition function. A host is defined as tuple H = (A,C,SF,CI,F,FM) where , A is the highest ASIL that can be reached by the component, C is the cost of the component, SF is the separation feature that could allow hosting mixed safety levels functions, CI is the communication interfaces that could be implemented by the component, F is the set of functions allocated to this host and FM is the set of failure modes of the host.

The subsystems are the main constituents over which the system functionalities are distributed. The subsystems are defined as SS = (C,H,F) where C is the cost of the subsystem, H is the set of hosts added to the subsystem, F is the set of functions implemented by the subsystem.

A physical architecture is the result of successful mapping. It is defined as PA = (SS,B) where SS is the set of subsystems in the architecture and B is the set of buses ensuring the flows exchange between the subsystems.

A bus is the implementation of the functional architecture flow at physical level to ensure the communication between the subsystems. A bus allows the communication between the functions that are allocated to different subsystems. It is defined as B = (FL,P,FS, FT) where FL is the sets of the flows exchanged through the bus, P is the communication protocol. FS and FT are the communicating hosts.

Objective

Our objective is to demonstrate that the problem of finding optimized system architecture, i.e. finding a suitable (safety-related) physical architecture and its corresponding functions mapping, can be solved jointly and correctly.
Therefore the goal is to reach functions and flows mapping that respects the available resources and the safety constraints and optimizes the design cost. This mapping includes the spatial allocation of the functions and the flows in form of their deployment to hardware resources and to subsystems considering the safety constraints.
The problem can be described as follows:
The joint HW resources choice and mapping problem consists of determining a suitable choice of the subsystem configuration and HW resources as well as the functions allocation. It consists of finding an architecture containing a set of subsystems {SS1, SS2,…SSn} connected using a set of Buses {B1…Bk} where each subsystems SSi contains a set of HW resources {H1,H2…Hm} implementing a set of functions from the functional architecture. In the end, it comes to finding a mapping where:

- For each functions F, the tuple (ss,h) specifying to which subsystem and host the function is allocated, is determined
- For each flow FL the bus B that will carry the flow is determined
- The safety constraints due to the safety levels values are respected

In this paper, we propose a simple formalization of the joint architecture synthesis and mapping problem to explore the design solution space and find the cost optimal solution.

Approach Overview

This section provides an overview of the approach and the different steps of the process.
We propose here an approach that allows to automatically reaching from a functional description, an optimal physical architecture taking into account the safety constraints due to the allocated ASIL.
The approach, as shown in Figure 1, covers a part of the design process that was often manually done allowing the exploration of different alternatives and eventually reaching an optimal design. It consists of five main steps: Functional architecture, SIL allocation, Safety constraints extraction, Mapping and cost estimation.

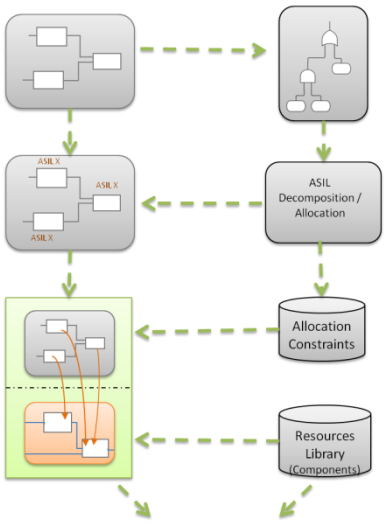

Figure 1. Optimization Flow Overview

1- Main Steps:

a Functional Model

The first step is to model the functional architecture. The functional model aims to describe how the functionalities (services) are ensured by the system. It allows describing the functional decomposition and functional redundancies introduced, if any. In order to ease the usage of the functional model for next steps of the approach, we added few rules on the granularity level and the flows. The functional model can be used for input for the mapping process only if it is described at a level where the nodes can only be allocated to a single host. Figure 2 shows an example of functional model that could be exploited in the next steps.

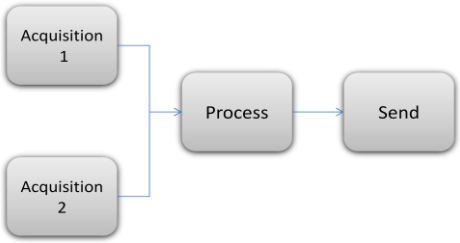

Figure 2. Functional Architecture Example

b Safety Levels allocation

A safety analysis based on the functional model leads to obtaining the minimal cut sets leading to the violation of the safety goals. These MCS are used to retrieve a possible ASIL allocation to the different functions. Due to the decomposition patterns defined in the standard, different combinations are possible. In a previous article (Dhouibi et al, 2014), we have studied the safety levels allocation problem and proposed an approach to retrieve the different possible combinations. These combinations are first filtered to remove the least practical solution. The designer chooses the best fitted solutions. These are further investigated in the next steps.

In the case of the example provided in figure 2, with a safety goal of an ASIL B consisting of sending the correct data, we choose the following allocations:

Function	Acquisition 1	Acquisition 2	Process	Send
ASIL	A	A	B	B

Table 1. ASIL allocations

c Safety constraints extraction

ASIL decomposition allows reducing the SIL of the redundant functions. But, the challenge in applying it does not consist in the redundancy introduction, but in the requirements that need to be respected afterwards. The redundant elements concerned by a decomposition need to verify:
- The absence of common cause failures
- The absence of cascading failures

In order to ensure that during the mapping these requirements are respected, the next step of mapping will take them as inputs. This step provides, thus, the set of constraints with the concerned functions.

On the previous example, the constraint would be to ensure that functions (Acquisition 1, Acquisition 2) are implemented by independent elements where no common cause would lead to the failure of these hosts and the violation of the safety goal.

d Mapping

The mapping of the functional architecture onto a physical element consists of choosing a subsystem (if any) and a possible host for each function. A successful mapping is the set of choices of components and allocations that respects the different constraints. The mapping process sets up also, at a second level, the buses between the subsystems. In the next sections we will detail how each constraint is verified during the mapping.
A mapping solution verifies:

$$\begin{gathered} PA \in S, iff, (i), (iii)\ and\ all\ constraints\ are\ verified \\ (i): \forall\ f \in FA.FS, \exists h\ \in PA\ where\ f \in h.FS \\ (ii): \forall\ fl\ \in FA.FLS, \exists CC\ \in PA.B\ where\ FL\ \in CC.FLS \end{gathered} \quad \{1\}$$

Fig 3 presents an example of a possible mapping solution for the example of Fig 2.

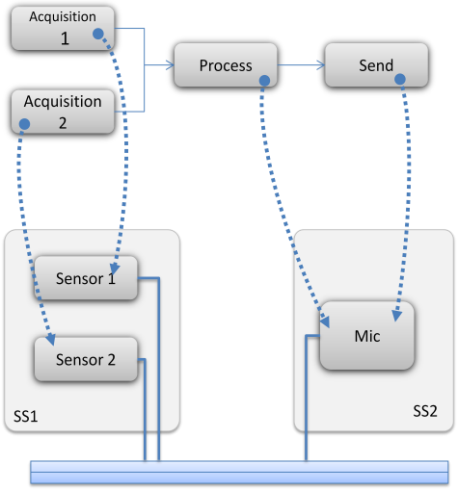

Figure 3. Mapping Example

e Cost Estimation

The mapping process is expected to provide multiple solutions. Making a choice between them can be guided by the cost. We choose to approach the design cost as the cost of the generated physical architecture. A solution's cost is the sum of the hosts cost as well as the subsystems and the buses:

$$cost(PA) = \sum_{SSi \in PA.HS} SSi.C + \sum_{Hi \in PA.HS} Hi.C + \sum_{Bi \in PA.B} Bi.C \quad \{2\}$$

2- Safety Constraints

To ensure that the architecture respects some preliminary ISO 26262 compliance issues we verify that the following checks are good:

- SIL level check :
  The functions are allocated to the hosts. The functions are tagged with an ASIL while the hosts are tagged with the highest ASIL they can reach. A good mapping should verify that functions allocated to each component of the physical architecture respect this SIL constraint:

$$\forall\, ss \in PA(SS), \forall\, h \in ss(H), \forall\, f \in h(F) : \quad F.A \leq H.A \quad \{3\}$$

- Mixed SIL cohabitation:
  The standard allows allocating functions of different SIL to the same component. It requires though ensuring the freedom from interference of low ASIL functions with higher ASIL functions. The freedom from interference is threatened by cascading failures. It is possible to ensure this requirement with built in features in some components. For example, for software functions it is possible to ensure the non interference with good scheduling and presence of Memory Management Unit.
  We consider that a mapping of functions with different ASILs to a component is still correct if this component have the needed features to ensure the non interference. The constraint is verified if:

$$\forall\, SS \in PA.SS, \forall\, H \in SS, if\ \exists\ (i,j)\ where\ i \neq j, (Fi, Fj) \in H.F\ and\ Fi.A = Fj.A \rightarrow H.MSF = true \quad \{4\}$$

- Independence check:
  The decomposition choices at functional level should be respected at physical level. This is translated by the need to map the concerned functions to sufficiently independent elements as required by the standard. From the standard point of view, the independency can be threatened though common cause failures. Consequently, The mapping of concerned function is good if no CCF could be found between their respective hosts.

$$\forall\, (i,j)\ where\ i \neq j\ and\ (Fi, Fj) \in CT.FS : \quad Fi.H \neq Fj.H\ and\ Fi.H.FL.C\ \cap Fj.H.FL.C = \emptyset \quad \{5\}$$

  To ensure the absence of CCF, we start from the MCS over which the decomposition is made. We transform this MCS from the functional level to the physical level by linking the failures modes of the functions to the failure modes of their hosts. The obtained MCS contains the set of hosts' failures modes that jointly lead to the violation of the safety goal. We compare the causes of these failures modes to identify CCFs.

Algorithm:

```
Loop in the Constraint set
        Transform the MCS at functional level to MCS at host level
        Compare the set of causes of the failure modes of the new MCS
        If a common cause exist
                Constraint is not verified
        Else Constraint is verified
        End if
End loop
```

The transformation consists of replacing in the MCS the failure mode at the functional level with its matching failure modes at host level. The matching between the failure modes is based on their impact on the output flow. For example, if a sensing function is allocated to a sensor. The loss of data flow from the acquisition function can be matched with the loss of the sensor output at the physical level.

3- Solving Approach

At the mapping step, the functional model is tagged with the SIL and with the possible subsystems and hosts for each function. These are the inputs to generate a physical level and map the functions onto its components. We have opted for an exhaustive approach allowing finding all the possible solutions.
Each function can be allocated to a subsystem and a host. This allocation must of course verifies and ensure the different constraints developed earlier. Yet, for this allocation to be acceptable, it must also be possible to map the flow.

The approach is described as followed:

```
While subsystems mapping combinations left not checked loop
        Add the subsystems to the physical architecture
        Allocate the functions to the subsystems
        While hosts mapping combinations left not checked loop
                Add the hosts to the subsystems
                Allocate the functions to the corresponding host
                If the constraints are not all verified
                        Go to next combination
                End if
                While flows allocation combinations left not checked loop
                                Allocate the flows to Bus
                                Add the Bus to the architecture
                                If the flows allocation don't respect all constraints
                                        Go to next combination
                                End if
                        Estimate the cost of the physical architecture
                        Add to the solution set
                End loop
        End loop
End loop
```

The flows mapping approach is described as follows:

```
        Loop in the set of the functional flows
                If the target and source are allocated to the same subsystem
                        If not in the same host
                                Add flow to the subsystem
                        End if
                Else
                        add flow to the physical architecture
                End if
        End loop
```

The flows are implemented at subsystem level or at physical architecture level depending on the mapping of the flow exchanging functions. If both of the functions are allocated to different subsystems, a bus is needed to be set up at this level. But in case, they are allocated to the same subsystems and to different hosts the implementation is different. For the functions allocated to the same hosts, no bus is needed since often the communication is ensured by memory access mechanisms.
For safety matters, the flows implementation must also verify the safety constraints in the same way it is done for functions allocation.

Example Use-Case

In this section we use a use case example to illustrate the approach. For this purpose we use a simple example inspired by real world systems. It consists of two services and two safety goals rated ASIL D and ASIL A. The system's main functionalities are to correctly drive an actuator and to display information about the system state. The bad actuation is rated ASIL D. As for the omission of the display, it is rated ASIL A. The functional architecture proposed is described by the following figure:

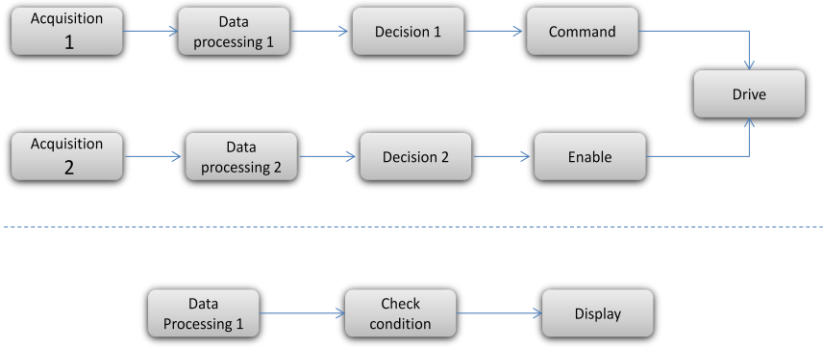

Figure 4. Use-case example with two services

The system consists of a set of 9 functions and 8 flows. We provide a set of 7 hosts with their failure modes and their causes and 3 potential subsystems. We introduced CCFs between some of the hosts and the buses to check the if the efficiency of the approach in respecting the independency constraints.
The decomposition from the first functionality resulted in 16 independence constraints to be verified. Since, ASIL decomposition is made over the two channels, all the function of both channels must be implemented by independent elements eg : acquisition 1 and acquisition 2 must be implemented by independent elements where no common cause will lead to failures modes associated to a bad value.

The ASIL allocation and the possible hosts for each function are as follows:

Function	ASIL	Possible hosts	Possible subsystems
Acquisition 1	B	S1,S2,S3	SS1
Acquisition 2	B	S1,S2,S3	SS1
Data Processing 1	B	C1,C2,C3	SS2,SS3
Data Processing 2	B	C1,C2,C3	SS2,SS3
Decision 1	B	C1,C2,C3	SS2,SS3
Decision 2	B	C1,C2,C3	SS2,SS3
Command	B	C1,C2,C3	SS2,SS3
Enable	B	C1,C2,C3	SS2,SS3
Drive	D	C4	SS3
Check condition	A	C1,C2,C3	SS2
Display	A	C1,C2,C3	SS2

Table 2. Functions definitions

The hosts' properties are specified in the next table. It is to be noted that the failures modes and their causes are not presented in details. For simplicity, we only specify the hosts presenting a CCF that could violate the independency requirement.

Host	ASIL max	Cost
Sensor 1	C	10
Sensor 2	B	8
Sensor 3	B	6
MC 1	A	8
MC 2	B	10
MC 3	C	15
Actuator	-	10

Table 3. Hosts definitions

As mentioned earlier the objective is to find a mapping that respects the constraints from the provided hosts. In our example, multiple solutions were found. The mapping showing the lowest cost is the following:

	Mapped to	
Function	host	subsystem
Acquisition 1	S2	SS1
Acquisition 2	S3	SS1
Data Processing 1	C3	SS2
Data Processing 2	C2	SS2
Decision 1	C3	SS2
Decision 2	C2	SS2
Command	C2	SS3
Enable	C2	SS2
Drive	C4	SS3
Check condition	C3	SS2
Display	C3	SS2

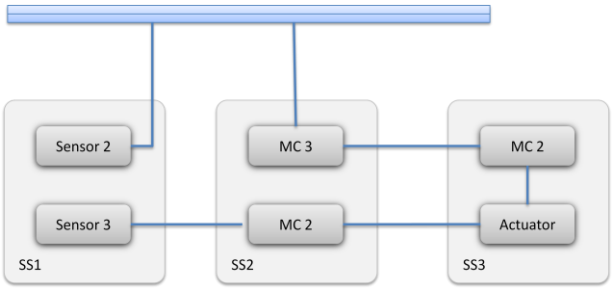

Figure 5. Obtained Solution

The found solution successfully provided an acceptable solution avoiding the CCF of the hosts and the buses. We can see that the two channels are implemented by independent elements. The flows are also implemented with different buses to avoid CCF. The safety levels are also respected for the components choice and during the mapping. The retained solution has a cost of "65" consisting of the cost of the components with the cost of the subsystems and the buses. It is reached in a very short time given the limited size of the example.
We used the approach on larger industrial examples with multiple services and safety goals. Even though for these examples, the exhaustive approach provided acceptable solutions, the scalability is becoming an issue to be addressed in the future works.

Conclusion

In this paper we proposed an approach to the combined synthesis of architecture synthesis and mapping that respects the safety constraints in the context of ISO 26262 and optimizes the cost. This paper proves that the proposed approach can be used efficiently to reach a preliminary architecture for critical automotive systems at the FSC level.

The approach helps to synthesize a physical architecture and map the functions over its elements. It has the advantage of treating at the same time the different services and taking into account the different safety goals. Unlike the manual process where each safety goal is treated separately, it ensures the compatibility of the choices made to deal with each and the respect of the constraints resulting from.

The exhaustive approach implemented and presented earlier helped to retrieve the solutions with the best cost. But with larger examples with multiple services and multiple safety goals, the exhaustive approach seems not adapted to deal with the large possible combinations to be checked. The computation time becomes unpractical especially if multiple ASIL allocations needed to be checked.

For the cost estimation, we used for now only the cost of the components to rate the solutions. This estimation lacks though precision. The software, a considerable part of the design, development cost is not taken into account.

For our next works, we are planning to tackle these limitations by using advanced design space exploration techniques. As for the cost estimation we are planning to implement a software cost estimation model such as COCOMO II. To improve the precision of the solution we plan also to add mechanisms to take into account performance parameters into account such as the CPU, Memory and network load. We will aim also to implement an approach for metrics evaluation to automatically introduce safety mechanisms where needed in the architecture which can impact the final cost of the design.

References

Aleti, A, and S Bjornander. 2009. "ArcheOpterix: An Extendable Tool for Architecture Optimization of AADL Models." … *Embedded Software,* …. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5069138.

Dhouibi, MS, JM Perquis, Laurent Saintis, and Mihaela Barreau. 2014. "Automatic Decomposition and Allocation of Safety Integrity Level Using System of Linear Equations." … *in Complex Systems and* …, no. c: 1–5. http://www.thinkmind.org/index.php?view=article&articleid=pesaro_2014_1_10_60029.

Dömer, Rainer, Andreas Gerstlauer, Junyu Peng, Dongwan Shin, Lukai Cai, Haobo Yu, Samar Abdi, and DanielD Gajski. 2008. “System-on-Chip Environment: A SpecC-Based Framework for Heterogeneous MPSoC Design.” *EURASIP Journal on Embedded Systems* 2008 (1): 647953. doi:10.1155/2008/647953. http://jes.eurasipjournals.com/content/2008/1/647953.

Gerstlauer, Andreas, and Christian Haubelt. 2009. “Electronic System-Level Synthesis Methodologies.” *… -Aided Design of …*, 1–14. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5247153.

Keinert, Joachim, Martin Streubühr, Thomas Schlichter, Joachim Falk, Jens Gladigau, Christian Haubelt, Jürgen Teich, and Michael Meredith. 2009. “SystemCoDesigner—an Automatic ESL Synthesis Approach by Design Space Exploration and Behavioral Synthesis for Streaming Applications.” *ACM Transactions on Design Automation of Electronic Systems* 14 (1): 1–23. doi:10.1145/1455229.1455230. http://portal.acm.org/citation.cfm?doid=1455229.1455230.

Nikolov, H, and M Thompson. 2008. “Daedalus: Toward Composable Multimedia MP-SoC Design.” *… the 45th Annual Design …*, 574–79. http://dl.acm.org/citation.cfm?id=1391615.

Papadopoulos, Yiannis, Martin Walker, David Parker, Erich Rüde, Rainer Hamann, Andreas Uhlig, Uwe Grätz, and Rune Lien. 2011. “Engineering Failure Analysis and Design Optimisation with HiP-HOPS.” *Engineering Failure Analysis* 18 (2): 590–608. doi:10.1016/j.engfailanal.2010.09.025. http://linkinghub.elsevier.com/retrieve/pii/S1350630710001779.